LEON Q. BRIN

Tea Time Numerical Analysis

Experiences in Mathematics, 2^{nd} edition

THE FIRST IN A SERIES OF TEA TIME TEXTBOOKS

SOUTHERN
CONNECTICUT
STATE
UNIVERSITY

To
Victorija, Cecelia, and Amy

Contents

Preface

About Tea Time Numerical Analysis

Greetings! And thanks for giving *Tea Time Numerical Analysis* a read. This textbook was born of a desire to contribute a viable, completely free, introductory Numerical Analysis textbook for instructors and students of mathematics. When this project began (summer 2012), there were traditionally published (very expensive hardcover) textbooks, notably the excellent *Numerical Analysis* by Burden and Faires, which was in its ninth edition. As you might guess by the number of editions, this text is a classic. It is one of very few numerical analysis textbooks geared for the mathematician, not the scientist or engineer. In fact, I studied from an early edition in the mid 1990's! Also in the summer of 2012 there were a couple of freely available websites, notably the popular http://nm.mathforcollege.com/, complete with video lectures. However, no resource I could find included a complete, single-pdf downloadable textbook designed for mathematics classes. To be just that is the ultimate goal of *Tea Time Numerical Analysis*.

The phrase "tea time" is meant to do more than give the book a catchy title. It is meant to describe the general nature of the discourse within. Much of the material will be presented as if it were being told to a student during tea time at University, but with the benefit of careful planning. There will be no big blue boxes highlighting the main points, no stream of examples after a short introduction to a topic, and no theorem. . . proof. . . theorem. . . proof structure. Instead, the necessary terms and definitions and theorems and examples will be woven into a more conversational style. My hope is that this blend of formal and informal mathematics will be easier to digest, and dare I say, students will be more invited to do their reading in this format.

Those who enjoy a more typical presentation might still find this textbook suits their preference to a large extent. There will be a summary of the key concepts at the end of each conversation and a number of the exercises will be solved in complete detail in the appendix. So, one can get a closer-to-typical presentation by scanning for theorems in the conversations, reading the key concepts, and then skipping to the exercises with solutions. I hope most readers won't choose to do so, but it is an option. In any case, the exercises with solutions will be critical reading for most. Learning by example is often the most effective means. After reading a section, or at least scanning it, readers are strongly encouraged to skip to the statements of the exercises with solutions (marked by $^{[\mathbb{S}]}$or $^{[\mathbb{S}]}$), contemplate their solutions, solve them if they can, and then turn to the back of the book for full disclosure. The hope is that, with their placement in the appendix, readers will be more apt to consider solving the exercises on their own before looking at the solutions.

The topical coverage in *Tea Time Numerical Analysis* is fairly typical. The book starts with an introductory chapter, followed by root finding methods, interpolation (part 1), numerical calculus, interpolation (part 2), and the second edition introduces a chapter on differential equations. The first five chapters cover what, at SCSU, constitutes a first semester course in numerical analysis. As this book is intended for use as a free download or an inexpensive print-on-demand volume, no effort has been made to keep the page count low or to spare copious diagrams and colors. In fact, I have taken the inexpensive mode of delivery as liberty to do quite the opposite. I have added many passages and diagrams that are not strictly necessary for the study of numerical analysis, but are at least peripherally related, and may be of interest to some readers. Most of these passages will be presented as digressions, so they will be easy to identify. For example, Taylor's theorem plays such a central role in the subject that not only its statement is presented. Its proof and a bit of history are added as "crumpets". Of course they can be skipped, but are included to provide a more complete understanding of this fundamental theorem of numerical analysis. For another example, as a fan of dynamical systems, I found it impossible to refrain from including a section on visualizing Newton's Method. The powerful and beautiful pictures of Newton's Method as a dynamical system should be eyebrow-raising and question-provoking even if only tangentially important. There are, of course, other examples of somewhat less critical content, but each is there to enhance the reader's understanding or appreciation of the subject, even if the material is not strictly necessary for an introductory study of numerical analysis.

Along the way, implementation of the numerical methods in the form of computer code will also be discussed. While one could simply ignore the programming sections and exercises and still get something out of this text, it is my firm belief that full appreciation for the content can not be achieved without getting ones hands "dirty" by doing some programming. It would be nice if readers have had at least some minimal exposure to programming whether it be Java, or C, web programming, or just about anything else. But I have made every effort to give enough detail so that even those who have never written even a one-line program will be able to participate in this part of the study.

In keeping with the desire to produce a completely free learning experience, GNU Octave was chosen as the programming language for this book. GNU Octave (Octave for short) is offered freely to anyone and everyone! It is free to download and use. Its source code is free to download and study. And anyone is welcome to modify or add to the code if so inclined. As an added bonus, users of the much better-known MATLAB will not be burdened by learning a new language. Octave is a MATLAB clone. By design, nearly any program written in MATLAB will run in Octave without modification. So, if you have access to MATLAB and would prefer to use it, you may do so without worry. I have made considerable effort to ensure that every line of Octave in this book will run verbatim under MATLAB. Even with this earnest effort, though, it is possible that some of the code will not run under MATLAB. It has only been tested in Octave! If you find any code that does not run in MATLAB, please let me know.

I hope you enjoy your reading of *Tea Time Numerical Analysis*. It was my pleasure to write it. Feedback is always welcome.

Leon Q. Brin
brinl1@southernct.edu

Acknowledgments

I gratefully acknowledge the generous support I received during the writing of this textbook, from the patience my immediate family, Amy, Cecelia, and Victorija exercised while I was absorbed by my laptop's screen, to the willingness of my Spring 2013 Seminar class, Elizabeth Field, Rachael Ivison, Amanda Reyher, and Steven Warner to read and criticize an early version of the first chapter. In between, the Woodbridge Public Library staff, especially Pamela Wilonski, helped provide a peaceful and inspirational environment for writing the bulk of the text. Many thanks to Dick Pelosi for his extensive review and many kind words and encouragements throughout the endeavor.

A note on the language agnostic version

First - a huge *thanks* to Leon Brin for producing a great, liberally licensed numerical analysis text! In the spirit of the Creative Commons license, I decided to make some minor changes. I certainly agree that Octave is a natural choice to use in a numerical methods course because it's freely available and it's very easy to transition from Octave to Matlab - which is so widely used in industry. Octave is not, however, the only reasonable choice. In fact, there are many other possibilities including Python, Julia, C, Fortran, and Matlab. Thus, I prefer a text that is language agnostic and have modified the text accordingly.

In my teaching, I'll be using Python and the extensive set of numerical libraries available in the SciPy ecosystem. Sample code is available on my class webpage:

`https://www.marksmath.org/classes/Spring2018NumericalAnalysis/`.

Mark McClure
Department of Mathematics
University of North Carolina at Asheville

Chapter 1

Preliminaries

1.1 Accuracy

Measuring Error

Numerical methods are designed to approximate one thing or another. Sometimes roots, sometimes derivatives or definite integrals, or curves, or solutions of differential equations. As numerical methods produce only approximations to these things, it is important to have some idea how accurate they are. Sometimes accuracy comes down to careful algebraic analysis—sometimes careful analysis of the calculus, and often careful analysis of Taylor polynomials. But before we can tackle those details, we should discuss just how error and, therefore, accuracy are measured.

There are two basic measurements of accuracy: absolute error and relative error. Suppose that p is the value we are approximating, and $\tilde{p}$ is an approximation of p. Then $\tilde{p}$ misses the mark by exactly the quantity $\tilde{p} - p$, the so-called error. Of course, $\tilde{p} - p$ will be negative when $\tilde{p}$ misses low. That is, when the approximation $\tilde{p}$ is less than the exact value p. On the other hand, $\tilde{p} - p$ will be positive when $\tilde{p}$ misses high. But generally, we are not concerned with whether our approximation is too high or too low. We just want to know how far off it is. Thus, we most often talk about the absolute error, $|\tilde{p} - p|$. You might recognize the expression $|\tilde{p} - p|$ as the distance between $\tilde{p}$ and p, and that's not a bad way to think about absolute error.

The absolute error in approximating $p = \pi$ by the rational number $\tilde{p} = \frac{22}{7}$ is $|\frac{22}{7} - \pi| \approx 0.00126$. The absolute error in approximating π^5 by the rational number $\frac{16525}{54}$ is $|\frac{16525}{54} - \pi| \approx 0.00116$. The absolute errors in these two approximations are nearly equal. To make the point more transparent, $\pi \approx 3.14159$ and $\frac{22}{7} \approx 3.14285$, while $\pi^5 \approx 306.01968$ and $\frac{16525}{54} \approx 306.01851$. Each approximation begins to differ from its respective exact value in the thousandths place. And each is off by only 1 in the thousandths place.

But there is something more going on here. π is near 3 while π^5 is near 300. To approximate π accurate to the nearest one hundredth requires the approximation to agree with the exact value in only 3 place values—the ones, tenths, and hundredths. To approximate π^5 accurate to the nearest one hundredth requires the approximation to agree with the exact value in 5 place values—the hundreds, tens, ones, tenths, and hundredths. To use more scientific language, we say that $\frac{22}{7}$ approximates π accurate to 3 significant digits while $\frac{16525}{54}$ approximates π^5 accurate to 5 significant digits. Therein lies the essence of relative errors—weighing the absolute error against the magnitude of the number being approximated. This is done by computing the ratio of the error to the exact value. Hence, the relative error in approximating π by $\frac{22}{7}$ is $\dfrac{|\frac{22}{7} - \pi|}{|\pi|} \approx 4.02(10)^{-4}$ while the relative error in approximating π^5 by $\frac{16525}{54}$ is $\dfrac{|\frac{16525}{54} - \pi^5|}{|\pi^5|} \approx 3.81(10)^{-6}$. The relative errors differ by a factor of about 100 (equivalent to about two significant digits of accuracy) even though the absolute errors are nearly equal. In general, the relative error in approximating p by $\tilde{p}$ is given by $\dfrac{|\tilde{p} - p|}{|p|}$.

Sources of Error

There are two general categories of error. Algorithmic error and floating-point error. Algorithmic error is any error due to the approximation method itself. That is, these errors are unavoidable even if we do exact calculations at every step. Floating-point error is error due to the fact that computers and calculators generally do not do exact arithmetic, but rather do floating-point arithmetic.

Crumpet 1: IEEE Standard 754

Floating-point values are stored in binary. According to the IEEE Standard 754, which most computers use, the mantissa (or significand) is stored using 52 bits, or binary places. Since the leading bit is always assumed to be 1 (and, therefore, not actually stored), each floating point number is represented using 53 consecutive binary place values. Now let's consider how 1/7 is represented exactly. In binary, one seventh is equal to 0.001001001... because $\frac{1}{7} = \sum_{i=1}^{\infty} 2^{-3i} = \frac{1}{8} + \frac{1}{64} + \frac{1}{512} + \cdots$. To see that this is true, remember from calculus that

$$\begin{aligned}\sum_{i=1}^{\infty} 2^{-3i} &= \sum_{i=1}^{\infty} \left(2^{-3}\right)^i \\ &= \frac{2^{-3}}{1-2^{-3}} \\ &= \frac{1/8}{7/8} \\ &= \frac{1}{7}.\end{aligned}$$

But in IEEE Standard 754, $\frac{1}{7}$ is chopped to

$$1.0010010010010010010010010010010010010010010010010010 \times (2)^{-3}$$

or $\sum_{i=1}^{18} 2^{-3i}$ which is exactly $\frac{2573485501354569}{18014398509481984}$. The floating point error in calculating 1/7 is, therefore,

$$\left|\frac{2573485501354569}{18014398509481984} - \frac{1}{7}\right| = \frac{1}{126100789566373888} \approx 7.93(10)^{-18}.$$

References [35, 11]

In floating-point arithmetic, a calculator or computer typically stores its values with about 16 significant digits. For example, in a typical computer or calculator (using double precision arithmetic), the number $\frac{1}{7}$ is stored as about 0.1428571428571428, while the exact value is 0.1428571428571428.... In the exact value, the pattern of 142857 repeats without cease, while in the floating point value, the repetition ceases after the third 8. The value is chopped to 16 decimal places in the floating-point representation. So the floating point error in calculating 1/7 is around $5(10)^{-17}$. I say "around" or "about" in this discussion because these claims are not precisely true, but the point is made. There is a small error in representing 1/7 as a floating point real number. And the same is true about all real numbers save a finite set.

Yes, there is some error in the floating-point representation of real numbers, but it is always small in comparison to the size of the real number being represented. The relative error is around 10^{-17}, so it may seem that the consideration of floating-point error is strictly an academic exercise. After all, what's an error of $7.93(10)^{-18}$ among friends? Is anyone going to be upset if they are sold a ring that is .14285714285714284921 inches wide when it should be .14285714285714285714 inches wide? Clearly not. But it is not only the error in a single calculation (sum, difference, product, or quotient) that you should be worried about. Numerical methods require dozens, thousands, and even millions of computations. Small errors can be compounded. Try the following experiment.

Experiment 1

Use your calculator or computer to calculate the numbers $p_0, p_1, p_2, \ldots, p_7$ as prescribed here:

- $p_0 = \pi$
- $p_1 = 10p_0 - 31$
- $p_2 = 100p_1 - 41$

- $p_3 = 100p_2 - 59$
- $p_4 = 100p_3 - 26$
- $p_5 = 100p_4 - 53$
- $p_6 = 100p_5 - 58$
- $p_7 = 100p_6 - 97$

According to your calculator or computer, p_7 is probably something like 0.93116. However, a little algebra will show that

$$p_7 = 10000000000000\pi - 31415926535897$$

exactly (which is approximately 0.932384). Even though p_0 is a very accurate approximation of π, after just a few (carefully selected) computations, round-off error has caused p_7 to have only one or two significant digits of accuracy!

This experiment serves to highlight the most important cause of floating-point error: subtraction of nearly equal numbers. We repeatedly subtract numbers whose tens and ones digits agree. Their two leading significant digits match. For example, $10\pi - 31 = 31.415926\ldots - 31$. 10π is held accurate to about 16 digits (31.41592653589793) but $10\pi - 31$ is held accurate to only 14 significant digits (0.41592653589793). Each subsequent subtraction decreases the accuracy by two more significant digits. Indeed, p_7 is represented with only 2 significant digits. We have repeatedly subtracted nearly equal numbers. Each time, some accuracy is lost. The error grows.

In computations that don't involve the subtraction of nearly equal quantities, there is the concern of algorithmic error. For example, let $f(x) = \sin x$. Then one can prove from the definition of derivative that

$$f'(1) = \lim_{h\to 0} \frac{\sin(1+h) - \sin(1-h)}{2h}.$$

Therefore, we should expect, in general, that $\tilde{p}(h) = \frac{\sin(1+h)-\sin(1-h)}{2h}$ is a good approximation of $f'(1)$ for small values of h; and that the smaller h is, the better the approximation is.

Experiment 2

Using a calculator or computer, compute $\tilde{p}(h)$ for $h = 10^{-2}$, $h = 10^{-3}$, and so on through $h = 10^{-7}$. Your results should be something like this:

h	$\tilde{p}^*(h)$
10^{-2}	0.5402933008747335
10^{-3}	0.5403022158176896
10^{-4}	0.5403023049677103
10^{-5}	0.5403023058569989
10^{-6}	0.5403023058958567
10^{-7}	0.5403023056738121

The second column is labeled $\tilde{p}^*(h)$ to indicate that the approximation $\tilde{p}(h)$ is calculated using approximate (floating-point) arithmetic, so it is technically an approximation of the approximation. Since $f'(1) = \cos(1) \approx .5403023058681398$, each approximation is indeed reasonably close to the exact value. Taking a closer look, though, there is something more to be said. First, the algorithmic error of $\tilde{p}(10^{-2})$ is

$$\begin{aligned} |\tilde{p}(10^{-2}) - f'(1)| &= \left|50\left(\sin\left(\frac{101}{100}\right) - \sin\left(\frac{99}{100}\right)\right) - \cos(1)\right| \\ &\approx 9.00(10)^{-6} \end{aligned}$$

accurate to three significant digits. That is, if we compute $\tilde{p}(10^{-2})$ using exact arithmetic, the value still misses $f'(1)$ by about $9(10)^{-6}$. The floating-point error is only how far the computed value of $\tilde{p}(10^{-2})$, what we have labeled $\tilde{p}^*(10^{-2})$ in the table, deviates from the exact value of $\tilde{p}(10^{-2})$. That is, the floating-point error is given by $|\tilde{p}^* - \tilde{p}|$:

$$\left|0.5402933008747335 - 50\left(\sin\left(\frac{101}{100}\right) - \sin\left(\frac{99}{100}\right)\right)\right| \approx 1.58(10)^{-17},$$

as small as one could expect. The absolute error $|\tilde{p}^*(10^{-2}) - f'(1)| = |0.5402933008747335 - \cos(1)|$ is essentially all algorithmic. The round-off error is dwarfed by the algorithmic error. The fact that we have used floating-point arithmetic is negligible.

On the other hand, the algorithmic error of $\tilde{p}(10^{-7})$ is

$$\begin{aligned} |\tilde{p}(10^{-7}) - f'(1)| &= \left|5000000\left(\sin\left(\frac{10000001}{10000000}\right) - \sin\left(\frac{9999999}{10000000}\right)\right) - \cos(1)\right| \\ &\approx 9.00(10)^{-16} \end{aligned}$$

accurate to three significant digits. But we should be a little bit worried about the floating-point error since $\sin\left(\frac{10000001}{10000000}\right) \approx 0.8414710388$ and $\sin\left(\frac{9999999}{10000000}\right) \approx .8414709307$ are nearly equal. We are subtracting numbers whose five leading significant digits match! Indeed, the floating-point error is, again $|\tilde{p}^* - \tilde{p}|$, or

$$\left|0.5403023056738121 - 5000000\left(\sin\left(\frac{10000001}{10000000}\right) - \sin\left(\frac{9999999}{10000000}\right)\right)\right| \approx 1.94(10)^{-10}.$$

Perhaps this error seems small, but it is very large compared to the algorithmic error of about $9(10)^{-16}$. So, in this case, the error is essentially all due to the fact that we are using floating-point arithmetic! This time, the algorithmic error is dwarfed by the round-off error. Luckily, this will not often be the case, and we will be free to focus on algorithmic error alone.

Crumpet 2: Chaos

Edward Lorenz, a meteorologist at the Massachusetts Institute of Technology, was among the first to recognize and study the mathematical phenomenon now called chaos. In the early 1960's he was busy trying to model weather systems in an attempt to improve weather forecasting. As one version of the story goes, he wanted to repeat a calculation he had just made. In an effort to save some time, he used the same initial conditions he had the first time, only rounded off to three significant digits instead of six. Fully expecting the new calculation to be similar to the old, he went out for a cup of coffee and came back to look. To his astonishment, he noticed a completely different result! He repeated the procedure several times, each time finding that small initial variations led to large long-term variations. Was this a simple case of floating-point error? No. Here's a rather simplified version of what happened. Let $f(x) = 4x(1-x)$ and set $p_0 = 1/7$. Now compute $p_1 = f(p_0)$, $p_2 = f(p_1)$, $p_3 = f(p_2)$, and so on until you have $p_{40} = f(p_{39})$. You should find that $p_{40} \approx 0.080685$. Now set $p_0 = 1/7 + 10^{-12}$ (so we can run the same computation only with an initial value that differs from the original by the tiny amount, 10^{-12}). Compute as before, $p_1 = f(p_0)$, $p_2 = f(p_1)$, $p_3 = f(p_2)$, and so on until you have $p_{40} = f(p_{39})$. This time you should find that $p_{40} \approx 0.91909$—a completely different result! If you go back and run the two calculations using 100 significant digit arithmetic, you will find that beginning with $p_0 = 1/7$ leads to $p_{40} \approx .080736$ while beginning with $p_0 = 1/7 + 10^{-12}$ leads to $p_{40} \approx 0.91912$. In other words, it is not the fact that we are using floating-point approximations that makes these two computations turn out drastically different. Using 1000 significant digit arithmetic would not change the conclusion, nor would any more precise calculation. This is a demonstration of what's known as sensitivity to initial conditions, a feature of all chaotic systems including the weather. Tiny variations at some point lead to vast variations later on. And the "errors" are algorithmic. This is the basic principle that makes long-range weather forecasting impossible. In the words of Edward Lorenz, "In view of the inevitable inaccuracy and incompleteness of weather observations, precise very-long-range forecasting would seem non-existent."

References [19, 14, 4]

Experiment 3

Let $a = 77617$ and $b = 33096$, and compute

$$333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}.$$

You will probably get a number like $-1.180591620717411(10)^{21}$ even though the exact value is

$$-\frac{54767}{66192} \approx -.8273960599468214.$$

That's an incredible error! But it's not because your calculator or computer has any problem calculating each term to a reasonable degree of accuracy. Try it.

$$\begin{aligned}
333.75b^6 &= 438605750846393161930703831040 \\
a^2(11a^2b^2 - b^6 - 121b^4 - 2) &= -7917111779274712207494296632228773890 \\
5.5b^8 &= 7917111340668961361101134701524942848 \\
\frac{a}{2b} &= \frac{77617}{66192} \approx 1.172603940053179
\end{aligned}$$

The reason the calculation is so poor is that nearly equal values are subtracted after each term is calculated. $a^2(11a^2b^2 - b^6 - 121b^4 - 2)$ and $5.5b^8$ have opposite signs and match in their greatest 7 significant digits, so calculating their sum decreases the accuracy by about 7 significant digits. To make matters worse, $a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 = -438605750846393161930703831042$, which has the opposite sign of $333.75b^6$ and matches it in every place value except the ones. That's 29 digits! So we lose another 29 significant digits of accuracy in adding this sum to $333.75b^6$. Doing the calculation exactly, the sum $333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8$ is -2. But the computation needs to be carried out to 37 significant digits to realize this. Calculation using only about 16 significant digits, as most calculators and computers do, results in 0 significant digits of accuracy since 36 digits of accuracy are lost during the calculation. That's why you can get a number like $-1.180591620717411(10)^{21}$ for your final answer instead of the exact answer $\frac{a}{2b} - 2 \approx -.8273960599468214$.

What may be even more surprising is that a simple rearrangement of the expression leads to a completely different result. Try computing

$$(333.75 - a^2)b^6 + a^2(11a^2b^2 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

instead. This time you will likely get a number like 1.172603940053179. Again the result is entirely inaccurate, and the reason is the same. This time the individual terms are

$$\begin{aligned}
(333.75 - a^2)b^6 &= -7917110903377385049079188237280149504 \\
a^2(11a^2b^2 - 121b^4 - 2) &= -437291576312021946464244793346 \\
5.5b^8 &= 7917111340668961361101134701524942848 \\
\frac{a}{2b} &= \frac{77617}{66192} \approx 1.172603940053179
\end{aligned}$$

so the problem persists. We still end up subtracting numbers of nearly equal value. The difference between this calculation and the last is rounding. In the first case, rounding caused two of the large numbers to disagree in their last significant digit, so they added up to something huge. In the second case, the sum of the first three terms turns out to be 0 because the large numbers agree in all significant digits. Note that in the second case, the final result is simply the value of $\frac{a}{2b}$.

As these examples show, sometimes floating-point error and sometimes algorithmic error can spoil a calculation. In general, it is very difficult to catch floating-point error, though. Algorithmic error is much more accessible. And most of the algorithms we will explore are not susceptible to floating point error. In almost all cases, the lion's share of the error will be algorithmic.

References [28, 18]

Key Concepts

p The exact value being approximated.

$\tilde{p}$ An approximation of the value p.

Absolute error: $|\tilde{p} - p|$ is known as the absolute error in using $\tilde{p}$ to approximate the value p.

Relative error: $\dfrac{|\tilde{p} - p|}{|p|}$ is known as the relative error in using $\tilde{p}$ to approximate the value p.

Accuracy: We say that $\tilde{p}$ is accurate to n significant digits if the leading n significant digits of $\tilde{p}$ match those of p. More precisely, we say that $\tilde{p}$ is accurate to $d(\tilde{p}) = \log \left| \frac{p}{\tilde{p}-p} \right|$ significant digits.

Floating-point arithmetic: Arithmetic using numbers represented by a fixed number of significant digits.

Algorithmic error: Error caused solely by the algorithm or equation involved in the approximation, $|\tilde{p}-p|$ where $\tilde{p}$ is an approximation of p and is computed using exact arithmetic.

Truncation error: Algorithmic error due to use of a partial sum in place of a series. In this type of error, the tail of the series is truncated—thus the name.

Floating-point error: Error caused solely by the fact that a computation is done using floating-point arithmetic, $|\tilde{p}^* - \tilde{p}|$ where $\tilde{p}^*$ is computed using floating-point arithmetic, $\tilde{p}$ is computed using exact arithmetic, and both are computed according to the same formula or algorithm.

Round-off error: Another name for floating-point error.

Exercises

1. Besides round-off error, how may the accuracy of a numerical calculation be adversely affected?

2. Compute the absolute and relative errors in the approximation of π by 3.

3. Calculate the absolute error in approximating p by $\tilde{p}$.

   (a) $p = 123; \quad \tilde{p} = \frac{1106}{9}$ [S]

   (b) $p = \frac{1}{e}; \quad \tilde{p} = .3666$

   (c) $p = 2^{10}; \quad \tilde{p} = 1000$ [S]

   (d) $p = 24; \quad \tilde{p} = 48$

   (e) $p = \pi^{-7}; \quad \tilde{p} = 10^{-4}$ [S]

   (f) $p = (0.062847)(0.069234); \quad \tilde{p} = 0.0042$

4. Calculate the relative errors in the approximations of question 3. [S]

5. How many significant digits of accuracy do the approximations of question 3 have? [S]

6. Compute the absolute error and relative error in approximations of p by $\tilde{p}$.

   (a) $p = \sqrt{2}$, $\tilde{p} = 1.414$

   (b) $p = 10^{\pi}$, $\tilde{p} = 1400$

   (c) $p = 9!$, $\tilde{p} = \sqrt{18\pi}(9/e)^9$

7. Calculate $\dfrac{1103\sqrt{8}}{9801}$ using the computer.

8. The number in question 7 is an approximation of $1/\pi$. Using the computer, find the absolute and relative errors in the approximation.

9. Using the computer, calculate

   (a) $\lfloor \ln(234567) \rfloor$

   (b) $e^{\lceil \ln(234567) \rceil}$

   (c) $\sqrt[3]{\lfloor \sin(e^{5.2}) \rfloor}$

   (d) $-e^{i\pi}$

   (e) $4\tan^{-1}(1)$

   (f) $\dfrac{\lfloor \cos(3) - \sqrt[5]{\ln(3)} \rfloor}{\lceil \arctan(3) - e^3 \rceil}$

10. Find $f(2)$ using the computrer.

    (a) $f(x) = e^{\sin(x)}$ [S]

    (b) $f(x) = \sin(e^x)$

    (c) $f(x) = \tan^{-1}(x - 0.429)$ [S]

    (d) $f(x) = x - \tan^{-1}(0.429)$

    (e) $f(x) = 10^x/5!$ [A]

    (f) $f(x) = 5!/x^{10}$

11. All of these equations are mathematically true. Nonetheless, floating point error causes some of them to be false according to the computer. Which ones? HINT: Use the boolean operator `==` to check. For example, to check if $\sin(0) = 0$, type `sin(0)==0` into the computer. `ans=1` means true (the two sides are equal according to the computer—no round-off error) and `ans=0` means false (the two sides are not equal according to the computer—round-off error).

    (a) $(2)(12) = 9^2 - 4(9) - 21$

    (b) $e^{3\ln(2)} = 8$

    (c) $\ln(10) = \ln(5) + \ln(2)$

    (d) $g(\frac{1+\sqrt{5}}{2}) = \frac{1+\sqrt{5}}{2}$ where $g(x) = \sqrt[3]{x^2 + x}$

    (e) $\lfloor 153465/3 \rfloor = 153465/3$

    (f) $3\pi^3 + 7\pi^2 - 2\pi + 8 = ((3\pi + 7)\pi - 2)\pi + 8$

12. Find an approximation $\tilde{p}$ of p with absolute error .001.

    (a) $p = \pi$ [S]

    (b) $p = \sqrt{5}$

    (c) $p = \ln(3)$ [S]

    (d) $p = \sqrt{23}^{\sqrt{23}}$

    (e) $p = \frac{10}{\ln(1.1)}$ [S]

    (f) $p = \tan(1.57079)$

13. Find an approximation $\tilde{p}$ of p with relative error .001 for each value of p in question 12. [S]

14. $\tilde{p}$ approximates what value with absolute error .0005?

    (a) $\tilde{p} = .2348263818643$ [A]

(b) $\tilde{p} = 23.89627345677$

(c) $\tilde{p} = -8.76257664363$

15. Repeat question 14 except with relative error .0005. [𝔸]

16. $\tilde{p}$ approximates p with absolute error $\frac{1}{100}$ and relative error $\frac{3}{100}$. Find p and $\tilde{p}$. [𝔸]

17. $\tilde{p}$ approximates p with absolute error $\frac{3}{100}$ and relative error $\frac{1}{100}$. Find p and $\tilde{p}$.

18. Suppose $\tilde{p}$ must approximate p with relative error at most 10^{-3}. Find the largest interval in which $\tilde{p}$ must lie if $p = 900$.

19. The number e can be defined by $e = \sum_{n=0}^{\infty}(1/n!)$. Compute the absolute error and relative error in the following approximations of e:

    (a) $\displaystyle\sum_{n=0}^{5} \frac{1}{n!}$

    (b) $\displaystyle\sum_{n=0}^{10} \frac{1}{n!}$

20. The golden ratio, $\dfrac{1+\sqrt{5}}{2}$, is found in nature and in mathematics in a variety of places. For example, if F_n is the n^{th} Fibonacci number, then

$$\lim_{n\to\infty} \frac{F_{n+1}}{F_n} = \frac{1+\sqrt{5}}{2}$$

Therefore, F_{11}/F_{10} may be used as an approximation of the golden ratio. Find the relative error in this approximation. HINT: The Fibonacci sequence is defined by $F_0 = 1$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$.

21. Find values for p and $\tilde{p}$ so that the relative and absolute errors are equal. Make a general statement about conditions under which this will happen. [𝔸]

22. Find values for p and $\tilde{p}$ so that the relative error is greater than the absolute error. Make a general statement about conditions under which this will happen.

23. Find values for p and $\tilde{p}$ so that the relative error is less than the absolute error. Make a general statement about conditions under which this will happen.

24. Calculate (i) $\tilde{p}^*$ using a calculator or computer, (ii) the absolute error, $|\tilde{p}^* - p|$, and (iii) the relative error, $\frac{|\tilde{p}^* - p|}{|p|}$. Then use the given value of $\tilde{p}$ to compute (iv) the algorithmic error, $|\tilde{p} - p|$ and (v) the round-off error, $|\tilde{p}^* - \tilde{p}|$.

    (a) Let $f(x) = x^4 + 7x^3 - 63x^2 - 295x + 350$ and let $p = f'(-2)$. The value $\tilde{p} = \frac{f(-2+10^{-7}) - f(-2-10^{-7})}{2(10)^{-7}}$ is a good approximation of p. $\tilde{p}$ is exactly 8.99999999999999. [𝔸]

    (b) Let $f'(x) = e^x \sin(10x)$ and $f(0) = 0$ and let $p = f(1)$. It can be shown that $p = \frac{1}{101}e(\sin 10 - 10\cos 10) + \frac{10}{101}$. Euler's method produces the approximation $\tilde{p} = \frac{1}{10}\sum_{i=1}^{10} e^{i/10}\sin i$. Accurate to 28 significant digits, $\tilde{p}$ is 0.2071647018159241499410798569.

    (c) Let $a_0 = \frac{5+\sqrt{5}}{8}$ and $a_{n+1} = 4a_n(1-a_n)$, and consider $p = a_{51}$. It can be shown that $p = a_{51} = \frac{5-\sqrt{5}}{8}$. The most direct algorithm for calculating a_{51} is to calculate $a_1, a_2, a_3, \ldots a_{51}$ in succession, according to the given recursion relation. Use this algorithm to compute $\tilde{p}^*$ and $\tilde{p}$.

1.2 Taylor Polynomials

One of the cornerstones of numerical analysis is Taylor's theorem about which you learned in Calculus. A short study bears repeating here, however.

Theorem 1. *Suppose that $f(x)$ has $n+1$ derivatives on (a,b), and $x_0 \in (a,b)$. Then for each $x \in (a,b)$, there exists a ξ, depending on x, lying strictly between x and x_0 such that*

$$f(x) = f(x_0) + \sum_{j=1}^{n} \left(\frac{f^{(j)}(x_0)}{j!}(x-x_0)^j \right) + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}.$$

Proof. Let I be the open interval between x and x_0 and $\overline{I}$ be the closure of I. Since $I \subset \overline{I} \subset (a,b)$ and f has $n+1$ derivatives on (a,b), we have that $f, f', f'', \ldots, f^{(n)}$ are all continuous on $\overline{I}$ and that $f^{(n+1)}$ exists on I. We now define

$$F(z) = f(x) - f(z) - \sum_{j=1}^{n} \frac{f^{(j)}(z)}{j!}(x-z)^j$$

and will prove the theorem by showing that $F(x_0) = \frac{(x-x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$ for some $\xi \in I$. Note that $F'(z)$, a telescoping sum, is given by

$$\begin{aligned} F'(z) &= -f'(z) - \sum_{j=1}^{n} \left[\frac{f^{(j+1)}(z)}{j!}(x-z)^j - \frac{f^{(j)}(z)}{(j-1)!}(x-z)^{j-1} \right] \\ &= -f'(z) - \left[\frac{f^{(n+1)}(z)}{n!}(x-z)^n - f'(z) \right] \\ &= -\frac{f^{(n+1)}(z)}{n!}(x-z)^n. \end{aligned}$$

Now define $g(z) = F(z) - \left(\frac{x-z}{x-x_0}\right)^{n+1} F(x_0)$. It is easy to verify that g satisfies the premises of Rolle's theorem. Indeed, $g(x_0) = g(x) = 0$ and the continuity and differentiability criteria are met. By Rolle's theorem, there exists $\xi \in I$ such that $g'(\xi) = 0 = F'(\xi) + (n+1)\frac{(x-\xi)^n}{(x-x_0)^{n-1}} F(x_0)$. Hence,

$$\begin{aligned} F(x_0) &= -F'(\xi) \frac{(x-x_0)^{n+1}}{(n+1)(x-\xi)^n} \\ &= \frac{f^{(n+1)}(\xi)}{n!(n+1)}(x-x_0)^{n+1} \\ &= \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}. \end{aligned}$$

This completes the proof. □

We will use the notation

$$T_n(x) = f(x_0) + \sum_{j=1}^{n} \left(\frac{f^{(j)}(x_0)}{j!}(x-x_0)^j \right)$$

and call this the n^{th} Taylor polynomial of f expanded about x_0. We will also use the notation

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}$$

and call this the remainder term for the n^{th} Taylor polynomial of f expanded about x_0.

Crumpet 3: ξ

ξ is the (lower case) fourteenth letter of the Greek alphabet and is pronounced `ksee`. It is customary, but, of course, not necessary to use this letter for the unknown quantity in Taylor's theorem. The capital version of ξ is Ξ, a symbol rarely seen in mathematics.

It will not be uncommon, for sake of brevity, to call $T_n(x)$ the n^{th} Taylor polynomial and $R_n(x)$ the remainder term when the function and center of expansion, x_0, are either unspecified or clear from context.

In calculus, you likely focused on the Taylor polynomial, or Taylor series, and did not pay much attention to the remainder term. The situation is quite the reverse in numerical analysis. Algorithmic error can often be ascertained by careful attention to the remainder term, making it more critical than the Taylor polynomial itself. The Taylor polynomial will, however, be used to derive certain methods, so won't be entirely neglected.

The most important thing to understand about the remainder term is that it tells us precisely how well $T_n(x)$ approximates $f(x)$. From Taylor's theorem, $f(x) = T_n(x) + R_n(x)$, so the absolute error in using $T_n(x)$ to approximate $f(x)$ is given by $|T_n(x) - f(x)| = |R_n(x)|$. But $|R_n(x)| = \left|\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}\right|$ for some ξ between x and x_0. Therefore,

$$\begin{aligned} |T_n(x) - f(x)| = |R_n(x)| &\leq \max_\xi \left|\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}\right| \\ &= \frac{|x-x_0|^{n+1}}{(n+1)!}\max_\xi \left|f^{(n+1)}(\xi)\right|. \end{aligned}$$

We learn several things from this observation:

1. The remainder term is precisely the error in using $T_n(x)$ to approximate $f(x)$. Hence, it is sometimes referred to as the error term.

2. The absolute error in using $T_n(x)$ to approximate $f(x)$ depends on three factors:

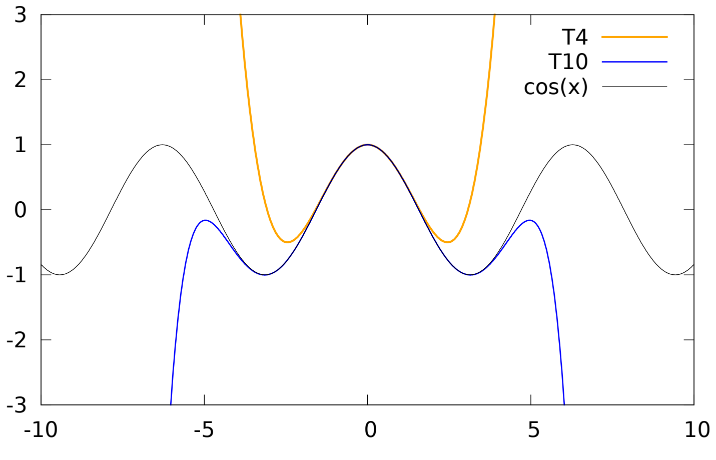
   (a) $|x - x_0|^{n+1}$

   (b) $\frac{1}{(n+1)!}$

   (c) $|f^{(n+1)}(\xi)|$

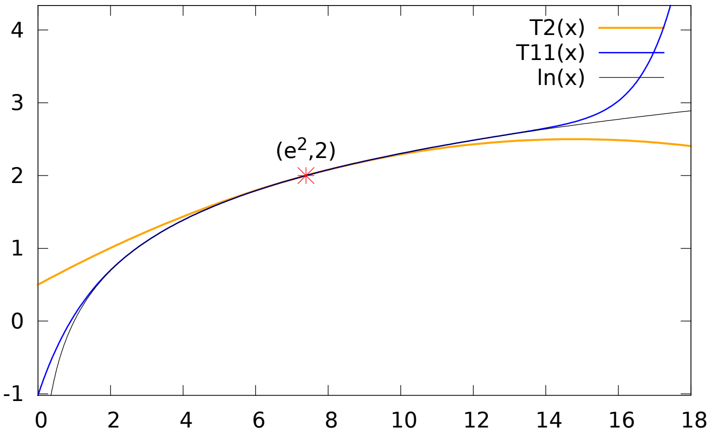
3. We can find an upper bound on $|T_n(x) - f(x)|$ by finding an upper bound on $\left|f^{(n+1)}(\xi)\right|$.

Figure 1.2.1: For small n, $T_n(x)$ is a good approximation only for small x.



Because $|R_n(x)|$ measures exactly the absolute error $|T_n(x) - f(x)|$, we will be interested in conditions that force $|R_n(x)|$ to be small. According to observation 2, there are three quantities to consider. First, $|x-x_0|^{n+1}$, or $|x-x_0|$, the distance between x and x_0. The approximation $T_n(x)$ will generally be better for x closer to x_0. Second, $\frac{1}{(n+1)!}$. This suggests that the more terms we use in our Taylor polynomial (the greater n is), the better the approximation will be. Finally, $|f^{(n+1)}(\xi)|$, the magnitude of the $(n+1)^{st}$ derivative of f. The tamer this derivative, the better $T_n(x)$ will approximate $f(x)$. Be warned, however, these are just rules of thumb for making $|R_n(x)|$ small. There are exceptions to these rules.

Figure 1.2.2: The actual error $|T_n(x) - f(x)|$ is often much smaller than the theoretical bound.



To see these factors in action, consider $f(x) = \ln(x)$ expanded about $x_0 = e^2$. According to Taylor's theorem,

$$T_2(x) = 2 + \frac{x-e^2}{e^2} - \frac{(x-e^2)^2}{2e^4} \quad \text{and} \quad R_2(x) = \frac{1}{3\xi^3}(x-e^2)^3;$$

$$T_{11}(x) = 2 + \sum_{j=1}^{11}\left(\frac{(-1)^{j-1}(x-e^2)^j}{je^{2j}}\right) \quad \text{and} \quad R_{11}(x) = \frac{-1}{12\xi^{12}}(x-e^2)^{12}.$$

After you have convinced yourself these formulas are correct, suppose that we are interested in approximating $\ln(x)$ with an absolute error of no more than 0.1. Since $|\xi^{-3}|$ and $|\xi^{-12}|$ are decreasing functions of ξ, they attain their maximum values on a closed interval at the lower endpoint of that interval. Hence, for $x \geq e^2$, we have $|R_2(x)| \leq \max_{\xi\in[e^2,x]}\left|\frac{1}{3\xi^3}(x-e^2)^3\right| = \frac{1}{3e^6}(x-e^2)^3$. But for $0 < x < e^2$, we have $|R_2(x)| \leq \max_{\xi\in[x,e^2]}\left|\frac{1}{3\xi^3}(x-e^2)^3\right| = \frac{1}{3x^3}(e^2-x)^3$. To determine where these remainders are less than 0.1, we need to solve the equations $\frac{1}{3e^6}(x-e^2)^3 = 0.1$ and $\frac{1}{3x^3}(e^2-x)^3 = 0.1$. The values we seek are $x = \left(1+\sqrt[3]{\frac{3}{10}}\right)e^2 \approx 12.33$ and $x = \frac{\sqrt[3]{8100}+10\sqrt[3]{90}-30}{13\sqrt[3]{90}}e^2 \approx 4.427$. So Taylor's theorem guarantees that $T_2(x)$ will approximate $\ln(x)$ to within 0.1 over the entire interval $[4.427, 12.33]$. Since $e^2 \approx 7.389$, $T_2(x)$ approximates $\ln(x)$ to within 0.1 from about 3 below e^2 to about 5 above e^2. In other words, as long as x is close enough to $x_0 = e^2$, the approximation is good. A similar calculation for $R_{11}(x)$ reveals that $T_{11}(x)$ is guaranteed to approximate $\ln(x)$ to within 0.1 over the interval $[3.667, 14.89]$. In other words, for a larger value of n, x doesn't need to be as close to x_0 to achieve the same accuracy.

But remember, these are only theoretical bounds on the errors. The actual errors are often much smaller than the bounds. For example, our analysis gives the upper bound $|R_2(3)| \leq \frac{1}{3\cdot 3^3}(e^2-3)^3 \approx 1.05$ where the actual error, $|T_2(3) - \ln(3)| = \left|2 + \frac{3-e^2}{e^2} - \frac{(3-e^2)^2}{2e^4} - \ln(3)\right| \approx .131$. The bound is about 8 times the actual error. If we take this point a bit further, the graphs of $T_2(x)$ and $T_{11}(x)$ versus $\ln(x)$ (and a bit of calculation we will discuss later) reveal that $T_2(x)$ actually approximates $\ln(x)$ to within 0.1 over the interval $[3.296, 13.13]$ and $T_{11}(x)$ actually approximates $\ln(x)$ to within 0.1 over the interval $[0.9030, 15.33]$. These intervals are a bit larger than the theoretical guaranteed intervals. See Figure 1.2.2. This figure reveals something else too. $T_2(18)$ does a much better job of approximating $\ln(18)$ than does $T_{11}(18)$. It's not always the case that more terms means a better approximation.

We now turn our attention to perhaps the most often analyzed Taylor polynomials-–those for the sine and cosine functions. They provide examples with beautiful visualization and simple analysis. The n^{th} Taylor polynomial for $f(x) = \cos(x)$ expanded about 0 is

$$\begin{aligned} T_n(x) &= \cos(0) + \sum_{j=1}^{n}\left(\frac{\frac{d^j}{dx^j}(\cos(x))\Big|_{x=0}}{j!}(x-0)^j\right) \\ &= \cos(0) - \sin(0)\cdot x - \frac{\cos(0)}{2}x^2 + \frac{\sin(0)}{6}x^3 + \frac{\cos(0)}{24}x^4 - \cdots \\ &= 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \cdots \end{aligned}$$

and its remainder term is

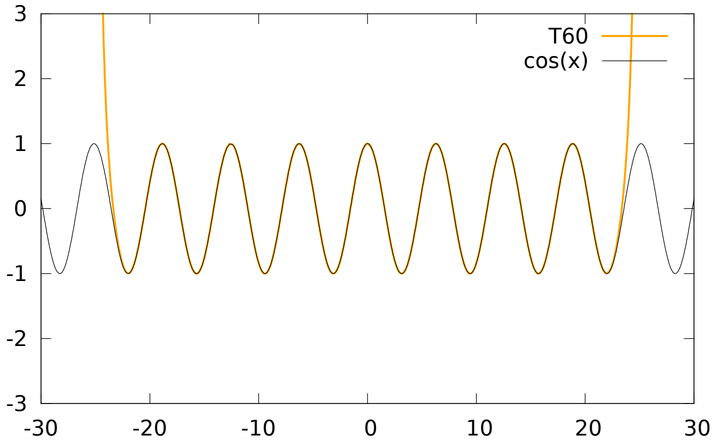
$$\begin{aligned} R_n(x) &= \frac{\frac{d^{n+1}}{dx^{n+1}}(\cos(x))\Big|_{x=\xi}}{(n+1)!}(x-0)^{n+1} \\ &= \frac{x^{n+1}}{(n+1)!}\begin{cases} -\sin(\xi) & \text{when } n \bmod 4 \equiv 0 \\ -\cos(\xi) & \text{when } n \bmod 4 \equiv 1 \\ \sin(\xi) & \text{when } n \bmod 4 \equiv 2 \\ \cos(\xi) & \text{when } n \bmod 4 \equiv 3 \end{cases}. \end{aligned}$$

Since the sine and cosine functions are bounded between -1 and 1 we know that

$$-\frac{|x|^{n+1}}{(n+1)!} \le R_n(x) \le \frac{|x|^{n+1}}{(n+1)!}.$$

There are two ways this remainder term will be small. First, if x is close to 0, then $|x|$ is small, making $R_n(x)$ small. Second, if n is large, then $\frac{1}{(n+1)!}$ is small, making $R_n(x)$ small. In other words, for small values of n, the remainder term is small for small values of x. $T_n(x)$ is a good approximation of $\cos(x)$ for such combinations of x and n. On the other hand, for large values of n, the remainder term is small even for large values of x. For example, $|R_{61}(x)| \le \frac{|x|^{62}}{62!}$, so $|R_{61}(x)|$ will remain less than 1 for all x with magnitude less than $\sqrt[62]{62!} \approx 23.933$. Figures 1.2.1 and 1.2.3 illustrate these points.

Figure 1.2.3: For large n, $T_n(x)$ is a good approximation even for large x.



Key Concepts

Rolle's theorem: Suppose that $f(x)$ is continuous on $[a,b]$ and differentiable on (a,b). If $f(a)=f(b)$, then there exists $\xi \in (a,b)$ such that $f'(\xi)=0$.

Taylor's theorem: Suppose that $f(x)$ has $n+1$ derivatives on (a,b), and $x_0 \in (a,b)$. Then for each $x \in (a,b)$, there exists ξ, depending on x, lying strictly between x and x_0 such that

$$f(x) = f(x_0) + \sum_{j=1}^{n}\left(\frac{f^{(j)}(x_0)}{j!}(x-x_0)^j\right) + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}.$$

n^{th} **Taylor polynomial:** $T_n(x) = f(x_0) + \sum_{j=1}^{n}\left(\frac{f^{(j)}(x_0)}{j!}(x-x_0)^j\right)$.

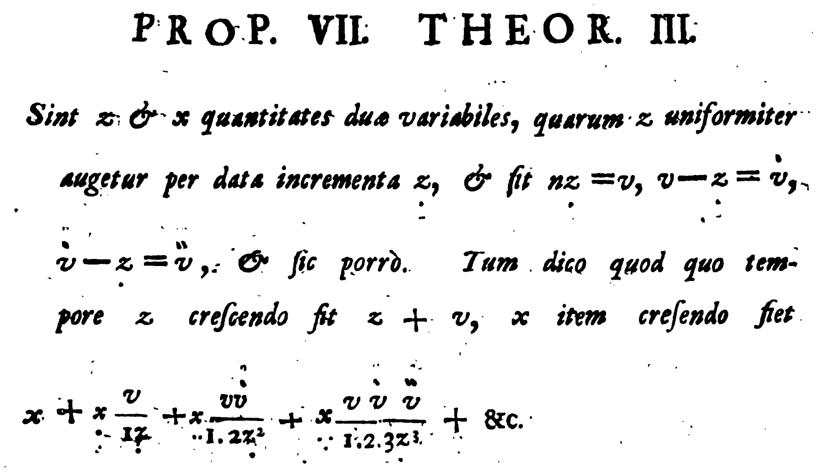
Maclaurin polynomial: A Taylor polynomial expanded about $x_0 = 0$ is also called a Maclaurin polynomial.

Remainder term: $R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}$ is precisely $-(T_n(x)-f(x))$.

Error term: Another name for the remainder term.

Crumpet 4: The original theorem of Brook Taylor

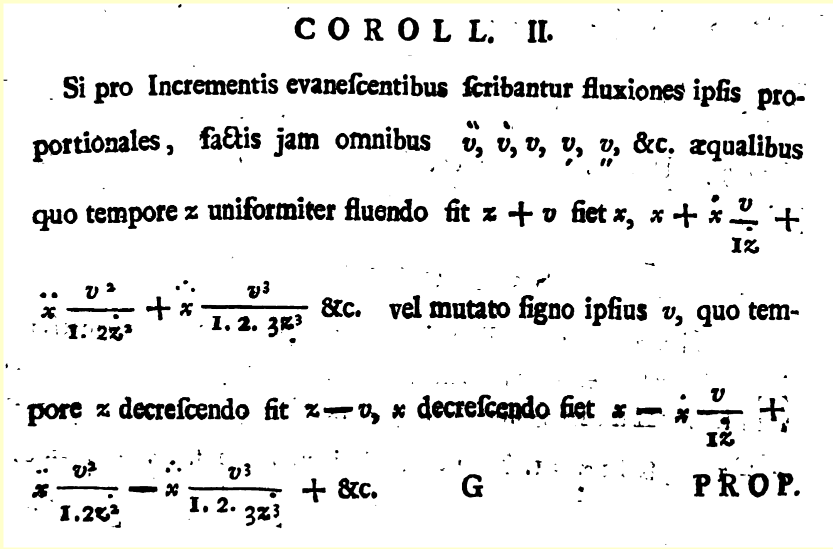
The original theorem of Brook Taylor was published in his opus magnum *Methodus Incrementorum Directa & Inversa* of 1715. In *Methodus*, it appears as the second corollary to Proposition VII Theorem III, bearing faint resemblance to any modern statement of the theorem.



PROP. VII. THEOR. III.

Sint z & x quantitates duæ variabiles, quarum z uniformiter augetur per data incrementa $\underset{\cdot}{z}$, & ſit $n\underset{\cdot}{z} = v$, $v - \underset{\cdot}{z} = \dot{v}$, $\dot{v} - \underset{\cdot}{z} = \ddot{v}$, & ſic porrò. Tum dico quod quo tempore z creſcendo fit $z + v$, x item creſendo fiet

$$x + \underset{\cdot}{x}\frac{v}{1\underset{\cdot}{z}} + \underset{\cdot\cdot}{x}\frac{v\dot{v}}{1.2\underset{\cdot}{z}^2} + \underset{\cdot\cdot\cdot}{x}\frac{v\,\dot{v}\,\ddot{v}}{1.2.3\underset{\cdot}{z}^3} + \&c.$$



COROLL. II.

Si pro Incrementis evaneſcentibus ſcribantur fluxiones ipſis proportionales, factis jam omnibus $\ddot{v}, \dot{v}, v, \underset{\prime}{v}, \underset{\prime\prime}{v}$, &c. æqualibus quo tempore z uniformiter fluendo fit $z + v$ fiet x, $x + \dot{x}\frac{v}{1\dot{z}} + \ddot{x}\frac{v^2}{1.2\dot{z}^2} + \dddot{x}\frac{v^3}{1.2.3\dot{z}^3}$ &c. vel mutato ſigno ipſius v, quo tempore z decreſcendo fit $z - v$, x decreſcendo fiet $x - \dot{x}\frac{v}{1\dot{z}} + \ddot{x}\frac{v^2}{1.2\dot{z}^2} - \dddot{x}\frac{v^3}{1.2.3\dot{z}^3}$ + &c.

G PROP.

There is no mention of a remainder term. There is no use of the familiar $f(x)$-type function notation. It's written in Latin. And there is no laundry list of hypotheses.

Here is the original statement of Taylor's theorem in English as translated by Ian Bruce. Proposition VII. Theorem III: There are two variable quantities, z & x, of which z is regularly increased by the given increment $\underset{\cdot}{z}$, and $n\underset{\cdot}{z} = v$, $v - \underset{\cdot}{z} = \overset{\backslash}{v}$, $\overset{\backslash}{v} - \underset{\cdot}{z} = \overset{\backslash\backslash}{v}$, and thus henceforth. Moreover, I say that in the time z increases to $z + v$, x increases likewise to become $x + \underset{\cdot}{x}\frac{v}{1\underset{\cdot}{z}} + \underset{\cdot\cdot}{x}\frac{v\overset{\backslash}{v}}{1\cdot 2\underset{\cdot}{z}^2} + \underset{\cdots}{x}\frac{v\overset{\backslash}{v}\overset{\backslash\backslash}{v}}{1\cdot 2\cdot 3\underset{\cdot}{z}^3} + \&c.$ Corollary II: If for the evanescent increments, the fluxions of the proportionals themselves are written, now with all the $\overset{\backslash\backslash}{v}$, $\overset{\backslash}{v}$, v, $\underset{/}{v}$, $\underset{//}{v}$, &c. equal to the time z uniformly flows to become $z + v$, x becomes $x + \dot{x}\frac{v}{1\dot{z}} + \ddot{x}\frac{v^2}{1\cdot 2\dot{z}^2} + \dddot{x}\frac{v^3}{1\cdot 2\cdot 3\dot{z}^3} + \&c$ …

Crumpet 5: Interpretation of the original theorem of Brook Taylor

Unfortunately, the English translation of Taylor's theorem is only moderately helpful to anyone who is not well acquainted with early 18th century mathematics. In 1715, function notation was still 20 years in the making. Today, we would interpret the declaration of the two variables as declaring that x is a function of z. The claim in Theorem III is that we can rewrite $x(z+v)$ as $x + \underset{\cdot}{x}\frac{v}{1\underset{\cdot}{z}} + \underset{\cdot\cdot}{x}\frac{v\overset{\backslash}{v}}{1\cdot 2\underset{\cdot}{z}^2} + \underset{\cdots}{x}\frac{v\overset{\backslash}{v}\overset{\backslash\backslash}{v}}{1\cdot 2\cdot 3\underset{\cdot}{z}^3}$ + &c. Just as x should be interpreted as a function of z so should $\underset{\cdot}{x}$, $\underset{\cdot\cdot}{x}$, and $\underset{\cdots}{x}$. More precisely, $\underset{\cdot}{x}$ means $x(z+\underset{\cdot}{z}) - x(z)$, the amount x is incremented as z is incremented by $\underset{\cdot}{z}$. Likewise, $\underset{\cdot\cdot}{x}$ is the amount $\underset{\cdot}{x}$ is incremented as z is incremented by $\underset{\cdot}{z}$, so $\underset{\cdot\cdot}{x} = \underset{\cdot}{x}(z+\underset{\cdot}{z}) - \underset{\cdot}{x}(z) = \Big[x(z+2\underset{\cdot}{z}) - x(z+\underset{\cdot}{z})\Big] - \Big[x(z+\underset{\cdot}{z}) - x(z)\Big] = x(z+2\underset{\cdot}{z}) - 2x(z+\underset{\cdot}{z}) + x(z)$. Similarly, $\underset{\cdots}{x}$ is the amount $\underset{\cdot\cdot}{x}$ is incremented as z is incremented by $\underset{\cdot}{z}$. Now would be a good time to break from reading to verify that $\underset{\cdots}{x} = x(z+3\underset{\cdot}{z}) - 3x(z+2\underset{\cdot}{z}) + 3x(z+\underset{\cdot}{z}) - x(z)$, that $\underset{\cdot\cdot\cdot\cdot}{x} = x(z+4\underset{\cdot}{z}) - 4x(z+3\underset{\cdot}{z}) + 6x(z+2\underset{\cdot}{z}) - 4x(z+\underset{\cdot}{z}) + x(z)$, and so on. With this understanding and the conventions $\underset{0}{x}$ for x, $\underset{1}{x}$ for $\underset{\cdot}{x}$, $\underset{2}{x}$ for $\underset{\cdot\cdot}{x}$, $\overset{0}{v}$ for v, $\overset{1}{v}$ for $\overset{\backslash}{v}$, $\overset{2}{v}$ for $\overset{\backslash\backslash}{v}$, and so on, it is then an algebraic exercise to see that

$$\begin{aligned}
x(z+n\underset{\cdot}{z}) &= \sum_{j=0}^{n}\left(\begin{array}{c} n \\ j \end{array}\right)\underset{j}{x} = \underset{0}{x} + \underset{1}{x}\frac{n}{1} + \underset{2}{x}\frac{n(n-1)}{1\cdot 2} + \underset{3}{x}\frac{n(n-1)(n-2)}{1\cdot 2\cdot 3} + \cdots + \underset{n}{x}\frac{n(n-1)\cdots 1}{1\cdot 2\cdot 3\cdots n} \\
&= \underset{0}{x} + \underset{1}{x}\frac{n\underset{\cdot}{z}}{1\underset{\cdot}{z}} + \underset{2}{x}\frac{n\underset{\cdot}{z}(n-1)\underset{\cdot}{z}}{1\cdot 2\underset{\cdot}{z}^2} + \underset{3}{x}\frac{n\underset{\cdot}{z}(n-1)\underset{\cdot}{z}(n-2)\underset{\cdot}{z}}{1\cdot 2\cdot 3\underset{\cdot}{z}^3} + \cdots + \underset{n}{x}\frac{n\underset{\cdot}{z}(n-1)\underset{\cdot}{z}\cdots 1\underset{\cdot}{z}}{1\cdot 2\cdot 3\cdots n\underset{\cdot}{z}^n} \\
&= \underset{0}{x} + \underset{1}{x}\frac{v}{1\underset{\cdot}{z}} + \underset{2}{x}\frac{v\overset{1}{v}}{1\cdot 2\underset{\cdot}{z}^2} + \underset{3}{x}\frac{v\overset{2}{v}\overset{3}{v}}{1\cdot 2\cdot 3\underset{\cdot}{z}^3} + \cdots + \underset{n}{x}\frac{v\overset{2}{v}\overset{3}{v}\cdots\overset{n}{v}}{1\cdot 2\cdot 3\cdots n\underset{\cdot}{z}^n}.
\end{aligned}$$

This calculation is essentially Taylor's proof of Theorem III.

Corollary II (which we would consider the theorem) is not proved by Taylor beyond the "obvious" application of Newton's theory of fluxions. In today's language, corollary II follows by applying the limit as $n \to \infty$ to the expression from Theorem III. It makes for another nice exercise to verify that $\lim_{n\to\infty}\frac{\underset{k}{x}}{\underset{\cdot}{z}^k} = x^{(k)}(z)$, the k^{th} derivative of x. And one final exercise to see that $\lim_{n\to\infty}\overset{k}{v} = v$. As Taylor took these results for granted, so shall we. Applying them to Theorem III, we see that $x(z+v) = x(z) + x'(z)\frac{v}{1!} + x''(z)\frac{v^2}{2!} + x'''(z)\frac{v^3}{3!} + \cdots$. In the notation of Taylor, $\frac{\dot{x}}{\dot{z}}$ is the first derivative of x, $\frac{\ddot{x}}{\dot{z}^2}$ is the second derivative of x, and so on. So we in fact have $x + \dot{x}\frac{v}{1\dot{z}} + \ddot{x}\frac{v^2}{1\cdot 2\dot{z}^2} + \dddot{x}\frac{v^3}{1\cdot 2\cdot 3\dot{z}^3}$ + &c as claimed.

It is interesting that Theorem III is true for any function x defined on the interval $[x, x+v]$. No matter if x is differentiable, or even continuous. It is a statement about finite differences. It is the corollary that requires many more assumptions because that is where we pass to the limit.

Exercises

1. Find $T_3(x)$ and $R_3(x)$ for the function expanded about x_0.
   (a) $f(x) = \sin(x)$; $x_0 = 0$. [S]
   (b) $f(x) = \sin(x)$; $x_0 = \pi/2$.
   (c) $f(x) = \sin(x)$; $x_0 = \pi$. [S]
   (d) $f(x) = e^x$; $x_0 = 0$.
   (e) $f(x) = e^x$; $x_0 = \ln 2$.
   (f) $f(x) = x\sin(x)$; $x_0 = 0$. [A]
   (g) $f(x) = \cos^2(x)$; $x_0 = 0$.
2. Let $f(x) = 4x^3 - 2x^2 + 8x - 9$.
   (a) Find $T_3(x)$ and $R_3(x)$ expanded about $x_0 = 0$.
   (b) Find $T_3(x)$ and $R_3(x)$ expanded about $x_0 = 2$.
   (c) Make a conjecture based on your answers to parts (a) and (b). Can you prove it?
3. Find the 36th Maclaurin Polynomial for $f(x) = e^x$.
4. Suppose $f(x)$ is a function whose fourth derivative exists on the whole real line, $(-\infty, \infty)$, and that $f(2) = 3$, $f'(2) = -1$, $f''(2) = 2$, and $f'''(2) = -1$.
   (a) Write down the third Taylor polynomial for $f(x)$ expanded about $x_0 = 2$.
   (b) Use the Taylor polynomial to approximate $f(4)$.

(c) Find a bound on the absolute error of the approximation using the fact that

$$-3 \le f^{(4)}(\xi) \le 5$$

for all $\xi \in [2, 4]$.

5. Compute the 3^{rd} Taylor Polynomial for $f(x) = x^5 - 2x^4 + x^3 - 9x^2 + x - 1$ expanded about $x_0 = 1$.

6. Find the second Taylor Polynomial for $f(x) = \csc x$ expanded about $x_0 = \frac{\pi}{4}$. Here are some facts you may find useful:

$$f'(x) = -\csc(x)\cot(x) \qquad \csc(x) = \frac{1}{\sin(x)}$$

$$f''(x) = \csc(x)(1 + 2\cot^2(x)) \qquad \cot(x) = \frac{\cos(x)}{\sin(x)}$$

7. The hyperbolic sine, $\sinh(x)$, and hyperbolic cosine, $\cosh(x)$, are derivatives of one another. That is,

$$\frac{\mathrm{d}}{\mathrm{d}x}(\sinh(x)) = \cosh(x)$$

and

$$\frac{\mathrm{d}}{\mathrm{d}x}(\cosh(x)) = \sinh(x).$$

Find the remainder term, R_{43}, associated with the 43^{rd} Maclaurin polynomial for $f(x) = \cosh(x)$.

8. Use an `inline` function to evaluate the Taylor polynomial $T_4(x) = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4$ at the given value of x. [S]

   (a) 0

   (b) $\frac{1}{2}$

   (c) 1

   (d) π

9. Use an `inline` function to evaluate the Taylor polynomial $T_3(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3$ at the given value of x.

   (a) 0

   (b) $\frac{3}{2}$

   (c) 2

   (d) e [A]

10. Write and run a `.m` file that finds all the answers for exercise 8. [S]

11. Write and run a `.m` file that finds all the answers for exercise 9.

12. Find $\xi(x)$ as guaranteed by Taylor's theorem in the following situation.

   (a) $f(x) = \cos(x)$, $x_0 = 0$, $n = 3$, $x = \pi$. [A]

   (b) $f(x) = e^x$, $x_0 = 0$, $n = 3$, $x = \ln 4$.

   (c) $f(x) = \ln(x)$, $x_0 = 1$, $n = 4$, $x = 2$.

13. Let $f(x) = x^3$.

   (a) Find the second Taylor polynomial, $P_2(x)$, about $x_0 = 0$.

   (b) Find the remainder term, $R_2(0.5)$, and the actual error in using $P_2(0.5)$ to approximate $f(0.5)$.

   (c) Repeat part (a) using $x_0 = 1$.

   (d) Repeat part (b) using the polynomial from part (c).

14. Find the second Taylor polynomial, $P_2(x)$, for $f(x) = e^x \cos x$ about $x_0 = 0$.

   (a) Use $P_2(0.5)$ to approximate $f(0.5)$. Find an upper bound on the error $|f(0.5) - P_2(0.5)|$ using the remainder term and compare it to the actual error.

   (b) Find a bound on the error $|f(x) - P_2(x)|$ good on the interval $[0, 1]$.

   (c) Approximate $\int_0^1 f(x)\,dx$ by calculating $\int_0^1 P_2(x)\,dx$ instead.

   (d) Find an upper bound for the error in (c) using $\int_0^1 |R_2(x)|\,dx$ and compare the bound to the actual error.

15. Let $f(x) = e^x$.

   (a) Find the n^{th} Maclaurin polynomial $P_n(x)$ for $f(x)$.

   (b) Find a bound on the error in using $P_4(2)$ to approximate $f(2)$.

   (c) How many terms of the Maclaurin polynomial would you need to use in order to approximate $f(2)$ to within 10^{-10}? In other words, for what n does $P_n(2)$ have an error bound less than or equal to 10^{-10}?

16. Find the fourth Taylor Polynomial for $\ln x$ expanded about $x_0 = 1$.

17. What is the 50^{th} term of $T_{100}(e^x)$ expanded about $x_0 = 6$?

18. The Maclaurin series for the arctangent function converges for $-1 < x \le 1$ and is given by

$$\arctan x = \lim_{n\to\infty} P_n(x) = \lim_{n\to\infty} \sum_{i=n+1}^{\infty} (-1)^{i+1} \frac{x^{2i-1}}{2i-1}.$$

Use the fact that $\tan(\pi/4) = 1$ to determine the number of terms, n, of the series that need to be summed to ensure that $|4P_n(1) - \pi| < 10^{-3}$.

19. Exercise 18 details a rather inefficient means of obtaining an approximation to π. The method can be improved substantially by observing that $\pi/4 = \arctan \frac{1}{2} + \arctan \frac{1}{3}$ and evaluating the series for the arctangent at $\frac{1}{2}$ and at $\frac{1}{3}$. Determine the number of terms that must be summed to ensure an approximation to π within 10^{-3}.

20. For $f(x) = \tan^{-1}(x)$,

$$f^{(n)}(0) = \begin{cases} 0 & \text{if } n \text{ is even} \\ (-1)^{(n-1)/2}(n-1)! & \text{if } n \text{ is odd.} \end{cases}$$

Find the n^{th} Maclaurin polynomial $P_n(x)$ for f.

21. How many terms of the Maclaurin Series of $\sin x$ are needed to guarantee an approximation with error no more than 10^{-2} for any value of x between 0 and 2π?

22. Suppose you are approximating $f(x) = e^x$ using the tenth Maclaurin polynomial. Find the largest interval over which the approximation is guaranteed to be accurate to within 10^{-3}.

23. Find a bound on the error in approximating e^{10} by using the twenty-fifth Taylor polynomial of $g(x) = e^x$ expanded about $x_0 = 0$.

24. Find a bound on the error of the approximation

$$e^2 \approx 1 + 2 + \frac{1}{2}(2)^2 + \frac{1}{6}(2)^3 + \frac{1}{24}(2)^4 + \frac{1}{120}(2)^5$$

according to Taylor's Theorem. Compare this bound to the actual error.

25. Suppose $f^{(8)}(x) = e^x \cos x$ for some function f. Find a bound on the error in approximating $f(x)$ over the interval $[0, \pi/2]$ using $T_7(x)$ expanded about $x_0 = 0$.

26. Let $f(x) = \frac{1}{x}$, and $x_0 = 5$. [S]

    (a) Find $T_2(x)$.

    (b) Find $R_2(x)$.

    (c) Use $T_2(x)$ to approximate $f(1)$ and $f(9)$.

    (d) Find a theoretical upper bound on the absolute error of each of the approximations in part (c).

    (e) Find a theoretical lower bound on the absolute error of each of the approximations in part (c).

    (f) Find the actual absolute error for each of the approximations in part (c). Verify that they are indeed between the theoretical bounds.

    (g) Sketch graphs of $f(x)$ and $T_2(x)$ on the same set of axes for $x \in [1, 9]$.

27. Let $f(x) = \ln(1 + x)$ and $x_0 = 0$.

    (a) Find $T_3(x)$.

    (b) Find $R_3(x)$.

    (c) Use $T_3(x)$ to approximate $f(1)$ and $f(26)$.

    (d) Find a theoretical upper bound on the absolute error of each of the approximations in part (c).

    (e) Find a theoretical lower bound on the absolute error of each of the approximations in part (c).

    (f) Find the actual absolute error for each of the approximations in part (c). Verify that they are indeed between the theoretical bounds.

    (g) Sketch graphs of $f(x)$ and $T_2(x)$ on the same set of axes for $x \in [1, 26]$.

28. Suppose $f(x)$ is such that $-3 \leq f^{(10)}(x) \leq 7$ for all $x \in [0, 10]$. Find lower and upper bounds on the absolute error in using $T_9(x)$ expanded about $x_0 = 3$ to approximate

    (a) $f(0)$.

    (b) $f(10)$.

29. Suppose you wish to approximate the value of $-e^4 \sin 4$ using separate Maclaurin polynomials (Taylor polynomials expanded about $x_0 = 0$) for the sine and exponential functions instead of a single Maclaurin polynomial for the function $f(x) = -e^x \sin x$. How many terms of each would you need in order to get accuracy within 10^{-20}? Ignore round-off error.

30. Find a theoretical upper bound, as a function of x, for the absolute error in using $T_4(x)$ to approximate $f(x)$.

    (a) $e^x \sin x$; $x_0 = 0$.

    (b) e^{-x^2}; $x_0 = 0$. [S]

    (c) $\frac{10}{x} + \sin(10x)$; $x_0 = \pi$.

31. The Maclaurin Series for $f(x) = e^{-x}$ is

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{i!} x^i = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \ldots$$

Find a bound on the error in approximating $1/e$ by $1 - 1 + 1/2 - 1/6 + 1/24$.

32. The Taylor series for $f(x) = e^x$ is $T(x) = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 + \cdots$. This series converges to $f(x)$ for all values of x. In particular, for $x = 1$, this means that

$$f(1) = T(1) = 1 + 1 + \frac{1}{2!}(1)^2 + \frac{1}{3!}(1)^3 + \frac{1}{4!}(1)^4 + \cdots$$

Simplifying this equation, we see that

$$e = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \cdots$$

Use Taylor Series to find infinite sums that sum to

    (a) $\ln(2)$

    (b) $2/3$

    (c) $\pi/4$

    (d) $\sqrt{2}$

1.3 Speed

Besides accuracy, there is nothing more important about a numerical method than speed. There is almost always a trade-off between one and the other, however. Fast computations are often not particularly accurate, and accurate calculations are often not particularly fast. There are certain algorithms that produce accurate results quickly, however. Deriving them, or identifying them once derived is what numerical analysis is all about.

The first type of numerical method we will encounter produces a sequence of approximations that, when everything is working, approach some desired value, say p. With these methods, we will get a sequence $\langle p_n \rangle$ with $\lim_{n\to\infty} p_n = p$. You should be familiar with the concept of the limit of a sequence from Calculus, but the purpose there was much different from ours here. Generally, you were concerned with whether a given sequence converged at all. And when it did converge, and you were very lucky, you were able to determine the limit. In numerical analysis, we know certain sequences converge, and are only interested in how quickly they do so.

Simple observation (and a little common sense) can tell you which cars on a highway are traveling faster than which. Simple observation (and a little common sense) will also often tell you which sequences converge faster than which. Consider the sequences in Table 1.1 which all converge to $e \approx 2.71828182845904$. $\langle t_n \rangle$ is accurate to 15 significant digits by the sixth term; $\langle s_n \rangle$ is accurate to 15 significant digits by the eighth term; $\langle r_n \rangle$ is still not accurate to 15 significant digits by the eleventh term, but seems likely to gain 15 significant digits of accuracy on the twelfth term; and $\langle q_n \rangle$ is only accurate to 2 significant digits by the eleventh term, so seems likely to take considerably more than twelve terms to gain 15 significant digits of accuracy. Since they all started at 3, it seems reasonable to say that, ordered from fastest to slowest, they are $\langle t_n \rangle$, $\langle s_n \rangle$, $\langle r_n \rangle$, $\langle q_n \rangle$. And that is correct as we will see soon. But just like knowing which cars are faster than which is different from knowing how fast each is going, knowing which sequences converge faster than which is different from knowing how quickly each one converges. To measure the speed of a given car, you need access to its speedometer or a radar gun. To measure the order of convergence (speed) of a sequence, you need a definition and a little algebra.

Table 1.1: Some sequences that converge to e.

n	q_n	r_n	s_n	t_n
0	3	3	3	3
1	2.9436563656918	2.86799618929986	2.82129001274358	2.78177393100014
2	2.89858145824525	2.78315514435127	2.73850656616954	2.72150682612711
3	2.86252153228801	2.73974041668143	2.71973377603211	2.71829014894701
4	2.83367359152222	2.72324781752852	2.71830229432561	2.71828182851442
5	2.81059523890958	2.71899828870116	2.71828184916891	2.71828182845904
6	2.79213255681947	2.71833715075158	2.71828182845934	2.71828182845904
7	2.77736241114739	2.71828369688657	2.71828182845904	2.71828182845904
8	2.76554629460972	2.71828184959225	2.71828182845904	2.71828182845904
9	2.75609340137958	2.71828182851528	2.71828182845904	2.71828182845904
10	2.74853108679547	2.71828182845907	2.71828182845904	2.71828182845904
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Order of convergence of a sequence

Suppose the sequence $\langle p_n \rangle$ converges to p. Then we say $\langle p_n \rangle$ converges with order $\alpha \geq 1$ if

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} = \lambda$$

for some real number $\lambda > 0$.

Let's see how to use this definition to calculate the orders of convergence of the sequences in Table 1.1. According to the definition, α, should it exist, gives the speed (or order) of convergence of a sequence. Now assuming that α does exist, we have that $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^{\alpha}} = \lambda$, so for large enough n,

$$\frac{|p_{n+1} - p|}{|p_n - p|^{\alpha}} \approx \frac{|p_{n+2} - p|}{|p_{n+1} - p|^{\alpha}} \approx \lambda.$$

In particular, we can solve for α to find $\alpha \approx \dfrac{\ln\left|\frac{p_{n+2}-p}{p_{n+1}-p}\right|}{\ln\left|\frac{p_{n+1}-p}{p_n-p}\right|}$.

Crumpet 6: Order of Convergence Less than or equal to 1?

There is no such thing as an order of convergence less than one because if $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^\alpha} = \lambda$ for some $0 < \alpha < 1$, then

$$\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|} = \lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^\alpha} \cdot |p_n-p|^{\alpha-1},$$

a contradiction. On the one hand, the ratio test implies that $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|}$ exists and is less than or equal to 1. On the other hand, $\alpha < 1 \implies \alpha - 1 < 0$ so for $|p_n - p|$ small, $|p_n - p|^{\alpha-1}$ is large. Hence, $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^\alpha} \cdot |p_n - p|^{\alpha-1}$ does not exist. To be rigorous, let M be any real number. Then there exists an N_1 such that $n > N_1$ implies $\frac{|p_{n+1}-p|}{|p_n-p|^\alpha} > 0.9\lambda$. There also exists N_2 such that $n > N_2$ implies $|p_n - p| < \left(\frac{0.9\lambda}{M}\right)^{\frac{1}{1-\alpha}}$, so $|p_n - p|^{\alpha-1} > \frac{M}{0.9\lambda}$. Letting $N = \max\{N_1, N_2\}$ we have that $n > N$ implies both $\frac{|p_{n+1}-p|}{|p_n-p|^\alpha} > 0.9\lambda$ and $|p_n - p|^{\alpha-1} > \frac{M}{0.9\lambda}$. Hence, for $n > N$, we have

$$\frac{|p_{n+1}-p|}{|p_n-p|} = \frac{|p_{n+1}-p|}{|p_n-p|^\alpha} \cdot |p_n-p|^{\alpha-1} > 0.9\lambda \cdot \frac{M}{0.9\lambda} = M.$$

Therefore, $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|}$ does not exist. When $\alpha = 1$, it must be that $\lambda \le 1$ because otherwise the ratio test implies that $\langle |p_n - p| \rangle$ diverges, and, therefore, $\langle p_n \rangle$ diverges.

For example, $\dfrac{\ln\left|\frac{q_2-e}{q_1-e}\right|}{\ln\left|\frac{q_1-e}{q_0-e}\right|} \approx \dfrac{\ln\left|\frac{2.8985-e}{2.9436-e}\right|}{\ln\left|\frac{2.9436-e}{3-e}\right|} \approx 1$ and $\dfrac{\ln\left|\frac{q_{10}-e}{q_9-e}\right|}{\ln\left|\frac{q_9-e}{q_8-e}\right|} = \dfrac{\ln\left|\frac{2.7485-e}{2.7560-e}\right|}{\ln\left|\frac{2.7560-e}{2.7655-e}\right|} \approx 1$. And if we try other sets of three consecutive terms of $\langle q_n \rangle$, we get the same results. The order of convergence of $\langle q_n \rangle$ is about 1. Of course, we would need a formula for $|q_n - e|$ to determine whether the limit were truly 1, but we have some evidence. Repeating the calculations for $\langle r_n \rangle$, $\langle s_n \rangle$, and $\langle t_n \rangle$, we get approximate orders of convergence 1.322, 1.618, and 2, respectively. Again we see that, ordered from fastest to slowest, they are $\langle t_n \rangle$, $\langle s_n \rangle$, $\langle r_n \rangle$, $\langle q_n \rangle$.

If you attempted to calculate the orders of convergence yourself, you may have noticed that more information is needed to use s_n with $n > 6$ or t_n with $n > 4$. All of these terms in the table are equal, so the formula for α fails to produce a real number! A more useful table for calculating orders of convergence is one listing absolute errors: In addition to making it easier to calculate α, this chart makes it painfully obvious that our common sense conclusion

Table 1.2: Absolute errors.

n	$\lvert q_n - e\rvert$	$\lvert r_n - e\rvert$	$\lvert s_n - e\rvert$	$\lvert t_n - e\rvert$
0	$2.817(10)^{-1}$	$2.817(10)^{-1}$	$2.817(10)^{-1}$	$2.817(10)^{-1}$
1	$2.253(10)^{-1}$	$1.497(10)^{-1}$	$1.03(10)^{-1}$	$6.349(10)^{-2}$
2	$1.802(10)^{-1}$	$6.487(10)^{-2}$	$2.022(10)^{-2}$	$3.224(10)^{-3}$
3	$1.442(10)^{-1}$	$2.145(10)^{-2}$	$1.451(10)^{-3}$	$8.32(10)^{-6}$
4	$1.153(10)^{-1}$	$4.965(10)^{-3}$	$2.046(10)^{-5}$	$5.538(10)^{-11}$
5	$9.231(10)^{-2}$	$7.164(10)^{-4}$	$2.07(10)^{-8}$	$2.453(10)^{-21}$
6	$7.385(10)^{-2}$	$5.532(10)^{-5}$	$2.953(10)^{-13}$	$4.817(10)^{-42}$
7	$5.908(10)^{-2}$	$1.868(10)^{-6}$	$4.263(10)^{-21}$	$1.856(10)^{-83}$
8	$4.726(10)^{-2}$	$2.113(10)^{-8}$	$8.777(10)^{-34}$	$2.757(10)^{-166}$
9	$3.781(10)^{-2}$	$5.623(10)^{-11}$	$2.608(10)^{-54}$	$6.084(10)^{-332}$
10	$3.024(10)^{-2}$	$2.22(10)^{-14}$	$1.595(10)^{-87}$	$2.961(10)^{-663}$

about which sequences converge faster than which was quite right. Just compare the accuracy (absolute errors) of the eleventh terms.

So now we can calculate orders of convergence, but what does it all mean? What does the order of convergence tell us about successive terms in the sequence? Solving the approximation $\frac{|p_{n+1}-p|}{|p_n-p|^\alpha} \approx \lambda$ gives us that $|p_{n+1}-p| \approx \lambda|p_n-p|^\alpha$. So, roughly speaking, convergence of order α means that, for large enough n, the approximation p_{n+1} is about $\lambda|p_n-p|^{\alpha-1}$ times closer to the limit p than is p_n. To rephrase in terms of significant digits of accuracy, a little bit of algebra:

$$\begin{aligned}
|p_{n+1}-p| &\approx \lambda|p_n-p|^\alpha \\
\left|\frac{p_{n+1}-p}{p}\right| &\approx \lambda\left|\frac{p_n-p}{p}\right|^\alpha \cdot |p|^{\alpha-1} \\
-\log\left|\frac{p_{n+1}-p}{p}\right| &\approx -\log\left|\frac{p_n-p}{p}\right|^\alpha - \log\left(\lambda|p|^{\alpha-1}\right) \\
d(p_{n+1}) &\approx \alpha d(p_n) - \log\left(\lambda|p|^{\alpha-1}\right).
\end{aligned}$$

Based on this calculation, we conclude these rules of thumb:

1. for linear convergence ($\alpha = 1$), $d(p_{n+1}) \approx d(p_n) - \log\lambda$, so each term has a fixed number more significant digits of accuracy (approximately equal to $-\log\lambda$) than the previous;

2. for quadratic convergence ($\alpha = 2$), $d(p_{n+1}) \approx 2d(p_n) - \log(\lambda|p|)$, so each term has double the number of significant digits of accuracy of the previous, give or take some;

3. for cubic convergence ($\alpha = 3$), $d(p_{n+1}) \approx 3d(p_n) - \log\left(\lambda|p|^2\right)$, so each term has triple the number of significant digits of accuracy of the previous, give or take some;

and so on. Summarizing, for large n, you can expect that each term will have $-\log\left(\lambda|p|^{\alpha-1}\right)$ more than α times as many significant digits of accuracy as the previous term. We can see this claim in action by calculating λ for the sequences $\langle t_n\rangle$, $\langle s_n\rangle$, $\langle r_n\rangle$, and $\langle q_n\rangle$. Using the fact that $\lambda \approx \frac{|p_{n+1}-p|}{|p_n-p|^\alpha}$, we find that $\lambda = 0.8$ for each sequence. Therefore, $\langle q_n\rangle$ should show each term having $-\log 0.8 \approx .1$ more significant digits of accuracy than the previous. More sensibly, this means the sequence will show about one more significant digit of accuracy every ten terms. This is borne out by observing that q_0 has error about $3(10)^{-1}$ while q_{10} has error about $3(10)^{-2}$. For $\langle r_n\rangle$, we should expect each term to have about $-\log(0.8 \cdot e^{.322}) \approx -0.04$ more than 1.322 times as many significant digits of accuracy as the previous. For example, r_3 has about $\log\left|\frac{e}{2.145(10)^{-2}}\right| \approx 2.1$ significant digits of accuracy while r_4 has about $1.322(2.1) - .04 \approx 2.73$ significant digits of accuracy, r_5 has $1.322(2.73) - .04 \approx 3.57$ significant digits of accuracy, and so on until r_8 has about 8.1 significant digits of accuracy. Again this is borne out by the table as $\log\left|\frac{e}{r_8-e}\right| = \log\left|\frac{e}{2.113(10)^{-8}}\right| \approx 8.1$. Though we can do a similar calculation for $\langle t_n\rangle$, it's easier just to eyeball it since all we need to see is that the exponent in the scientific notation doubles, give or take a little, from one term to the next. Indeed it does as it goes from 1 to 2 to 3 to 6 to 11, and so on.

Note that in all this analysis, we have ignored the requirement that n be "large". That was acceptable in this case since these sequences were contrived so that even $n = 0$ was large enough! In practical applications this will not be the case.

To appreciate just how much faster one order of convergence is over another, consider the relation

$$d(p_{n+1}) \approx \alpha d(p_n) - \log\left(\lambda|p|^{\alpha-1}\right)$$

again. Now suppose we know that $d(p_{n_0}) = d_{n_0}$ for some particular n_0 large enough that the approximation is reasonable. Then it can be shown that, for $\alpha > 1$,

$$d(p_{n_0+k}) \approx (d_{n_0} - C)\alpha^k + C$$

where $C = \dfrac{\log\left(\lambda|p|^{\alpha-1}\right)}{\alpha-1}$.

Crumpet 7: Solving a Recurrence Relation

The relation $d(p_{n+1}) \approx \alpha d(p_n) - \log\left(\lambda|p|^{\alpha-1}\right)$ is an example of a recurrence relation. In particular, a first order linear nonhomogeneous recurrence relation with constant coefficients since it has the form

$$a_{n+1} = k_1 a_n + k_2$$

where k_1 and k_2 are constants. Linear nonhomogeneous recurrence relations can be solved by summing a homogeneous solution and a particular solution. For the particular solution, we seek a solution of the form $a_n = A$ (for all n) by substituting this assumed solution into the recurrence relation. Doing so gives $A = k_1 A + k_2$, so $A = \frac{k_2}{1-k_1}$ is such a solution. For the homogeneous solution, we seek a sequence of the form $a_n = r^n$ that satisfies $a_{n+1} = k_1 a_n + 0$. Substituting our assumed solution into the modified (homogeneous) recurrence relation gives $r^{n+1} = k_1 r^n$. Rearranging, $r^n(r - k_1) = 0$ so $r = 0$ or $r = k_1$. Notice that Bk_1^n is also a solution for any constant B. This includes the solution $a_n = 0$ which would arise from setting $r = 0$. Finally, putting the particular and homogeneous solutions together, the solution of $a_{n+1} = k_1 a_n + k_2$ is $a_n = Bk_1^n + \frac{k_2}{1-k_1}$ for any constant B. In the case of $d(p_{n+1}) \approx \alpha d(p_n) - \log\left(\lambda|p|^{\alpha-1}\right)$, $k_1 = \alpha$ and $k_2 = -\log\left(\lambda|p|^{\alpha-1}\right)$ so $d(p_n) = B\alpha^n + \frac{\log\left(\lambda|p|^{\alpha-1}\right)}{\alpha-1}$. The value of B is determined by substituting any known element of the sequence into this formula and solving for B. Supposing $d(p_{n_0}) = d_{n_0}$ yields $d(p_n) = \left(d_{n_0} - \frac{\log\left(\lambda|p|^{\alpha-1}\right)}{\alpha-1}\right)\alpha^n + \frac{\log\left(\lambda|p|^{\alpha-1}\right)}{\alpha-1}$.

The important thing to see here is that $d(p_{n_0+k})$ is an exponential function when $\alpha > 1$. The number of significant digits of accuracy grows exponentially with base α. As we saw before, for $\alpha = 1$, the number of significant digits grows linearly. In calculus you learned that any exponential function grows much faster than any polynomial function, so it is reasonable and correct to conclude that sequences converging with orders greater than 1 are markedly faster converging than are sequences converging with linear ($\alpha = 1$) order.

But be careful. Based on this same memory of calculus, you would also conclude that the sequence $\langle 2^{-n}\rangle$ converges to 0 much faster than does $\langle n^{-2}\rangle$. By some measures, that's true, but not by all measures. Consider the orders of convergence of these two sequences. We seek values α_1 and α_2 such that

$$\lim_{n\to\infty} \frac{|2^{-(n+1)} - 0|}{|2^{-n} - 0|^{\alpha_1}} = \lambda_1 \qquad \text{and} \qquad \lim_{n\to\infty} \frac{|(n+1)^{-2} - 0|}{|n^{-2} - 0|^{\alpha_2}} = \lambda_2$$

for some real numbers λ_1 and λ_2. A little bit of algebra will lead to solutions:

$$\frac{|2^{-(n+1)} - 0|}{|2^{-n} - 0|^{\alpha_1}} = \frac{2^{-n-1}}{2^{-\alpha_1 n}} = 2^{(\alpha_1-1)n-1}$$

while

$$\frac{|(n+1)^{-2} - 0|}{|n^{-2} - 0|^{\alpha_2}} = \frac{n^{2\alpha_2}}{n^2 + 2n + 1}.$$

The only way $\lim_{n\to\infty} 2^{(\alpha_1-1)n-1}$ will be a nonzero constant is if $\alpha_1 = 1$. The only way $\lim_{n\to\infty} \frac{n^{2\alpha_2}}{n^2+2n+1}$ will be a nonzero constant is if the leading coefficients of the numerator and denominator are equal. That means α_2 must be 1 as well. So $\langle 2^{-n}\rangle$ and $\langle n^{-2}\rangle$ both converge to zero with linear order. They are equally extremely slow to converge by this measure! Still, something should not feel quite right about claiming that $\langle 2^{-n}\rangle$ and $\langle n^{-2}\rangle$ converge at the same speed.

Rate of Convergence of a Sequence

For sequences that converge with linear order, we need a finer measure than order to determine which is faster than which. Recall from calculus,

$$\begin{aligned}
\lim_{n\to\infty} \frac{2^{-n}}{n^{-2}} &= \lim_{n\to\infty} \frac{n^2}{2^n} \\
&= \lim_{n\to\infty} \frac{2n}{2^n \ln 2} \\
&= \lim_{n\to\infty} \frac{2}{2^n (\ln 2)^2} = 0,
\end{aligned}$$

indicating that $\langle 2^{-n}\rangle$ approaches 0 much faster than does $\langle n^{-2}\rangle$. You may also recall comparisons between power functions:

$$\lim_{n\to\infty}\frac{n^{-p}}{n^{-q}}=0$$

whenever $p>q>0$; and between exponential functions:

$$\lim_{n\to\infty}\frac{a^{-n}}{b^{-n}}=0$$

whenever $a>b\ge 1$; and between the two:

$$\lim_{n\to\infty}\frac{a^{-n}}{n^{-q}}=0$$

whenever $a>1$. In other words, sequences of the form $\left\langle\frac{1}{a^n}\right\rangle$ converge to zero faster than sequences of the form $\left\langle\frac{1}{n^p}\right\rangle$ whenever $a>1$. The sequence $\left\langle\frac{1}{a^n}\right\rangle$ converges to zero faster than $\left\langle\frac{1}{b^n}\right\rangle$ whenever $a>b\ge 1$. The sequence $\left\langle\frac{1}{n^p}\right\rangle$ converges to zero faster than $\left\langle\frac{1}{n^q}\right\rangle$ whenever $p>q>0$. Not all functions are as simple as these, but we can use these as our yard sticks. Suppose $\langle p_n\rangle$ converges to p, $\langle b_n\rangle$ converges to 0 and $|p_n-p|\le\lambda|b_n|$ for some constant λ and all sufficiently large n. Then we say that $\langle p_n\rangle$ converges to p with rate of convergence $O(b_n)$, read "big-oh of b_n". Since we are familiar with sequences of the forms $\left\langle\frac{1}{a^n}\right\rangle$ for some constant $a>1$ and $\left\langle\frac{1}{n^p}\right\rangle$ for some constant $p>0$, and they are simple enough, typically $\langle b_n\rangle$ will be one of them. For example, $\left\langle\frac{2n+1}{4n}\right\rangle$ converges to $\frac{1}{2}$, and

$$\left|\frac{2n+1}{4n}-\frac{1}{2}\right|=\frac{1}{4n}\le\frac{1}{4}\cdot\frac{1}{n},$$

so $\left\langle\frac{2n+1}{4n}\right\rangle$ converges with rate $O(\frac{1}{n})$. We may also say that $\frac{2n+1}{4n}=\frac{1}{2}+O(\frac{1}{n})$ to convey exactly the same message. Normally, when we find a rate of convergence, we try to find the fastest converging sequence from our stock of simple examples that satisfies the definition. In this case, there is none faster.

Basically all the sequences studied in any depth in calculus converge with linear order. So what does it take to converge with a higher order? Let's have a look at $\left\langle e^{-2^n}\right\rangle$.

$$\lim_{n\to\infty}\frac{|e^{-2^{n+1}}-0|}{|e^{-2^n}-0|^\alpha}=\lim_{n\to\infty}\frac{e^{-2\cdot 2^n}}{e^{-\alpha 2^n}}=1$$

when $\alpha=2$. So $\left\langle\frac{1}{e^{2^n}}\right\rangle$ is quadratically convergent. Essentially, it takes an exponentially growing exponent to converge with an order greater than 1.

Crumpet 8: Approximating π

The sequence

$$\frac{1103\cdot 2^{3/2}}{9801},\ \frac{1130173253125}{313826716467\cdot 2^{7/2}},\ \frac{1029347477390786609545}{1116521080257783321\cdot 2^{23/2}},\ldots$$

converges to $\frac{1}{\pi}$. Its terms are given by the formula

$$\left\langle\frac{\sqrt{8}}{9801}\sum_{j=0}^{n}\frac{(4j)!(1103+26390j)}{(j!)^4\cdot 396^{4j}}\right\rangle_{n=0,1,2,3,\ldots}$$

of Srinivasa Ramanujan. For all practical purposes, it converges very quickly. The first term already has about 8 significant digits of accuracy:

$$\begin{aligned}\frac{1103\cdot 2^{3/2}}{9801}&\approx 0.31830987844047012321768445317\\ \frac{1}{\pi}&\approx 0.31830988618379067153776752674,\end{aligned}$$

and the second has about 16:

$$\left|\frac{1130173253125}{313826716467\cdot 2^{7/2}}-\frac{1}{\pi}\right|\approx 6.48(10)^{-17},$$

double the accuracy of the first term. The third term is already more than double-precision accurate.

It's tempting to believe, or hope, the sequence is quadratically convergent, but it is not. The third term has an accuracy of about 24 significant digits. Each term in the sequence is approximately 8 significant digits more accurate than the previous—the hallmark of a linearly convergent sequence.

Key Concepts

Order of convergence: The sequence $\langle p_n \rangle$ converges to p with order of convergence $\alpha \geq 1$ if

$$\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n - p|^\alpha} = \lambda$$

for some real number $\lambda > 0$.

Absolute error: For a sequence $\langle p_n \rangle$ that converges to p with order α, the absolute errors of consecutive terms are related by the approximation

$$|p_{n+1} - p| \approx \lambda |p_n - p|^\alpha$$

for large enough n.

Significant digits of accuracy: For a sequence $\langle p_n \rangle$ that converges to p with order α, the numbers of significant digits of accuracy of consecutive terms are related by the approximation

$$d(p_{n+1}) \approx \alpha d(p_n) - \log\left(\lambda |p|^{\alpha-1}\right)$$

for large enough n. In closed form (for $\alpha \neq 1$)

$$d(p_{n+k}) = (d_n - C)\alpha^k + C$$

where $C = \dfrac{\log\left(\lambda|p|^{\alpha-1}\right)}{\alpha - 1}$.

Rate of convergence: The sequence $\langle p_n \rangle$ converges to p with rate of convergence $O(b_n)$ if $\langle b_n \rangle$ converges to 0 and

$$|p_n - p| \leq \lambda |b_n|$$

for some constant λ and all sufficiently large n.

Exercises

1. Some convergent sequences and their limits are given. Find the order of convergence for each.

   (a) $\left\langle \dfrac{n!}{n^n} \right\rangle \to 0$

   (b) $\left\langle \dfrac{1}{3^{e^n}} \right\rangle \to 0$ [𝕊]

   (c) $\left\langle \dfrac{2^{2^n} - 2}{2^{2^n} + 3} \right\rangle \to 1$ [𝕊]

   (d) $\left\langle \dfrac{n^2}{1+n^2} \right\rangle \to 1$ [𝔸]

   (e) $\left\langle \dfrac{e^n}{e^{e^n}} \right\rangle \to 0$

2. Show that the sequence $\left\langle \dfrac{n+1}{n-1} \right\rangle$ converges to 1 linearly.

3. Show that the sequence $p_n = 2^{1-2^n}$ is quadratically convergent.

4. Give an example of a sequence which converges to 0 with order $\alpha = 10$.

5. Approximate the order of convergence of the sequence p_n and explain your answer.

n	$\frac{\lvert p_{n+1}-p\rvert}{\lvert p_n-p\rvert^{1.2}}$	$\frac{\lvert p_{n+1}-p\rvert}{\lvert p_n-p\rvert^{1.3}}$	$\frac{\lvert p_{n+1}-p\rvert}{\lvert p_n-p\rvert^{1.4}}$
25	$9.07(10)^{-6}$	.0110	13.39
26	$1.88(10)^{-7}$	.00303	48.65
27	$1.01(10)^{-9}$	.000530	277.8
28			

6. Some linearly convergent sequences and their limits are given. Find the (fastest) rate of convergence of the form $O\left(\frac{1}{n^p}\right)$ or $O\left(\frac{1}{a^n}\right)$ for each. If this is not possible, suggest a reasonable rate of convergence.

   (a) $6, \dfrac{6}{7}, \dfrac{6}{49}, \dfrac{6}{343}, \dfrac{6}{2401}, \ldots \to 0$

   (b) $\left\langle \dfrac{11n-2}{n+3} \right\rangle \to 11$

   (c) $\left\langle \dfrac{\sin n}{\sqrt{n}} \right\rangle \to 0$ [𝕊]

   (d) $\left\langle \dfrac{4}{10^n + 35n + 9} \right\rangle \to 0$ [𝕊]

   (e) $\left\langle \dfrac{4}{10^n - 35n - 9} \right\rangle \to 0$ [𝕊]

   (f) $\left\langle \dfrac{2n}{\sqrt{n^2+3n}} \right\rangle \to 2$ [𝔸]

   (g) $\left\langle \dfrac{5^n - 2}{5^n + 3} \right\rangle \to 1$

   (h) $\left\langle \sqrt{n+47} - \sqrt{n} \right\rangle \to 0$ [𝔸]

   (i) $\left\langle \dfrac{n^2}{3n^2+1} \right\rangle \to \dfrac{1}{3}$

(j) $\left\langle \dfrac{\pi}{e^n - \pi^n} \right\rangle \to 0$

(k) $\left\langle \dfrac{n^2}{2^n} \right\rangle \to 0$ [S]

(l) $\left\langle \dfrac{7+\cos(5n)}{n^3+1} \right\rangle \to 0$

(m) $\left\langle \dfrac{8n^2}{3n^2+12} + \dfrac{n}{3n+10} \right\rangle \to 3$

(n) $\left\langle \dfrac{2n^2+3n}{1-n^2} \right\rangle \to -2$ [A]

(o) $\left\langle \dfrac{3n^5-5n}{1-n^5} \right\rangle \to -3$

7. Find the rates of convergence of the following sequences as $n \to \infty$.

   (a) $\lim\limits_{n\to\infty} \sin \dfrac{1}{n} = 0$

   (b) $\lim\limits_{n\to\infty} \sin \dfrac{1}{n^2} = 0$

   (c) $\lim\limits_{n\to\infty} \left(\sin \dfrac{1}{n}\right)^2 = 0$

   (d) $\lim\limits_{n\to\infty} [\ln(n+1) - \ln(n)] = 0$

For questions on this page- on the current page, use the following definition for rate of convergence for a function. For a function $f(h)$, we say $\lim_{h\to a} f(h) = L$ with rate of convergence $g(h)$ if $|f(h)-L| \le \lambda |g(h)|$ for some $\lambda > 0$ and all sufficiently small $|h-a|$.

8. Use a Taylor polynomial to find the rate of convergence of
$$\lim_{h\to 0}(2 - e^h) = 1.$$

9. Use a Taylor polynomial to find the rate of convergence of
$$\lim_{h\to 0} \frac{\sin(h) - e^h + 1}{h} = 0.$$

10. Find rates of convergence for the following functions as $h \to 0$.

    (a) $\lim\limits_{h\to 0} \dfrac{\sin h}{h} = 1$

    (b) $\lim\limits_{h\to 0} \dfrac{1-\cos h}{h} = 0$

    (c) $\lim\limits_{h\to 0} \dfrac{\sin h - h\cos h}{h} = 0$

    (d) $\lim\limits_{h\to 0} \dfrac{1-e^h}{h} = -1$

11. Find the rate of convergence of
$$\lim_{h\to 0} \frac{h^2 + \cos h - e^h}{h} = -1.$$

12. Show that
$$(\sin h)(1-\cos h) = 0 + O(h^3).$$

13. C Write computer code (.m file) that uses a loop and the `disp()` command to produce the following output (powers of 7). [S]

```
1
7
49
343
2401
16807
117649
823543
5764801
40353607
```

14. C Write computer code (.m file) that uses a loop and the `disp()` command to output the first 10 powers of 5 starting with 5^0.

15. C Write computer code (.m file) that uses a loop, an array, and the disp() command to find the values of $f(n) = \dfrac{2^{2^n}-2}{2^{2^n}+3}$ for $n = 0, 1, 2, 4, 6, 10.$ [S]

16. C Write computer code (.m file) that uses a loop, an array, and the disp() command to find the values of $f(n) = \dfrac{2n}{\sqrt{n^2+3n}}$ for $n = 0, 2, 5, 10, 100, 1000, 20000.$

17. C The following code is intended to calculate the sum
$$\sum_{k=1}^{30} \frac{1}{k^2}$$
but it does not. Find as many mistakes in the code as you can. Classify each mistake as either a compilation error (an error that will prevent the program from running at all) or a bug (an error that will not prevent the program from running, but will cause improper calculation of the sum).

```
sum=1;
for k=1:30
  sum=sum+1.0/k*k;
end
disp(sum)
```

18. Some sequences do not have an order of convergence. Let $p_n = \frac{2^n}{n!}$.

    (a) Show that $\lim_{n\to\infty} p_n = 0$.

    (b) Show that $\lim_{n\to\infty} \frac{|p_{n+1}|}{|p_n|} = 0$.

    (c) Show that $\left\langle \frac{|p_{n+1}|}{|p_n|^\alpha} \right\rangle$ diverges for any $\alpha > 1$.

19. Use the rules of thumb for order of convergence to approximate the number of iterations it will take to achieve 12 significant digits of accuracy of π for each order of convergence. Assume each sequence starts with one significant digit of accuracy.

    (a) $\alpha = 1$, $\lambda = 0.8$

    (b) $\alpha = 1$, $\lambda = 0.5$ [S]

    (c) $\alpha = 1$, $\lambda = 0.1$

    (d) $\alpha = 1.5$

    (e) $\alpha = 2$ [A]

    (f) $\alpha = 3$

20. Prove that the order of convergence of a sequence is unique.

21. Write a `for` loop that outputs the sequence of numbers.

    (a) $7, 8, 9, 10, 11, 12, 13, 14, 15$

    (b) $20, 19, 18, 17, 16, 15, 14, 13$

    (c) $12, 12.333, 12.667, 13, 13.333, 13.667, 14$

    (d) $1, 9, 25, 49, 81, 121, 169, 225, 289, 361, 441$

    (e) $1, .5, .25, .125, .0625, .03125, .015625$

Chapter 2

Root Finding

2.1 Bisection

In Section 1.2 (page 10), we claimed that "$T_2(x)$ actually approximates $\ln(x)$ to within 0.1 over the interval $[3.296, 13.13]$", with a promise that we would discuss the calculation later. It is now later. First, we rephrase the claim as "the distance between $T_2(x)$ and $\ln(x)$ is less than or equal to 0.1 for all $x \in [3.296, 13.13]$." In other words,

$$|T_2(x) - \ln(x)| < \frac{1}{10} \quad \text{for all } x \in [3.296, 13.13].$$

One way to begin solving this inequality is to consider the pair of equations $T_2(x) - \ln(x) = \pm\frac{1}{10}$. With a focus on solving

$$T_2(x) - \ln(x) = \frac{1}{10}, \tag{2.1.1}$$

recall that $T_2(x) = 2 + \frac{x-e^2}{e^2} - \frac{(x-e^2)^2}{2e^4}$. We are thus looking to solve the equation

$$2 + \frac{x-e^2}{e^2} - \frac{(x-e^2)^2}{2e^4} - \ln(x) = \frac{1}{10}.$$

Finally, having written the equation in full detail, it should come as no surprise that we will not be solving for x exactly. There is no analytic method for solving such an equation. Generally, equations with both polynomial terms and transcendental terms will not be solvable. However, from the graph in Figure 1.2.2, we can get a first approximation of the solution. We are looking for the place where $T_2(x)$ exceeds $\ln(x)$ by 0.1. Since the two graphs essentially overlap at $x = 6$, we might aver that $T_2(6)$ exceeds $\ln(x)$ by *less than* 0.1 there. Since there is a reasonably large gap between the graphs at $x = 2$, we might also aver that $T_2(2)$ exceeds $\ln(x)$ by *more than* 0.1 there. In other words, $T_2(2) - \ln(2) > \frac{1}{10}$ while $T_2(6) - \ln(6) < \frac{1}{10}$. Since $T_2(x) - \ln(x)$ is continuous on the interval $[2, 6]$, the Intermediate Value theorem guarantees there is a value $c \in (2, 6)$ such that $T_2(c) - \ln(c) = \frac{1}{10}$. It is this value of c we are after. And we know it is between 2 and 6. It's a start, but we can do better!

What about 4? Well, $T_2(4) - \ln(4) \approx .04986 < 0.1$, so now we know $T_2(4)$ exceeds $\ln(4)$ by *less than* 0.1. Now the Intermediate Value theorem tells us that c is between 2 and 4 ($T_2(2)$ exceeds $\ln(x)$ by *more than* 0.1). Shall we check on $x = 3$? Yes. $T_2(3) - \ln(3) \approx .131 > 0.1$, so now we know $T_2(3)$ exceeds $\ln(3)$ by *more than* 0.1. Recapping, $T_2(4) - \ln(4) < 0.1$ while $T_2(3)\ln(3) > 0.1$. By the Intermediate Value theorem again, we know c is between 3 and 4. And we may continue the process, limited only by our patience. This is the process we call the bisection method:

1. Identify an interval $[a, b]$ such that either a or b overshoots the mark while the other undershoots it.

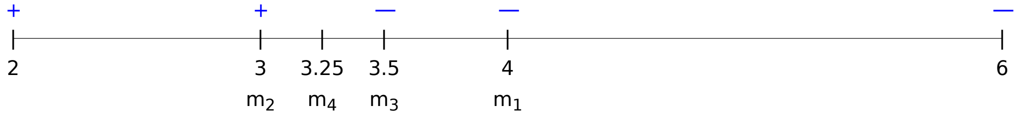
2. Calculate the midpoint, m, of the identified interval.

3. If a and m both overshoot or both undershoot the mark, the desired value lies in $[m, b]$.

4. If b and m both overshoot or both undershoot the mark, the desired value lies in $[a, m]$.

5. Return to step 2 using the newly identified interval.

Figure 2.1.1: + indicates $T_2(x) - \ln(x) > \frac{1}{10}$ and − indicates $T_2(x) - \ln(x) < \frac{1}{10}$.



Using a + sign for values of x for which $T_2(x) - \ln(x)$ overshoots the desired value $\frac{1}{10}$ and a − sign for values of x for which $T_2(x) - \ln(x)$ undershoots the desired value $\frac{1}{10}$, we may diagram this procedure, including the next two iterations, as in Figure 2.1.1. We might also reproduce the calculations in a table:

a	m	b	$T_2(a) - \ln(a)$	$T_2(m) - \ln(m)$	$T_2(b) - \ln(b)$
2	4	6	.3116	.04986	.002582
2	3	4	.3116	0.131	.04986
3	3.5	4	0.131	0.0824	.04986
3	3.25	3.5			

No matter how the procedure is understood, the sequence of approximations

$$4,\ 3,\ 3.5,\ 3.25,\ \ldots$$

is produced. What is the next value? Answer on page 30.

Not only do we have a sequence of numbers approaching the solution, we know for certain that 4 is accurate to within 2 units of the exact value. 3 is accurate to within 1 unit. 3.5 is accurate to within 0.5 units. And 3.25 is accurate to within 0.25 units. In general, each approximation is accurate to within half the length of the interval from which it was computed as midpoint. After all, the exact value is guaranteed to lie within the interval. The farthest the midpoint can possibly be from the exact value is half the length of the interval.

Though the method works perfectly well as described, normally the equation to be solved is simplified so that one side is zero. In that way, the other side can be thought of as a function whose roots are desired. Plus, it simplifies the implementation of the method slightly. For example, we would consider solving the equation

$$T_2(x) - \ln(x) - \frac{1}{10} = 0$$

instead of 2.1.1. Then the procedure boils down to finding a root of $f(x) = T_2(x) - \ln(x) - \frac{1}{10}$. This is why this method is called a root-finding method. It is used to find zeros, or roots, of functions. In this light, we might summarize the first 8 iterations of this procedure as follows:

a	m	b	$f(a)$	$f(m)$	$f(b)$
2	4	6	> 0	< 0	< 0
2	3	4	> 0	> 0	< 0
3	3.5	4	> 0	< 0	< 0
3	3.25	3.5	> 0	> 0	< 0
3.25	3.375	3.5	> 0	< 0	< 0
3.25	3.3125	3.375	> 0	< 0	< 0
3.25	3.28125	3.3125	> 0	> 0	< 0
3.28125	3.296875	3.3125			

Notice two things. The actual values of $f(a)$, $f(m)$, and $f(b)$ are not needed. Only their sign is important because all we need to do is maintain one endpoint where the function is greater than 0 (overshoots) and one where the function is less than 0 (undershoots). Furthermore, the $f(a)$ and $f(b)$ columns are not strictly necessary either. If the procedure is carried out faithfully, they will never change sign. In fact, that's what it means to carry out the procedure faithfully! In steps 3 and 4, you choose which subinterval to keep by maintaining opposite signs of the function on opposite endpoints.

As the last line indicates, the desired value is approximately 3.296 as promised. The other value, 13.13, is determined by finding a root of the function $g(x) = T_2(x) - \ln(x) + \frac{1}{10}$. Give it a shot! Start with $a = 10$ and $b = 14$, for example. Solution on page 30.

Though it works, the only real point of carrying out the procedure using a table is to make sure you understand exactly how it works. If we were actually to use the method in practice, we would write a short computer program instead. Computers are very good at repetitious calculations, something at which humans are not particularly adept. In this procedure, we need to calculate a midpoint, decide whether this midpoint should then become the left or right endpoint, make it so, and repeat.

That leaves only one question—how many repetitions, or iterations, should we compute? And that depends on the user. Perhaps an answer to within 10^{-2} of the exact value will suffice, and maybe only 10^{-6} accuracy will do. The program we write should be flexible enough to calculate the answer to whatever accuracy is desired, within reason. With that in mind, here is some pseudo-code for the bisection method.

The Bisection Method (pseudo-code)

Though technically not necessary for coding, when we can, we will preface each method's pseudo-code with mathematical assumptions that guarantee success. The implication is that if the method is run in a situation where the assumptions are not met, then the method should not be expected to provide dependable results. It may or may not give useful information. The old adage "garbage in...garbage out" applies!

```
Assumptions: $f$ is continuous on $[a, b]$. $f(a)$ and $f(b)$ have opposite signs.
Input: Interval $[a, b]$; function $f$; desired accuracy $tol$; maximum number of iterations $N$.
Step 1: Set $err = |b - a|$; $L = f(a)$;
Step 2: For $j = 1 \ldots N$ do Steps 3-5:
    Step 3: Set $m = \frac{a+b}{2}$; $M = f(m)$; $err = err/2$;
    Step 4: If $M = 0$ or $err \leq tol$ then return $m$;
    Step 5: If $LM < 0$ then set $b = m$; else set $a = m$ and $L = M$;
Step 6: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $m$ within $tol$ of exact root, or message of failure.
```

As noted earlier, this method should calculate a midpoint (Step 3), decide whether this midpoint should then become the left or right endpoint (Step 5), make it so (Step 5), and repeat some number of times (Steps 1, 2, and 4). Much of the code is dedicated to determining when to stop. This is typical of numerical methods. The calculations are half the battle. Controlling the calculations is the other half. If we didn't have to worry about stopping, the pseudo-code might look something like this:

```
Step 1: Set $L = f(a)$;
Step 2: Set $m = \frac{a+b}{2}$; $M = f(m)$;
Step 3: If $LM < 0$ then set $b = m$; else set $a = m$ and $L = M$;
Step 4: Go to Step 2.
```

There would be no need for j, err, tol, or N, making the algorithm quite a bit simpler. Of course, programmed this way, the program would never stop, so j, err, tol, and N, are indeed necessary. Nonetheless, this pseudo-code without the ability to stop is important. It can be thought of as the guts of the program. This is the code that executes the method. Sometimes it is easiest to start with the guts and then add the controls afterward.

As for determining whether the midpoint should become the left or right endpoint, Step 5 (Step 3 of the guts) uses a somewhat slick method. By slick, I mean short, efficient, and not immediately obvious. The sign of $LM = f(a) \cdot f(m)$ is checked. If it is negative ($LM < 0$) then m should become the right endpoint (should replace b) because this means $f(a)$ and $f(m)$ have opposite signs. That's the only way LM can be negative. On the other hand, if $LM > 0$ then we know $f(a)$ and $f(m)$ have the same sign, so m should become the left endpoint (should replace a). In Step 3 the midpoint is calculated without any fanfare.

The rest of the code is there to make sure the program doesn't do more than necessary and doesn't end up spinning its wheels indefinitely. It is important to be able to separate, at least in your mind, the guts of the program from the stopping logic. As for the stopping logic, in Step 4, we stop if $err \leq tol$ as we should. But we also check the unlikely event that $M = 0$ in which case we happened to hit the root exactly so should quit. Though it could be argued overkill to set a maximum number of iterations, N, in this program, it's a good habit to get into. Some numerical methods provide no guarantee the required tolerance will ever be reached. For these methods, a fallback exit criterion is needed. Also, if tol were accidentally set to a negative value, it would certainly never be reached. The algorithm would have no way to stop without N.

Analysis of the bisection method

There are two good reasons to study the bisection method. First, its assumptions for guaranteed success are much simpler to verify than those of other methods. Even so, be somewhat cautious. Faithful execution of any numerical method is subject to proper programming, accurate computation, and proper input. Programmers and users are not infallible. Nor are computers. Remember the lessons of Section 1.1. At the same time you should be wary of the results, you should temper your skepticism with a good dose of confidence in the method. It is only in rare circumstances that the computer will be the source of any problems.

Second, error analysis is straightforward. Let $m_1 = \frac{a+b}{2}$, the midpoint of $[a, b]$. Let succeeding midpoints be m_2, m_3, m_4, and so on. Then the Intermediate Value theorem guarantees $|m_j - p| \leq \frac{b-a}{2^j}$ for some root p of $f(x)$. As we learned in section 1.3, this means the sequence $\langle m_n \rangle$ converges to p with linear order, and rate of convergence $O\left(\frac{1}{2^n}\right)$. This method should be considered slow to converge because it does so with linear order. But among those methods with linear order, it should be considered fast. The error decays exponentially—faster than any polynomial decay.

Key Concepts

The Intermediate Value Theorem: Suppose f is a continuous function on $[a, b]$ and y is between $f(a)$ and $f(b)$. Then there is a number c between a and b such that $f(c) = y$.[1]

Iteration: (1) Repeating a computation or other process, using the output of one computation as the input of the next.

Iteration: (2) Any of the intermediate results of an iteration. Also called an iterate.

The bisection method: Produces a sequence of approximations $\langle m_j \rangle$ that converges to some root in $[a, b]$.

Error bound for the bisection method: The error of approximation m_j is no more than $\frac{b-a}{2^j}$. That is, $|m_j - p| \leq \frac{b-a}{2^j}$ for some root p of $f(x)$.

Convergence for the bisection method: The bisection method converges with linear order and has rate of convergence $O\left(\frac{1}{2^n}\right)$.

Exercises

1. C Write computer code implementing the bisection method as shown on page ??. Save it as a `.m` file for future use.

2. Use the Intermediate Value Theorem to show that the function has a root in the indicated interval.

   (a) $f(x) = 3 - x - \sin x$; $[2, 3]$

   (b) $g(x) = 3x^4 - 2x^3 - 3x + 2$; $[0, 1]$

   (c) $g(x) = 3x^4 - 2x^3 - 3x + 2$; $[0, 0.9]$ [S]

   (d) $h(x) = 10 - \cosh(x)$; $[-3, -2]$

   (e) $f(t) = \sqrt{4 + 5\sin t} - 2.5$; $[-6, -5]$

   (f) $g(t) = \frac{3t^2 \tan t}{1-t^2}$; $[21.5, 22.5]$ [S]

   (g) $h(t) = \ln(3\sin t) - \frac{3t}{5}$; $[1, 2]$

   (h) $f(r) = e^{\sin r} - r$; $[-20, 20]$

   (i) $g(r) = \sin(e^r) + r$; $[-3, 3]$

   (j) $h(r) = 2^{\sin r} - 3^{\cos r}$; $[1, 3]$

3. Create a table showing three iterations of the bisection method with the function and starting interval indicated in question 2. [S]

4. Use your `bisection.m` code to find a root of the function in the interval of question 2 to within 10^{-8}. [A]

5. Use the bisection method to find m_3 for the given function on the given interval. Do this without a computer program. Just use a pencil, paper, and a calculator. You may check your answers with a computer program if you wish. [A]

   (a) $f(x) = \sqrt{x} - \cos x$ on $[0, 1]$

   (b) $f(x) = 3(x + 1)(x - \frac{1}{2})(x - 1)$ on $[-1.25, 2.5]$

6. Use the Bisection Method to find m_4 for $g(x) = x \sin x + 1$ on $[9, 10]$.

7. Use the bisection method to find m_3 for the equation $x \cos x - \ln x = 0$ on the interval $[7, 8]$.

8. Use the bisection method to find a root of $g(x) = \sin x - x^2$ between 0 and 1 with absolute error no more than 1/4.

9. Approximate the root of $g(x) = 2 + x - e^x$ between 1 and 2 to within 0.05 of the exact value using the bisection method.

10. There are 21 roots of the function $f(x) = \cos(x)$ on the interval $[0, 65]$. To which root will the bisection method converge, starting with $a = 0$ and $b = 65$? [A]

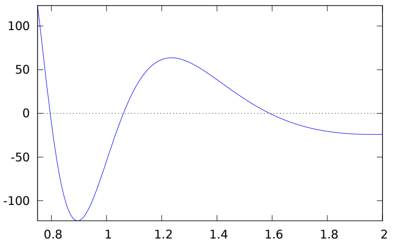
11. Find a bound on the number of iterations needed to achieve an approximation with accuracy 10^{-3} to the solution of $x^3 + x - 4 = 0$ on the interval $[1, 4]$ using

[1] The word “between” in this theorem can be interpreted as inclusive or exclusive of the endpoint values as long as the same interpretation is made for each instance of the word.

the bisection method. Do not actually compute the approximation. Just find the bound. [S]

12. Find a bound on the number of iterations needed to achieve an approximation with accuracy 10^{-4} to the solution of $x^3 - x - 1 = 0$ on the interval $[1, 2]$ using the bisection method. Do not actually compute the approximation. Just find the bound.

13. The graph of $f(x)$ over the interval $[0.75, 2]$ is shown below. Notice $f(x)$ has three roots on this interval: approximately .795, 1.06, and 1.59. To which of the three roots does the bisection method converge if we let $a = .75$ and $b = 2$? How do you know?



14. Suppose you are trying to find the root of $f(x) = x - e^{-x}$ using the bisection method. Find an integer a such that the interval $[a, a+2]$ is an appropriate one in which to start the search.

15. Find a lower bound on the number of iterations it would take to guarantee accuracy of 10^{-20} in question 6.

16. How many steps (iterations) of the bisection method are necessary to guarantee a solution with 10^{-10} accuracy if a root is known to be within $[4.5, 5.3]$? [A]

17. Suppose you are using the bisection method on an interval of length 3. How many iterations are necessary to guarantee accuracy of the approximation to within 10^{-6}?

18. Suppose a function g satisfies the assumptions of the bisection method on the given interval. Starting with that interval, how many iterations are needed to approximate the root to within the given tolerance?

    (a) $[-7, 10]$; 10^{-6}

    (b) $[5, 9]$; 10^{-3}

    (c) $[9, 15]$; 10^{-10}

    (d) $[-6, -1]$; 10^{-105} (assume the computer calculates with 300 significant digits so round-off error is not a problem)

19. 1 is a root of $f(x) = \ln(x^4 - x^3 - 7x^2 + 13x - 5)$ that can not be found by the bisection method.

    (a) Use a graph of the function near 1 to explain why. You may use the code below to produce an appropriate graph.

    (b) Run the bisection method on f over the interval $[0.8, 1.2]$ anyway. What happens instead of finding the root?

```
x=0.8:.01:1.2;
f=inline("log(x.^4-x.^3-7*x.^2+13*x-5)");
plot(x,f(x))
```

20. 4 is a root of $g(x) = |\sin(\pi x)|$ that can not be found by the bisection method.

    (a) Use a graph of the function near 4 to explain why. You may use the code below to produce an appropriate graph.

    (b) Run the bisection method on f over the interval $[3.5, 4.5]$ anyway. What happens instead of finding the root?

```
x=3.5:.05:4.5;
f=inline("abs(sin(pi*x))");
plot(x,f(x))
```

21. Let $f(x) = \sin(x^2)$. f is continuous on $[4, 5]$, but $f(4) < 0$ and $f(5) < 0$, so the assumptions of the bisection method are not met. Nonetheless, using the bisection method as described in the pseudo-code on f over the interval $[4, 5]$ *does* produce a root. Explain. [S]

22. The functions in questions 2e, 2f, and 2g all fail to meet the assumptions of the bisection method on the interval $[-4, -0.5]$. For each one, explain how so.

23. Write computer code called `collatz` that takes one integer input, n, and returns $3n+1$ if n is odd and $n/2$ if n is even. Save it as a `collatz.m` file. Use an `if then else` statement in your function. HINT: Use the ceiling function. If `ceil(n/2)` equals `n/2`, then `n` must be even (no remainder when divided by 2). Use your `collatz` function to calculate [A]

    (a) `collatz(17)`

    (b) `collatz(10)`

    (c) `collatz(109)`

    (d) `collatz(344)`

24. Write your own absolute value function called `absval` (`abs` is already defined on the computer, so it is best to use a different name) that takes a real number input and returns the absolute value of the input. Use an `if then else` statement in your function. Save it as `absval.m` and test it on the following computations.

    (a) $|-3|$

    (b) $|123.2|$

    (c) $\left|\pi - \frac{22}{7}\right|$

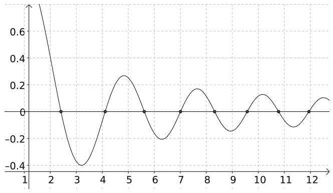
    (d) $|10 - \pi^2|$

25. $f(x) = \sin(x^2)$ has five roots on the interval $[7, 8]$. $f(7) < 0$, $f(8) > 0$, and f is continuous on $[7, 8]$, so the assumptions of the bisection method are met. The method will converge to a root.

    (a) Use your `bisection.m` file (Exercise 1) to determine which one. [A]

    (b) Find 4 different intervals for which the bisection method will converge to the other four roots in $[7, 8]$.

26. The function shown has roots at approximately 2.41, 4.11, 5.62, 7.01, 8.32, 9.57, 10.78, and 11.94. To which root will the bisection method converge with the given starting interval?



(a) $[2, 3]$

(b) $[6, 8]$

(c) $[2, 6]$

(d) $[5, 9]$

(e) $[10, 12]$ Note: the assumptions of the bisection are not met on this interval. Nonetheless, the method as outlined in the pseudo-code *will* converge to a root!

27. Find an interval of length 1 over which the bisection method may be applied in order to find a root of $f(x) = x^4 - 7.6746x^3 - 40.7477022x^2 + 200.9894434x + 319.0914281$.

28. The following algorithm is one possible incarnation of the bisection method.

```
Assumptions: $f$ is continuous on $[a, b]$. $f(a)$ and $f(b)$ have opposite signs.
Input: Interval $[a, b]$; function $f$
Step 1: For $j = 1 \dots 15$ do Steps 2 and 3:
    Step 2: Set $m = \frac{a+b}{2}$;
    Step 3: If $f(a)f(m) < 0$ then set $b = m$; else set $a = m$;
Step 4: Print $m$.
Output: Approximation $m$.
```

(a) Apply this algorithm to the function $f(x) = (x)(x-2)(x+2)$ over the interval $[-3, 3]$. Which root will this algorithm approximate?

(b) How accurate is the approximation guaranteed to be according to the formula

$$|p_n - p| \leq \frac{b-a}{2^n}?$$

(c) How accurate is the approximation in reality? Compare this to the bound in (b).

(d) Modify the algorithm so it will approximate a different root using the same starting interval.

(e) Modify the algorithm so it does not use multiplication.

29. Use the following pseudo-code to write a slightly different implementation of the bisection method. Refer to Table **??** if you are unsure how to program the quantity $\lceil(\ln(b-a) - \ln(TOL))/\ln 2\rceil$. The while loop is discussed on page ??.

```
Input function $f$, endpoints $a$ and $b$; tolerance $TOL$.
Return approximate solution $p$ and $f(p)$ and the number of iterations done $N_0$.
Step 1 Set $i = 1$; $FA = f(a)$; $N_0 = \lceil(\ln|b-a| - \ln(TOL))/\ln 2\rceil$;
Step 2 While $i \leq N_0$ do Steps 3-6.
    Step 3 Set $p = (a+b)/2$; $FP = f(p)$;
    Step 4 If $FP = 0$ then
        Return($p$, $f(p)$, $N_0$); STOP.
    Step 5 Set $i = i + 1$;
    Step 6 If $FA \cdot FP > 0$ then
        Set $a = p$; $FA = FP$;
        else
        Set $b = p$;
Step 7 Return($p$, $f(p)$, $N_0$);
    STOP.
```

(a) Discuss the advantages/disadvantages of this algorithm compared to the one on page ??.

(b) Where does the calculation $N_0 = \lceil(\ln(b-a) - \ln(TOL))/\ln 2\rceil$ come from?

30. Use the code you wrote for question 29 to find solutions accurate to within 10^{-5} for the following problems.

(a) $x - 2^x = 0$ on $[0, 1]$

(b) $e^x - x^2 + 3x - 2 = 0$ on $[0, 1]$

(c) $2x\cos(2x) - (x+1)^2 = 0$ on $[-3, -2]$ and on $[-1, 0]$

31. Find an approximation of $\sqrt{3}$ correct to within 10^{-4} using the bisection method. Write an essay on how you solved this problem. Include your bisection code, what function and what interval you used and why.

32. A trough of length L has a cross section in the shape of a semicircle with radius r. When filled with water to within a distance h of the top, the volume V of water is

$$V = L\left[0.5\pi r^2 - r^2 \arcsin\left(\frac{h}{r}\right) - h\sqrt{r^2 - h^2}\right]$$

Suppose $L = 10$ ft, $r = 1$ ft, and $V = 12.4$ ft^3. Find the depth of the water in the trough to within 0.01 ft. Note: Use `asin(x)` for $\arcsin(x)$ and `pi` for π.

Answers

What is the next value?: $T_2(3.25) - \ln(3.25) \approx .10429$, which overshoots the mark. So 3.25 becomes the new left endpoint, and the next value is $\frac{3.25+3.5}{2} = 3.375$, the midpoint of 3.25 and 3.5.

The right endpoint is 13.13**:** Starting with $a = 10$ and $b = 14$, note that $g(a) \approx .088 > 0$ and $g(b) \approx -.044 < 0$, so g of the left endpoint should always be positive and g of the right endpoint should always be negative:

a	m	b	$g(m)$	
10	12	14	.044	⇒ m becomes left endpoint
12	13	14	.006	⇒ m becomes left endpoint
13	13.5	14	$-.017$	⇒ m becomes right endpoint
13	13.25	13.5	$-.005$	⇒ m becomes right endpoint
13	13.125	13.25	.0004	⇒ m becomes left endpoint
13.125	13.1875	13.25	$-.002$	⇒ m becomes right endpoint
13.125	13.15625	13.1875	$-.0009$	⇒ m becomes right endpoint
13.125	13.140625	13.15625	$-.0002$	⇒ m becomes right endpoint
13.125	13.1328125	13.140625		

2.2 Fixed Point Iteration

Grab your calculator. Anything with a cosine button will do nicely. Presuming you have a simple scientific calculator, press the all-clear button, usually marked `AC` or just `C`. The screen should now display 0. Press the cosine button, which should be marked `cos`. The screen should display 1. Press the cosine button again. The screen should display 0.540302 . . .. Repeat. Repeat again. In fact, continue pressing the cosine button until you notice a pattern.

If you have a fancier calculator with a previous-answer button, usually marked `Ans`, press 0 and then `Enter` or `=`. Then press the cosine button and then the previous-answer button. Then press `Enter` or `=` to do the computation. The first time around, the screen should display 1 (just as with a scientific calculator). To repeat, however, just press `Enter` or `=` again. This will repeat the last computation. In this case, the cosine of the previous answer. The screen should display 0.540302 . . .. Now repeat until you notice a pattern.

After about 30 repetitions, or, as we will call them, iterations, your calculator should display a number like 0.739083847 . . .. And no matter how many times you repeat, or iterate, the calculation, it won't change much. In fact, once it reaches 0.7390851332 . . ., it won't change at all (unless your calculator shows more decimal places—after about 90 iterations, a calculator showing 15 decimal places will display 0.739085133215161 and it won't change from there). What that means is $\cos(0.7390851332\ldots) = 0.7390851332\ldots$. And we call 0.7390851332 . . . a fixed point of the cosine function. The value is fixed (does not change) when the cosine function is applied. Put another way, at 0.7390851332 . . ., the input and output of the cosine function are equal. See a simulation of this iteration online at the companion website.

Perhaps a whole series of questions now comes to mind. Why does this work? What if we start with a number other than 0? Does this work with any function? Can we predict when it will or won't work? Can we find roots this way? Is convergence fast? In this section and the next, we will give at least partial answers to all of these questions. We start with "Why does this work?".

Consider solving the system

$$\begin{cases} y = \cos(x) \\ y = x \end{cases}.$$

One way to do so is by the method of substitution. If we substitute $y = x$ into $y = \cos x$ we get $x = \cos x$ or $\cos x = x$. The solutions of the system coincide exactly with the fixed points of the cosine function, for any solution of $\cos x = x$ is a value x that is fixed by the cosine. Since systems of two equations in two unknowns can be solved, at least approximately, by graphing, this suggests that we might take a look at the graph of the system in order to learn more about what is happening during iteration.

Figure 2.2.1: Finding the fixed point of $\cos(x)$.

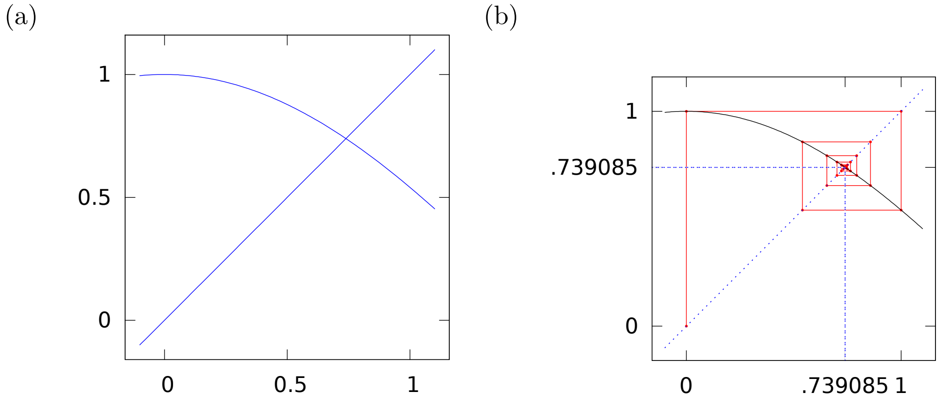


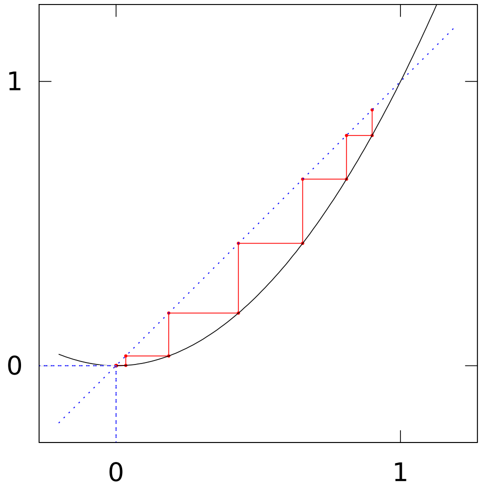
Figure 2.2.1(a) shows the graphs of $y = \cos(x)$ and $y = x$ over the interval $[0, 1]$. We can see the intersection at around $(0.75, 0.75)$ so we should think that the fixed point is around 0.75 (which of course we know is true from our calculator experiment). Figure 2.2.1(b) illustrates the exercise of computing $\cos(0), \cos(1), \cos(0.540302\ldots), \ldots$. Following the vertical line segment from $(0, 0)$ to $(0, 1)$ represents calculating $\cos(0)$. Following the horizontal continuation from $(0, 1)$ to $(1, 1)$ and subsequently the vertical line segment from $(1, 1)$ to $(1, 0.540302\ldots)$ represents calculating $\cos(1)$. Following the horizontal line from $(1, 0.540302\ldots)$ to $(0.540302\ldots, 0.540302\ldots)$ and subsequently the vertical line from $(0.540302\ldots, 0.540302\ldots)$ to $(0.540302\ldots, 0.857553\ldots)$ represents calculating

$\cos(0.540302\ldots)$, and so on. With each pair of line segments, one going horizontally from the graph of $y = \cos(x)$ to the graph of $y = x$ followed by one going vertically from the line $y = x$ to the graph of $y = \cos(x)$, another iteration is shown. Figure 2.2.1(b) is sometimes called a web diagram [2], and is commonly used to illustrate the concept of iteration. That the path of the web diagram tends toward $(0.739085\ldots, 0.739085\ldots)$ is an unavoidable consequence of the geometry of the graph of $\cos(x)$.

What if we start with a number other than 0? Using figure 2.2.1, you should be able to convince yourself that convergence to the point $(0.7390851332\ldots, 0.7390851332\ldots)$ is assured for any initial value between 0 and 1. Try it. Start anywhere on the line $y = x$. Proceed vertically to the graph of $y = \cos(x)$. Then horizontally to the line $y = x$. And repeat. You should find that the path of the web diagram always tends toward the intersection of the graphs. Now consider starting with any real number, r. The cosine of any real number is a number in the interval $[-1, 1]$ so $\cos(r) \in [-1, 1]$. And the cosine of any number in the interval $[-1, 1]$ is a number in the interval $[0, 1]$ so $\cos(\cos(r)) \in [0, 1]$. That is, the second iteration is in the interval from 0 to 1. So after only two iterations, any initial value will become a value between 0 and 1. And our web diagram implies that further iteration will lead to the fixed point. So, regardless of the initial value, iteration leads to the fixed point. And the preceding argument forms the seed for a proof of this fact.

Not all functions are so well behaved, however. For example, $1^2 = 1$. In other words, 1 is a fixed point of the function $y = x^2$. However, iteration starting with any number other than 1 or -1 does not lead to this fixed point. If we start with any number greater than 1 and square it, it becomes greater. And if we square the result, it becomes greater still. And squaring again only increases the value, without bound. Hence, iteration starting with any value greater than 1 (or less than -1) does not lead to convergence to the fixed point 1. Nor does iteration starting with any number of magnitude less than 1. Figure 2.2.2 illustrates iteration of $y = x^2$ with initial value 0.9.

Figure 2.2.2: Visualizing the iteration of $f(x) = x^2$.

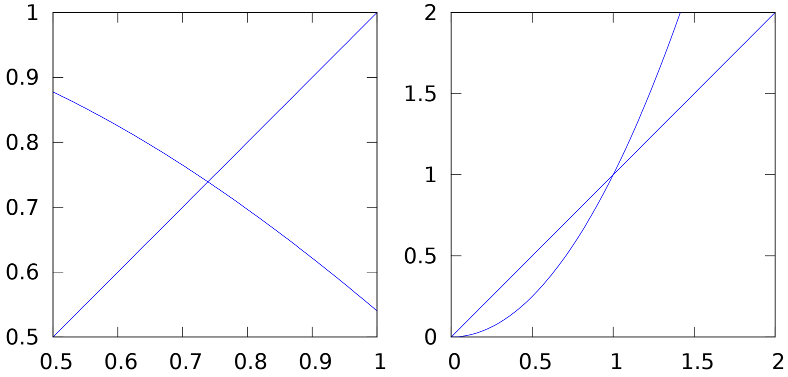


Follow the web diagram from the point $(0.9, 0.9)$ vertically to the graph of $y = x^2$ and then horizontally back to the line $y = x$, and so on, to check for yourself. This is a nice illustration of the fact that the square of any number between 0 and 1, exclusive, is smaller than the number itself. With starting values between -1 and 1 exclusive of ±1, iteration gives a sequence converging to 0, not 1. To summarize, excepting -1 and 1, no initial value will produce a sequence converging to 1 under iteration of the function $y = x^2$.

There is a fundamental difference between the fixed point $0.7390851332\ldots$ of $f(x) = \cos(x)$ and the fixed point 1 of $g(x) = x^2$. Fixed point iteration converges to $0.7390851332\ldots$ under $f(x) = \cos(x)$ for any initial value. Fixed point iteration fails to converge to 1 under $g(x) = x^2$ for all initial values but ±1.[2] Examining the graphs of $f(x)$ and $g(x)$ each superimposed against the line $y = x$ in the neighborhood of their respective fixed points can give a clue [Figure 2.2.3] as to the difference. True, $f(x)$ has a negative slope at its fixed point while $g(x)$ has a positive slope at its fixed point. You can see this from the graphs or you can "do the calculus". The important difference, though, is more subtle. It's not the sign of the slope at the fixed point that matters. It's the magnitude of the slope at the fixed point that matters. For smooth functions, neighborhoods of points with slopes of magnitude greater than 1 tend to be expansive. That is, points move away from one another under application of the function.

[2] For a third type of behavior, fixed point iteration converges to 0 under $g(x)$ for initial values near 0, but not for others!

Figure 2.2.3: Left: $f(x) = \cos(x)$ and $y = x$. Right: $g(x) = x^2$ and $y = x$.



However, neighborhoods of points with slopes of magnitude less than 1 tend to be contractive. That is, points move toward one another under application of the function.

Proposition 2. *If $h(x)$ is differentiable on (a, b) with $|h'(x)| < 1$ for all $x \in (a, b)$, then whenever $x_1, x_2 \in (a, b)$, $|h(x_2) - h(x_1)| < |x_2 - x_1|$.*

Proof. Let $x_1, x_2 \in (a, b)$ and, without loss of generality, let $x_2 > x_1$ so that we may properly refer to the interval from $x_1$to x_2. Since $h(x)$ is continuous on $[x_1, x_2]$ and differentiable on (x_1, x_2), the mean value theorem gives us $c \in (x_1, x_2) \subseteq (a, b)$ such that $h'(c) = \left|\frac{h(x_2)-h(x_1)}{x_2-x_1}\right|$. But $h'(c) < 1$ by assumption, so $h'(c) = \left|\frac{h(x_2)-h(x_1)}{x_2-x_1}\right| < 1$, from which we immediately conclude that $|h(x_2) - h(x_1)| < |x_2 - x_1|$. □

Moreover, a function whose derivative has magnitude less than 1 can only cross the line $y = x$ one time. Once it has crossed, it can never "catch up" because that would require a slope greater than 1, the slope of the line $y = x$.

Proposition 3. *Suppose $h(x)$ is continuous on $[a, b]$, differentiable on (a, b) with $|h'(x)| < 1$ for all $x \in (a, b)$, and $h([a, b]) \subseteq [a, b]$. Then h has a unique fixed point in $[a, b]$.*

Proof. If $h(a) = a$ or $h(b) = b$, we have proved existence, so suppose $h(a) \neq a$ and $h(b) \neq b$. Since $h([a, b]) \subseteq [a, b]$ it must be the case that $h(a) > a$ and $h(b) < b$. It immediately follows that $h(a) - a > 0$ and $h(b) - b < 0$. Since the auxiliary function $f(x) = h(x) - x$ is continuous on $[a, b]$, the Intermediate Value Theorem guarantees the existence of $c \in (a, b)$ such that $f(c) = 0$. By substitution, $h(c) - c = 0$, implying $h(c) = c$, so c is a fixed point of h. The existence of a fixed point is established. Now suppose $c_1 \in [a, b]$ and $c_2 \in [a, b]$ are distinct fixed points of h. Then

$$\frac{h(c_1) - h(c_2)}{c_1 - c_2} = \frac{c_1 - c_2}{c_1 - c_2} = 1.$$

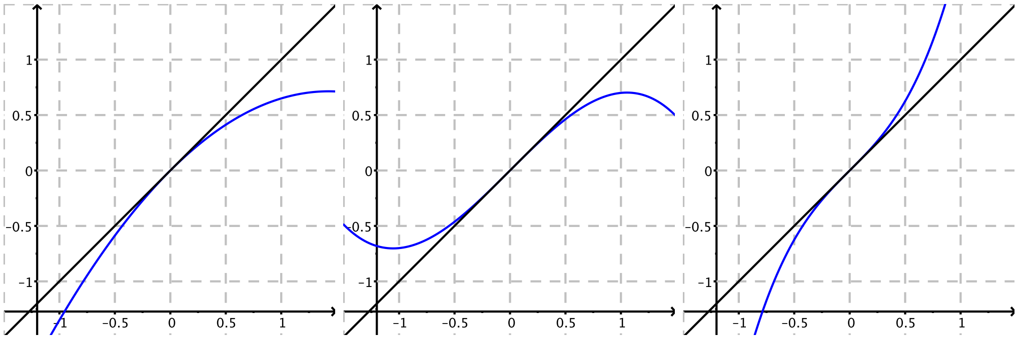
By the mean value theorem, there exists c_3 between c_1 and c_2 such that $h'(c_3) = 1$, contradicting the fact that $|h'(x)| < 1$ for all $x \in (a, b)$. Hence, it is impossible that c_1 and c_2 are distinct. □

Hence, we can reasonably expect that when the derivative at a fixed point has magnitude less than 1, iteration is a viable method for approximating (finding) the fixed point, but when the derivative at a fixed point has magnitude greater than 1, iteration is not a viable method of approximating the fixed point. We must be careful, though, not to take this rule of thumb as absolute. It only applies to so-called well-behaved functions. In this case, that the function has a continuous first derivative in the neighborhood of the fixed point is well-behaved enough. The following theorem establishes that fixed point iteration will converge in a neighborhood of a fixed point if the magnitude of the function's derivative is less than 1 there.

Theorem 4. *(Fixed Point Convergence Theorem) Given a function $f(x)$ with continuous first derivative and fixed point $\hat{x}$, if $|f'(\hat{x})| < 1$ then there exists a neighborhood of $\hat{x}$ in which fixed point iteration converges to the fixed point for any initial value in the neighborhood.*

Proof. By continuity, there exists $\varepsilon > 0$ such that $|f'(x)| < 1$ for all $x \in (\hat{x} - \varepsilon, \hat{x} + \varepsilon)$. Let $0 < \delta < \varepsilon$ and set $M = \max\limits_{x \in [\hat{x}-\delta, \hat{x}+\delta]} |f'(x)|$. Now suppose x_0 is a particular but arbitrary value in $(\hat{x} - \delta, \hat{x} + \delta)$. As in proposition 2, the Mean Value Theorem is applied. This time, we are guaranteed $c \in (\hat{x} - \delta, \hat{x} + \delta)$ such that $f'(c) = \frac{f(\hat{x})-f(x_0)}{\hat{x}-x_0}$.

Figure 2.2.4: Convergence behavior when the derivative at the fixed point is 1.



But $|f'(c)| \leq M$ so $|f(\hat{x}) - f(x_0)| \leq M|\hat{x} - x_0|$. Furthermore $\hat{x}$ is a fixed point, so $f(\hat{x}) = \hat{x}$, from which it follows that $|\hat{x} - f(x_0)| \leq M|\hat{x} - x_0|$. Now we define $x_k = f(x_{k-1})$ for all $k \geq 1$ and prove by induction that $|\hat{x} - x_k| \leq M^k|\hat{x} - x_0|$ for all $k \geq 1$. Since $x_1 = f(x_0)$, we have already shown $|\hat{x} - x_1| \leq M|\hat{x} - x_0|$, so the claim is true when $k = 1$. Now suppose $|\hat{x} - x_k| \leq M^k|\hat{x} - x_0|$ for some particular but arbitrary value $k \geq 1$. Note that $|\hat{x} - x_k| \leq M^k|\hat{x} - x_0|$ implies $x_k \in (\hat{x} - \delta, \hat{x} + \delta)$ so we apply the Mean Value Theorem as before and conclude that $|\hat{x} - f(x_k)| \leq M|\hat{x} - x_k|$. Substituting x_{k+1} for $f(x_k)$ and using the inductive hypothesis, we have $|\hat{x} - x_{k+1}| \leq M \cdot M^k|\hat{x} - x_0| = M^{k+1}|\hat{x} - x_0|$. Hence, we have $0 \leq |\hat{x} - x_k| \leq M^k|\hat{x} - x_0|$. Of course $\lim\limits_{k\to\infty} 0 = 0$ and $\lim\limits_{k\to\infty} M^k|\hat{x} - x_0| = 0$, so by the squeeze theorem, $\lim\limits_{k\to\infty} |\hat{x} - x_k| = 0$. □

As suggested earlier, we should not expect fixed point iteration to converge when the derivative at a fixed point has magnitude greater than one. In fact, more or less the opposite happens. There is a neighborhood of the fixed point in which fixed point iteration is guaranteed to escape the neighborhood for any initial value in the neighborhood not equal to the fixed point itself. Given that fact, it is tempting to think that perhaps the Fixed Point Convergence Theorem could be strengthened to a bi-directional implication, an if-and-only-if claim. And it "almost" can. What can be said here has direct parallels to the ratio test for series. Recall, for any sequence of real numbers $a_0, a_1, a_2, \ldots$, the limit $L = \lim\limits_{k\to\infty} \left|\dfrac{a_{k+1}}{a_k}\right|$ helps determine the convergence of $\sum\limits_{k=0}^{\infty} a_k$ in the following way:

- If $L < 1$, then $\sum\limits_{k=0}^{\infty} a_k$ converges (absolutely).
- If $L > 1$, then $\sum\limits_{k=0}^{\infty} a_k$ diverges.
- If $L = 1$, then $\sum\limits_{k=0}^{\infty} a_k$ may converge (absolutely or conditionally) or may diverge.

Analogously, for any function $f(x)$ with continuous first derivative and fixed point $\hat{x}$, the derivative $f'(\hat{x})$ helps determine the convergence of the fixed point iteration method in the following way:

- If $|f'(\hat{x})| < 1$, then fixed point iteration converges to $\hat{x}$ for any initial value in some neighborhood of $\hat{x}$.
- If $|f'(\hat{x})| > 1$, then fixed point iteration escapes some neighborhood of $\hat{x}$ for any initial value in the neighborhood other than $\hat{x}$.
- If $|f'(\hat{x})| = 1$, then fixed point iteration may converge to $\hat{x}$ for any initial value in some neighborhood of $\hat{x}$; or may escape some neighborhood for any initial value in the neighborhood other than $\hat{x}$; or may have no neighborhood in which all initial values lead to convergence and no neighborhood in which all values other than $\hat{x}$ escape.

The graphs in Figure 2.2.4 of functions with derivative equal to one at their fixed point help illustrate this last case.

For one of these functions, fixed point iteration converges for all values in a neighborhood of the fixed point. For another of these functions, fixed point iteration escapes some neighborhood of the fixed point for all initial values in the neighborhood except the fixed point itself. And for the third of these functions, fixed point iteration converges to the fixed point for some initial values and escapes a neighborhood of the fixed point for others (and *every* neighborhood of the fixed point will have both types of initial values). Can you tell which is which? Figure it out by creating web diagrams for each. Answer on page 41.

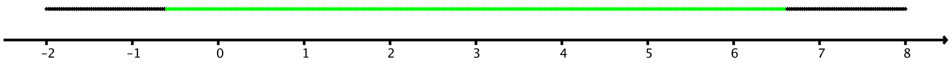
The proof of the Fixed Point Convergence Theorem can easily be extended to include initial values in any neighborhood of the fixed point in which the magnitude of the derivative remains less than 1. The size and symmetry of the interval are not important. For example, $f(x) = \frac{1}{8}x^3 - x^2 + 2x + 1$ has a fixed point at $\hat{x} = 2$. The proof of the Fixed Point Convergence Theorem establishes convergence to 2 in a symmetric interval about 2 such as $[1.9, 2.1]$. But this interval is far from the largest neighborhood of initial values for which fixed point iteration converges to 2. We can find bounds on the largest such interval by solving the equation $|f'(x)| = 1$. To that end:

$$\begin{array}{rcl}
\frac{3}{8}x^2 - 2x + 2 & = & \pm 1 \\
3x^2 - 16x + 16 & = & \pm 8 \\
3x^2 - 16x + 24 = 0 & \textit{or} & 3x^2 - 16x + 8 = 0 \\
x = \dfrac{8 \pm i2\sqrt{2}}{3} & \textit{or} & x = \dfrac{8 \pm 2\sqrt{10}}{3} \\
\dfrac{8 - 2\sqrt{10}}{3} \approx 0.558 & \textit{and} & \dfrac{8 + 2\sqrt{10}}{3} \approx 4.775,
\end{array}$$

so we should expect fixed point iteration to converge to 2 on any closed interval contained in

$$\left(\frac{8 - 2\sqrt{10}}{3}, \frac{8 + 2\sqrt{10}}{3}\right).$$

Now, if we have the computer execute fixed point iteration for a large number of evenly spaced initial values, say 100, on the interval $[-2, 8]$ and record the results on a number line where we color an initial value black if it does not converge to 2 and green if it does converge to 2 (we will call such diagram a convergence diagram), we get



,

which shows that fixed point iteration converges to 2 on approximately $[-0.5, 6.5]$. Indeed, the experiment confirms that fixed point iteration converges on any closed interval contained in $\left(\frac{8-2\sqrt{10}}{3}, \frac{8+2\sqrt{10}}{3}\right)$ as predicted. But the diagram shows convergence on an even larger set. We can conclude that the Fixed Point Convergence Theorem gives sufficient but not necessary conditions for convergence in a neighborhood of a fixed point.

A graph of the function $f(x)$ superimposed on the line $y = x$ (Figure 2.2.5) gives some insight as to why the bounds $\frac{8\pm2\sqrt{10}}{3}$ do not tell a complete story. By imagining the web diagram for any initial value between the two fixed points other than 2, that is -0.61 and 6.61, you should be able to convince yourself that fixed point iteration converges to 2 for any initial value in the interval $(-0.61, 6.61)$. Can you prove it? Graphs like those in Figures 2.2.3, 2.2.4, and 2.2.5 are indispensable and should always be consulted when trying to understand fixed point iteration, but they should not be relied upon as proof. For that, we need to rely on theorems like the Fixed Point Convergence Theorem.

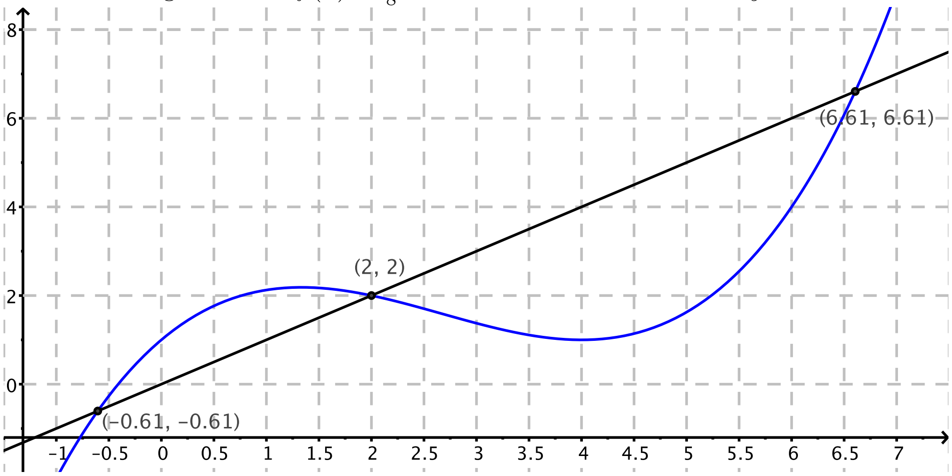
Crumpet 9: One interesting quadratic

Trying to find roots of the logistic equation

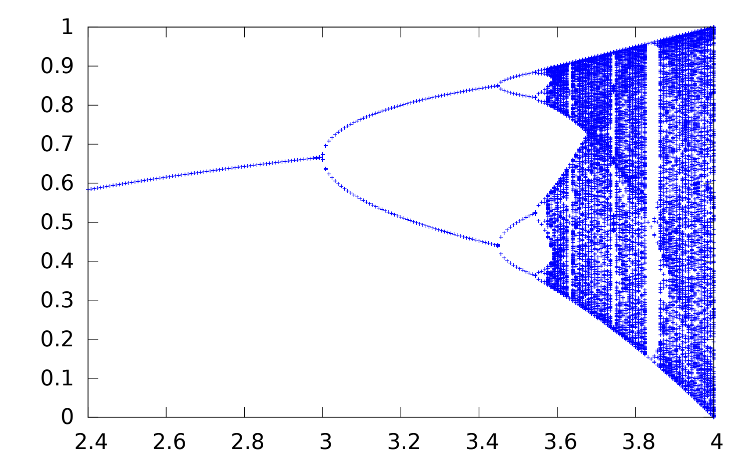
$$g(x) = (\alpha - 1)x - \alpha x^2$$

by applying fixed point iteration to the corresponding function $f(x) = x + g(x) = \alpha x(1 - x)$ is a famous exercise in dynamical systems which has a nasty habit of not working! Complete the following investigation to see what happens.

Figure 2.2.5: $f(x) = \frac{1}{8}x^3 - x^2 + 2x + 1$ and the line $y = x$



1. Show that $f(x) = \alpha x(1-x)$ as claimed.
2. For each of the values $\alpha = 2.5$, $\alpha = 3.2$, $\alpha = 3.833$, and $\alpha = 4$, do the following.
   (a) Find the positive fixed point of f (root of g) analytically (using a pencil, paper, and some algebra).
   (b) Set $x_0 = 0.1$ and use a computer program to calculate x_{975} through x_{1000}.
   (c) Examine the 26 iterations of part (b) and describe what you see.
3. Draw a connection between your results from part 2 and the following diagram.



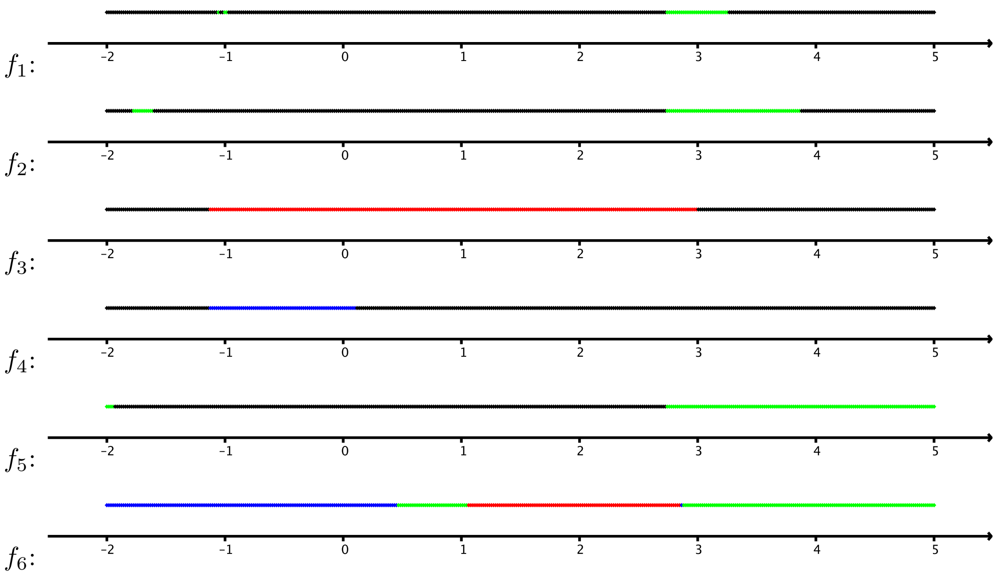
4. Use the diagram to predict a value of α for which you would expect fixed point iteration to lead to x_{975} through x_{1000} cycling through 4 different values. Check your prediction.

Root Finding

When successful, fixed point iteration finds solutions of an equation of the form $f(x) = x$. A root finding problem requires the solution of an equation of the form $g(x) = 0$. However, the equation $f(x) = x$ has exactly the same solutions as the equation $f(x) - x = 0$, so finding fixed points of $f(x)$ is equivalent to finding roots of $g(x) = f(x) - x$. Indeed, we can rephrase the example of finding fixed points of $f(x) = \frac{1}{8}x^3 - x^2 + 2x + 1$ as the problem of finding roots of $g(x) = f(x) - x = \frac{1}{8}x^3 - x^2 + x + 1$. But it is the opposite problem that is much more common. We have the question of finding the roots of a function and need to rephrase it in terms of a fixed point problem.

Suppose we want the roots of $g(x) = -x^3 + 5x^2 - 4x - 6$. We can rephrase the question of solving $g(x) = 0$ as

Figure 2.2.6: Convergence diagrams for 6 functions with the same fixed points.



black: does not converge; green: converges to 3; red: converges to $1+\sqrt{3}$; blue: converges to $1-\sqrt{3}$

the problem of finding the fixed points of many different functions! But you will have to ignore some sage advice of your algebra teacher to derive them! The key is to use algebra to rewrite the equation $-x^3+5x^2-4x-6=0$ as an equation of the form $x=f(x)$. The simplest way is to add x to both sides of the equation. This manipulation and several others are shown in the following list.

- $-x^3+5x^2-4x-6=0 \Rightarrow -x^3+5x^2-3x-6=x$
- $-x^3+5x^2-4x-6=0 \Rightarrow -x^3+5x^2-6=4x \Rightarrow \frac{-x^3+5x^2-6}{4}=x$
- $-x^3+5x^2-4x-6=0 \Rightarrow -x^3-4x-6=-5x^2 \Rightarrow \frac{x^3+4x+6}{5}=x^2 \Rightarrow \pm\sqrt{\frac{x^3+4x+6}{5}}=x$
- $-x^3+5x^2-4x-6=0 \Rightarrow 5x^2-4x-6=x^3 \Rightarrow \sqrt[3]{5x^2-4x-6}=x$

Can you see what has been done for each one? Thus, we have five candidates for fixed point iteration, $f_1(x)=-x^3+5x^2-3x-6$, $f_2(x)=\frac{-x^3+5x^2-6}{4}$, $f_3(x)=\sqrt{\frac{x^3+4x+6}{5}}$, $f_4(x)=-\sqrt{\frac{x^3+4x+6}{5}}$, and $f_5(x)=\sqrt[3]{5x^2-4x-6}$, all of which will potentially give roots of $g(x)$. There is a sixth function we will discuss in much more detail later: $f_6(x)=\frac{2x^3-5x^2-6}{3x^2-10x+4}$[3]. The roots of $g(x)$ are $1-\sqrt{3}\approx-0.73$, $1+\sqrt{3}\approx 2.73$, and 3, so we will consider convergence diagrams over the interval $[-2,5]$. Fixed point iteration converges to different fixed points for the different functions despite the fact that all 6 functions have exactly the same three fixed points. The convergence diagrams of Figure 2.2.6 are color-coded to reflect this fact. Black indicates lack of convergence just as before. Green, red, and blue indicate convergence to 3, $1+\sqrt{3}$, and $1-\sqrt{3}$, respectively. Notice that only f_6 provides convergence for, as far as we can tell, every initial value in $[-2,5]$, and is also the only one for which fixed point iteration converges to different fixed points for different initial values. See if you can understand why each function has the convergence behavior it does by looking at the graphs of $f_1, f_2, \ldots, f_6$. Pay special attention to the graphs around $1+\sqrt{3}$ and 3. Looks can be deceiving in that area because the two fixed points are so close together. Also, see if you can find two initial values in $[-2,5]$ for which fixed point iteration on f_6 does not converge. What happens instead? For an extra challenge, see if you can find a third point in $[-2,5]$ for which fixed point iteration on f_6 does not converge. Hint: you may need to use a computer algebra system to find such a point exactly or use fixed point iteration to approximate it! Answers on page 41.

[3] By calculating $f_6(1-\sqrt{3})$, $f_6(1+\sqrt{3})$, and $f_6(3)$, you can verify that f_6 has these three values as fixed points as well.

The Fixed Point Iteration Method (pseudo-code)

Though we spent a lot of time talking about how to determine whether we should expect the fixed point iteration method to converge or not, none of that information is strictly relevant to coding the method. Any implementation of the method should allow the user to try fixed point iteration for any function with any initial value. It is the user's responsibility to understand that when the assumptions are not met, the results are unpredictable. Remember, "garbage in...garbage out."

The fixed point iteration method presents a problem that the bisection method did not. In the bisection method, there was a simple and convenient formula for an upper bound on the error. To provide something similar in the fixed point iteration method, one would have to sacrifice simplicity or convenience or both, but the benefits do not outweigh the sacrifice. Instead, a more general stopping criterion is used. When two consecutive iterations are closer to one another than a given tolerance, the method stops. At this point, the difference between iterations, say x_k and x_{k+1}, is smaller than the tolerance. For a sequence derived from fixed point iteration, $x_{k+1} = f(x_k)$, so $|x_{k+1} - x_k| = |f(x_k) - x_k|$. When $|x_{k+1} - x_k|$ is small, $|f(x_k) - x_k|$ is small, so $f(x_k) \approx x_k$. x_k is "almost" a fixed point.

```
Assumptions: $f$ is differentiable. $f$ has a fixed point $\hat{x}$. $x_0$ is in a neighborhood $(\hat{x} - \delta, \hat{x} + \delta)$ where the magnitude of $f'$ is less than one.
Input: Initial value $x_0$; function $f$; desired accuracy $tol$; maximum number of iterations $N$.
Step 1: For $j = 1 \ldots N$ do Steps 2-4:
    Step 2: Set $x = f(x_0)$;
    Step 3: If $|x - x_0| \leq tol$ then return $x$;
    Step 4: Set $x_0 = x$;
Step 5: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $x$ near exact fixed point, or message of failure.
```

Key Concepts

Fixed point: x_0 is a fixed point of the function $f(x)$ if $f(x_0) = x_0$.

Fixed point iteration: Calculating the sequence $x_0, x_1 = f(x_0), x_2 = f(x_1), x_3 = f(x_2), \ldots$ given the function f and initial value x_0.

Attractive fixed point: A fixed point is called attractive (or attracting) if there is a neighborhood of the fixed point in which fixed point iteration converges for all initial values in the neighborhood.

Repulsive fixed point: A fixed point is called repulsive (or repelling) if fixed point iteration escapes some neighborhood of the fixed point for any initial value in the neighborhood other than the fixed point itself.

Mean Value Theorem: If f is continuous on $[a, b]$ and has a derivative on (a, b), then there exists $c \in (a, b)$ such that $f'(c) = \frac{f(b)-f(a)}{b-a}$.

Fixed Point Convergence Theorem: Given a function $f(x)$ with continuous first derivative and fixed point $\hat{x}$, if $|f'(\hat{x})| < 1$ then there exists a neighborhood of $\hat{x}$ in which fixed point iteration converges to the fixed point for any initial value in the neighborhood.

Exercises

1. Write an implementation of the fixed point iteration method. Save it as a `.m` file for future use.

2. (i) Decide whether or not the hypotheses of the Mean Value Theorem are met for the function over the interval. (ii) If the hypotheses are met, find a value c as guaranteed by the theorem.

   (a) $f(x) = 3 - x - \sin x$; $[2, 3]$

   (b) $g(x) = 3x^4 - 2x^3 - 3x + 2$; $[0, 1]$

   (c) $g(x) = 3x^4 - 2x^3 - 3x + 2$; $[0, 0.9]$ [S]

   (d) $h(x) = 10 - \cosh(x)$; $[-3, -2]$ [A]

   (e) $f(t) = \sqrt{4 + 5\sin t} - 2.5$; $[-6, -5]$

   (f) $g(t) = \frac{3t^2 \tan t}{1-t^2}$; $[20, 23]$ [S]

   (g) $h(t) = \ln(3\sin t) - \frac{3t}{5}$; $[2, 4]$ [A]

   (h) $f(r) = e^{\sin r} - r$; $[-20, 20]$ [A]

   (i) $g(r) = \sin(e^r) + r$; $[-3, 3]$

   (j) $h(r) = 2^{\sin r} - 3^{\cos r}$; $[1, 3]$

3. Find the fixed point(s) of the function exactly. Use algebra.

   (a) $f(x) = \sqrt[3]{2x^3 - x^2 - x}$

   (b) $f(x) = \frac{\ln(2}{2}$

   (c) $f(x) = \log(x^2 - 3x) - 1 + x$ [A]

   (d) $g(x) = 3x^2 + 5x + 1$ [A]

   (e) $g(x) = x + \frac{5000}{1+2e^{-3t}} - 2500$

   (f) $g(x) = e^{\ln(x+1)-3}$

   (g) $h(x) = \sqrt{4x^2 + 4x + 1}$

   (h) $h(x) = x - 10 + 3^x + 25 \cdot 3^{-x}$ [S]

   (i) $h(x) = x + 6 - 3\log_5(2x)$

4. Find at least two candidate functions, $f_1(x)$ and $f_2(x)$, for finding roots of $g(x)$ via fixed point iteration. In other words, convert the problem of finding a root of g into a problem of finding a fixed point of f_1 or f_2.

   (a) $g(x) = 7x^2 + 5x - 9$

   (b) $g(x) = x + \cos x$

   (c) $g(x) = 6x^5 + 12x^2 - 8$ [A]

   (d) $g(x) = x^2 - e^{3x+4}$ [S]

   (e) $g(x) = 7x - 3\cos(\pi x - 2) + \ln|2x^2 + 4x - 8|$

   (f) $g(x) = 3^{x^2-5x+1} - 2^{-x^2-5x-1}$ [A]

5. Compute the first 5 iterations of the fixed point iteration method using the given function and initial value. Based on these 5 iterations, do you expect the method to converge?

   (a) $f(x) = 3 - \sin x$; $x_0 = 2$

   (b) $g(x) = 10 + x - \cosh(x)$; $x_0 = -3$ [S]

   (c) $h(t) = \ln(3\sin t) + \frac{2t}{5}$; $t_0 = 1$ [A]

   (d) $w(r) = 2^{\sin r} - 3^{\cos r} + r$; $r_0 = 1$

6. Use your function from question 1 with the function and initial value in question 5. Set the tolerance to 10^{-10} and the maximum iterations to 100. Does the method converge within 100 iterations? If so, to what value? Report at least 10 significant digits. [S] [A]

7. Construct a web diagram for each function/initial value pair in question 5. [S] [A]

8. Compare the results from question 6 with the results of question 7. Are they consistent with one another?

9. Use proposition 3 to show that $g(x) = 2x(1-x)$ has a unique fixed point on $[0.3, 0.7]$.

10. Let $f(x) = \frac{3x^2-1}{6x+4}$. [S]

    (a) Show that f has a unique fixed point on $[-4, -0.9]$.

    (b) Use fixed point iteration to find an approximation to the fixed point that is accurate to within 10^{-2}.

11. Let $g(x) = \pi + 0.5\sin(x/2)$.

    (a) Show that g has a unique fixed point on $[0, 2\pi]$.

    (b) Use fixed point iteration to find an approximation to the fixed point that is accurate to within 10^{-2}.

12. Show that the fixed point iteration method applied to $f(x) = \sqrt[3]{8-4x}$ will converge to a root of $g(x) = x^3 + 4x - 8$ for any initial value $x_0 \in [1.2, 1.5]$. [S]

13. Show that fixed point iteration is guaranteed to converge to the fixed point of
    $$f(x) = (\sqrt{2})^x$$
    for any $x_0 \in [1, 3]$. HINT: $f'(x) = \frac{1}{2}\ln(2) \cdot (\sqrt{2})^x$.

14. Let $g(x) = x^2 - 3x - 2$.

    (a) Find a function f on which fixed point iteration will converge to a root of g.

    (b) Use your function to find a root of g to within 10^{-3} of the exact value.

    (c) State the initial value you used and how many iterations it took to get the approximation.

15. Use fixed point iteration with $p_0 = -1$ to approximate a root of $g(x) = x^3 - 3x + 3$ accurate to the nearest 10^{-4}.

16. Use a fixed point iteration method to find an approximation of $\sqrt{3}$ that is accurate to within 10^{-4}. What function and initial value did you use?

17. The function $f(x) = x^4 + 2x^2 - x - 3$ has two roots. One of them is in $[-1, 0]$ and the other is in $[1, 2]$.

    (a) In preparation for finding a root of $f(x)$ using fixed point iteration, one way to manipulate the equation $x^4 + 2x^2 - x - 3 = 0$ is to add x to both sides. This gives
    $$x = x^4 + 2x^2 - 3$$
    Draw appropriate graphs to determine whether iteration of the function $g(x) = x^4 + 2x^2 - 3$ will find the root in $[-1, 0]$. How about the root in $[1, 2]$? Explain how you came to your conclusions.

    (b) Manipulate the equation $x^4 + 2x^2 - x - 3 = 0$ in such a way that fixed point iteration does work to find the root in $[-1, 0]$. Draw the graphs that demonstrate that your method will work.

    (c) Does the same manipulation allow you to find the root in $[1, 2]$? If not, find another manipulation that will. Again, show the graphs that demonstrate that your method will work.

    (d) Use your method(s) from parts 17b and 17c to find the two roots accurate to 3 decimal places.

18. Fixed point iteration on $f(x) = \sqrt[3]{2x^3 - x^2 - x}$ will not converge to a fixed point. However, fixed point iteration on the function $g(x) = \sqrt[3]{x^2 + x}$ will converge to approximately 1.618033988749895 for any x_0 in $[0.5, 3.5]$. [A]

    (a) How many iterations does it take to achieve 10^{-4} accuracy using $g(x)$ with $x_0 = 2.5$?

    (b) Explain why $f(x)$ and $g(x)$ have the same fixed points.

19. Find a zero (any zero) of $g(x) = x^2 + 10\cos x$ accurate to within 10^{-4} using fixed point iteration. State

    (a) the function f to which you fixed point iteration

    (b) the initial value, x_0, you used

   (c) how many iterations it took

20. Let c be a nonzero real number. Argue that any fixed point of $f(x) = xe^{c \cdot g(x)}$ is a root of g.

21. Approximate $\sqrt{3}$ using the method suggested by question 20.

22. Suppose $g(\hat{x}) = 0$ and g has a continuous first derivative. Argue that there exists a value c for which fixed point iteration on $f(x) = x + cg(x)$ will converge to $\hat{x}$ on some neighborhood of $\hat{x}$.

23. Find a value of c for which fixed point iteration is guaranteed to converge for the function $f(x) = x + c(x - 5\cos x)$ with any initial value $x_0 \in [0, \pi/2]$. Explain. [𝔸]

24. Let $g(x) = \frac{1}{2}^x + \frac{1}{5}^x - 10^{-5}$.

    (a) Show that if $g(x)$ has a zero at p, then the function $f(x) = x + cg(x)$ has a fixed point at p.

    (b) Find a value of c for which fixed point iteration of $f(x)$ will successfully converge for any starting value, p_0, in the interval $[16, 17]$. Sketch the graphs that demonstrate that your choice of c is appropriate.

    (c) Use the function from part 24b with the value of c you have determined to find a root of $g(x)$ accurate to within 10^{-4}. State the value you used for p_0. Show the last 3 iterations. How many iterations did it take?

25. Prove that for $f(x) = \cos x$, fixed point iteration converges for any initial value.

26. The Fixed Point Convergence Theorem can be strengthened. The requirement that the first derivative be continuous can be replaced. Modify the proof in the text to show the following claim.

    Given a differentiable function $f(x)$ with fixed point $\hat{x}$, if $|f'(x)| \leq M < 1$ for all x in some neighborhood of $\hat{x}$, then fixed point iteration converges to the fixed point for any initial value in the neighborhood.

27. Create three graphs similar to those in Figure 2.2.4 to analyze the situation when the derivative at the fixed point equals -1. Does the situation differ from that when the derivative at the fixed point equals 1?

Answers

Figure 2.2.4: From left to right: every neighborhood of the fixed point will have both types of initial values; point iteration converges for all values in a neighborhood of the fixed point; fixed point iteration escapes some neighborhood of the fixed point for all initial values in the neighborhood except the fixed point itself

Figure 2.2.6: When its denominator is zero, $f_6(x)$ will be undefined (there is a vertical asymptote in the graph), so we solve $3x^2 - 10x + 4 = 0$ to find two initial values for which fixed point iteration will fail (since the first iteration will be undefined). They are $x = \frac{5 \pm \sqrt{13}}{3} \approx .4648$ and 2.868. To find a third point for which fixed point iteration will fail, we solve the equation $f_6(x) = \frac{5+\sqrt{13}}{5}$ (we could just as easily have solved $f_6(x) = \frac{5-\sqrt{13}}{5}$ instead). Then the second iteration will be undefined since the first iteration will be $\frac{5+\sqrt{13}}{5}$. The only real solution is approximately 1.055909763230534, which can be found by fixed point iteration on $\sqrt[3]{\frac{\sqrt{13}x^2 + 10x^2 - \frac{10\sqrt{13}x}{3} - \frac{50x}{3} + \frac{4\sqrt{13}}{3} + \frac{38}{3}}{2}}$. Prove it. Note, though, the claim that fixed point iteration will fail is based on the assumption of *exact* arithmetic. The fact that any reasonable implementation of the fixed point iteration method will involve floating point arithmetic might provide just enough error for the method to converge even for these initial values.

2.3 Order of Convergence for Fixed Point Iteration

Suppose f is a function with fixed point $\hat{x}$ and $f'(\hat{x})$ exists. Let $x_0, x_1, x_2, \ldots$ be a sequence derived from fixed point iteration ($x_{k+1} = f(x_k)$ for all $k \geq 1$) such that $\lim\limits_{k\to\infty} x_k = \hat{x}$ and $x_k \neq \hat{x}$ for all $k = 0, 1, 2, \ldots$. Then

$$\frac{|x_{n+1} - \hat{x}|}{|x_n - \hat{x}|^1} = \left|\frac{f(x_n) - f(\hat{x})}{x_n - \hat{x}}\right|$$

and

$$\lim_{n\to\infty}\left|\frac{f(x_n) - f(\hat{x})}{x_n - \hat{x}}\right| = |f'(\hat{x})|. \tag{2.3.1}$$

Therefore, fixed point iteration is linearly convergent as long as $f'(\hat{x}) \neq 0$. The following proposition could be presented as a corollary to the Fixed Point Convergence Theorem since much of the argument simply repeats what was noted there, but we choose to present it as a separate claim based on equation 2.3.1. To be more precise, we have the following result.

Proposition 5. *(Fixed Point Error Bound) Let f be a differentiable function with fixed point $\hat{x}$ and let $[a, b]$ be an interval containing $\hat{x}$. If $|f'(x)| \leq M < 1$ for all $x \in [a, b]$ and $f([a, b]) \subseteq [a, b]$, then for any initial value $x_0 \in [a, b]$, fixed point iteration, with $x_{k+1} = f(x_k)$ for all $k \geq 0$, gives an approximation of $\hat{x}$ with absolute error no more than $M^k|x_0 - \hat{x}|$.*

Proof. An elementary induction proof (requested in the exercises) will establish that $x_k \in [a, b]$ for all $k \geq 0$. We proceed to prove the error bound. The absolute error in approximating $\hat{x}$ by x_0 is $|x_0 - \hat{x}| = M^0|x_0 - \hat{x}|$ so the claim is true for $k = 0$. Now suppose the claim is true for some particular but arbitrary $k \geq 0$. By the Mean Value Theorem, there is a c in the interval from $\hat{x}$ to x_k such that $f'(c) = \frac{f(x_k) - f(\hat{x})}{x_k - \hat{x}}$. Since $\hat{x}$ and x_k are both in $[a, b]$, so is c. It follows that $|f'(c)| \leq M$, so $|f(x_k) - f(\hat{x})| \leq M|x_k - \hat{x}|$. But $\hat{x}$ is a fixed point of f, so $f(\hat{x}) = \hat{x}$, from which it follows that $|f(x_k) - \hat{x}| \leq M|x_k - \hat{x}|$, and, therefore, that $|x_{k+1} - \hat{x}| \leq M|x_k - \hat{x}|$. By the inductive hypothesis, $|x_k - \hat{x}| \leq M^k|x_0 - \hat{x}|$, so $|x_{k+1} - \hat{x}| \leq M \cdot M^k|x_0 - \hat{x}| = M^{k+1}|x_0 - \hat{x}|$. □

When $f'(\hat{x}) = 0$, equation 2.3.1 shows that fixed point iteration does not converge linearly. For any sequence $\langle p_n \rangle$ converging to p, if $\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} = 0$ we say the sequence is superlinearly convergent or that convergence is faster than linear.

Consider the functions $f(x) = \frac{1}{8}x^3 - x^2 + 2x + 1$ and $f_1(x) = -x^3 + 5x^2 - 3x - 6$ from section 2.2. Recall 2 is a fixed point of f and 3 is a fixed point of f_1 and observe that $f'(2) = \frac{3}{8} \cdot 2^2 - 2 \cdot 2 + 2 = -\frac{1}{2}$ and $f_1'(3) = -3 \cdot 3^2 + 10 \cdot 3 - 3 = 0$ Consequently, we should expect fixed point iteration of f_1 to converge to 3 faster than that of f converges to 2. With $s_0, s_1, s_2, \ldots = 1.75, f(1.75), f(f(1.75)), \ldots$ and $t_0, t_1, t_2, \ldots = 2.75, f_1(2.75), f_1(f_1(2.75)), \ldots$, table 2.1 shows the relative speeds of convergence. $\langle s_n \rangle$ is converging linearly as expected, and $\langle t_n \rangle$ seems to be converging quadratically. The last four exponents in the $|3 - t_n|$ column are $-4, -8, -14, -27$, indicating that the number of significant digits of accuracy is approximately doubling with each iteration. In other words, the error of one term is roughly the square of the previous error (meaning $\alpha = 2$ in the definition of order of convergence).

Table 2.1: Comparing order of convergence for fixed point iteration when the derivative at the fixed point is not zero (s_n) to that when the derivative at the fixed point is zero (t_n).

n	$\lvert 2 - s_n \rvert$	$\lvert 3 - t_n \rvert$
0	$2.5(10)^{-1}$	$2.5(10)^{-1}$
1	$1.074(10)^{-1}$	$2.343(10)^{-1}$
2	$5.644(10)^{-2}$	$2.068(10)^{-1}$
3	$2.740(10)^{-2}$	$1.623(10)^{-1}$
4	$1.388(10)^{-2}$	$1.010(10)^{-1}$
5	$6.894(10)^{-3}$	$3.984(10)^{-2}$
6	$3.459(10)^{-3}$	$6.286(10)^{-3}$
7	$1.726(10)^{-3}$	$1.578(10)^{-4}$
8	$8.640(10)^{-4}$	$9.966(10)^{-8}$
9	$4.318(10)^{-4}$	$3.973(10)^{-14}$
10	$2.159(10)^{-4}$	$6.317(10)^{-27}$

Table 2.2: Accelerating the convergence of a linearly converging sequence.

n	c_n	a_n	$\lvert c_n - c\rvert$	$\lvert a_n - c\rvert$	$\left\lvert\frac{a_n-c}{c_n-c}\right\rvert$	$\frac{\lvert a_{n+1}-c\rvert}{\lvert a_n-c\rvert^2}$
0	1	.728010	$2.609(10)^{-1}$	$1.107(10)^{-2}$	.0934	.0110
1	.5403	.733665	$1.987(10)^{-1}$	$5.419(10)^{-3}$	.0639	44.19
2	.8575	.736906	$1.184(10)^{-1}$	$2.178(10)^{-3}$	.0400	74.17
3	.6542	.738050	$8.479(10)^{-2}$	$1.034(10)^{-3}$	.0274	217.9
4	.7934	.738636	$5.439(10)^{-2}$	$4.490(10)^{-4}$	.0180	419.4
5	.7103	.738876	$3.771(10)^{-2}$	$2.085(10)^{-4}$	.0122	1034
6	.7639	.738992	$2.487(10)^{-2}$	$9.289(10)^{-5}$	.0081	
7	.7221					
8	.7504					

Taylor's theorem will provide the proof we need that this convergence really is quadratic. Suppose f has a third derivative in a neighborhood of $\hat{x}$. Define $e_n = \hat{x} - x_n$. Then according to Taylor's theorem, $\hat{x} = f(\hat{x}) = f(x_n + e_n) = f(x_n) + e_n f'(x_n) + \frac{1}{2}e_n^2 f''(x_n) + O(e_n^3)$. But $f(x_n) = x_{n+1}$ so we get

$$\hat{x} - x_{n+1} = e_{n+1} = e_n f'(x_n) + \frac{1}{2}e_n^2 f''(x_n) + O(e_n^3). \tag{2.3.2}$$

Also from Taylor's theorem, $f'(\hat{x}) = f'(x_n + e_n) = f'(x_n) + e_n f''(x_n) + O(e_n^2)$. But $f'(\hat{x}) = 0$ so

$$f'(x_n) = -e_n f''(x_n) - O(e_n^2). \tag{2.3.3}$$

Substituting 2.3.3 into 2.3.2,

$$\begin{aligned} e_{n+1} &= e_n(-e_n f''(x_n) - O(e_n^2)) + \frac{1}{2}e_n^2 f''(x_n) + O(e_n^3) \\ &= -\frac{1}{2}e_n^2 f''(x_n) + O(e_n^3). \end{aligned}$$

Hence, $\frac{\hat{x}-x_{n+1}}{(\hat{x}-x_n)^2} = \frac{e_{n+1}}{{e_n}^2} = -\frac{1}{2}f''(x_n) + O(e_n)$ and

$$\lim_{n\to\infty} \frac{|\hat{x} - x_{n+1}|}{|\hat{x} - x_n|^2} = \lim_{n\to\infty} \left|\frac{1}{2}f''(x_n) + O(e_n)\right| = \left|\frac{1}{2}f''(\hat{x})\right|,$$

showing that convergence is at least quadratic. If $f''(\hat{x})$ happens to be 0, then the convergence is superquadratic.

To summarize, on the off-chance that, at a fixed point $\hat{x}$, $f'(\hat{x}) = 0$, fixed point iteration is successful and fast for initial values near $\hat{x}$. But when $f'(\hat{x}) \neq 0$, fixed point iteration may fail to converge to $\hat{x}$, and when it does converge, the convergence is slow. There is a quick fix (quick to implement, not quick to explain) for some of this deficiency when $f'(\hat{x}) \neq 0$, however. We will first concentrate on the speed of convergence.

Let the sequence $\langle c_n \rangle$ be defined by

$$\begin{aligned} c_0 &= 1 \\ c_k &= \cos(c_{k-1}), \quad k > 0. \end{aligned}$$

You should be able to verify that the first few terms of this sequence are (approximately)

$$1, .5403, .8575, .6542, .7934, \ldots$$

This is exactly the sequence you created in the calculator experiment on page 32 of section 2.2. Define a new sequence $\langle a_n \rangle$ by

$$a_n = c_n - \frac{(c_{n+1} - c_n)^2}{c_{n+2} - 2c_{n+1} + c_n}.$$

Table 2.2 shows the first few terms of each sequence along with some error analysis. As promised, the sequence $\langle a_n \rangle$ is converging more quickly than $\langle c_n \rangle$, evidenced by the fact that $\left|\frac{a_n-c}{c_n-c}\right|$ is tending to zero. The last column of the table indicates that the convergence of $\langle a_n \rangle$ to c is not quadratic, however.

More generally, suppose $\langle p_n \rangle$ is any sequence that converges linearly to p. Then we have $\lim_{n\to\infty} \frac{|p-p_{n+1}|}{|p-p_n|} = \lambda \neq 0$, so we should expect $\frac{|p-p_{n+2}|}{|p-p_{n+1}|} \approx \frac{|p-p_{n+1}|}{|p-p_n|} \approx \lambda$ for large enough n, from which we get $|(p-p_{n+2})(p-p_n)| \approx |p-p_{n+1}|^2$. Assuming $p - p_{n+2}$ and $p - p_n$ have the same sign for large n[4], we can remove the absolute values to find

$$\begin{array}{rcl} (p-p_{n+2})(p-p_n) & \approx & (p-p_{n+1})^2 \\ p^2 - (p_{n+2}+p_n)p + p_{n+2}p_n & \approx & p^2 - 2p_{n+1}p + p_{n+1}^2 \\ (-p_{n+2}+2p_{n+1}-p_n)p & \approx & -p_{n+2}p_n + p_{n+1}^2 \\ p & \approx & \dfrac{p_{n+2}p_n - p_{n+1}^2}{p_{n+2}-2p_{n+1}+p_n}. \end{array}$$

Therefore, we may take any three consecutive terms of $\langle p_n \rangle$ and predict p from this formula. For large enough n, this prediction will be a much better estimate of p than is p_n. But just as we were able to claim $|(p-p_{n+2})(p-p_n)| \approx |p-p_{n+1}|^2$, it must also be the case that $p_{n+2}p_n \approx p_{n+1}^2$, so the numerator of our approximation is nearly zero. Of course, that means the denominator must be nearly zero as well, since the quotient is p, a value that may not be zero. To avoid some of the error inherent in this calculation, it is advisable to compute the algebraically equivalent approximation

$$p \approx p_n - \frac{(p_{n+1}-p_n)^2}{p_{n+2}-2p_{n+1}+p_n} \tag{2.3.4}$$

instead. Let's go back and revisit the sequence $\langle s_n \rangle$ and apply this approximation.

Define $a_n = s_n - \frac{(s_{n+1}-s_n)^2}{s_{n+2}-2s_{n+1}+s_n}$ and consider table 2.3 comparing the two sequences $\langle s_n \rangle$ and $\langle a_n \rangle$. $\langle a_n \rangle$ converges significantly faster than the linearly convergent sequence from which is was derived, just as before! The fact that $|2-a_n| \approx |2-s_{n+2}|^2$ is evidence of this claim, but the convergence of $\langle a_n \rangle$ is still linear. Make sure you can calculate the a_n in this table yourself before reading on.

Table 2.3: Comparing fixed point iteration when the derivative at the fixed point is not zero, s_n, to the Aitken's delta-squared sequence, a_n.

n	s_n	$\lvert 2-s_n \rvert$	a_n	$\lvert 2-a_n \rvert$
0	1.75	$2.5(10)^{-1}$	1.99506842493985	$4.931(10)^{-3}$
1	2.107421875	$1.074(10)^{-1}$	1.999022858310434	$9.771(10)^{-4}$
2	1.943559146486223	$5.644(10)^{-2}$	1.999737171760319	$2.628(10)^{-4}$
3	2.027401559734717	$2.740(10)^{-2}$	1.999937151202653	$6.284(10)^{-5}$
4	1.986114080555812	$1.388(10)^{-2}$	1.999983969455146	$1.603(10)^{-5}$
5	2.006894420349172	$6.894(10)^{-3}$		
6	1.996540947531514	$3.459(10)^{-3}$		

On a practical note, there is no sense in calculating all the terms $a_0, a_1, \ldots, a_{n-2}$ as done in the table. The terms of $\langle a_n \rangle$ are dependent only on those of $\langle s_n \rangle$ so a_{n-2} can be calculated just as well without having calculated $a_0, a_1, \ldots, a_{n-3}$. The table shows all of them only for illustrative purposes and so you can get some practice with formula 2.3.4. The important thing to notice is that a_n has approximately twice as many significant digits of accuracy as does s_{n+2}. Consequently, a_0 is a much better approximation than is s_2.

Crumpet 10: Aitken's delta-squared method is designed for any linearly convergent sequence, not just sequences derived from fixed point iteration.

The derivation of 2.3.4, referred to as Aitken's delta-squared formula, makes no reference to fixed point iteration. In fact it makes no assumptions about the origin of the sequence. It makes no difference. It may be a sequence of partial sums, a sequence of partial products, a sequence derived from any recurrence relation, a sequence derived from number theory, or anything else. The only important characteristics are that the sequence converges and it does so linearly.

[4] This will happen in the common events that the $\hat{x} - x_n$ all have the same sign or the $\hat{x} - x_n$ have alternating signs, so this is not an unrealistic assumption.

Table 2.4: Steffensen's method applied to $f(x) = \cos x$.

n	a_n	$g(a_n)$	$g(g(a_n))$	$\lvert a_n - c\rvert$	$\frac{\lvert a_{n+1}-c\rvert}{\lvert a_n-c\rvert^2}$
0	1	.5403023058681398	.8575532158463934	$2.609(10)^{-1}$	.162
1	.7280103614676171	.7464997560452203	.7340702837365296	$1.107(10)^{-2}$	.148
2	.7390669669086738	.7390973701357808	.7390768902228948	$1.816(10)^{-5}$	.148
3	.7390851331660755	.739085133248225	.739085133192888	$4.908(10)^{-11}$	.148
4	.7390851332151607			$3.063(10)^{-17}$	

The sum $\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$ converges to $\frac{\pi}{4}$ linearly so Aitken's delta-squared method should be helpful. If we let $p_n = \sum_{k=1}^{n} \frac{(-1)^{k+1}}{2k-1}$ be the n^{th} partial sum, then $p_2 = \frac{13}{15}$, $p_3 = \frac{76}{105}$, $p_4 = \frac{263}{315}$, and $p_5 = \frac{2578}{3465}$. Aitken's extrapolation gives $a_2 = \frac{13}{15} - \frac{(\frac{76}{105} - \frac{13}{15})^2}{\frac{263}{315} - 2\frac{76}{105} + \frac{13}{15}} = \frac{1321}{1680}$ and $a_3 = \frac{76}{105} - \frac{(\frac{263}{315} - \frac{76}{105})^2}{\frac{2578}{3465} - 2\frac{263}{315} + \frac{76}{105}} = \frac{989}{1260}$. $\frac{|\frac{\pi}{4} - p_4|^2}{|\frac{\pi}{4} - a_2|} \approx 2.6$ and $\frac{|\frac{\pi}{4} - p_5|^2}{|\frac{\pi}{4} - a_3|} \approx 3.5$ so extrapolation gives an error less than the square of the error in the original sequence.

Perhaps this fact gives you an idea. Once s_2 is calculated, we can use equation 2.3.4, also known as Aitken's delta-squared method, to calculate a better approximation than we already have. And once we have this good approximation, it seems a bit silly to cast it aside and continue computing $s_3 = f(s_2), s_4 = f(s_3)$, and so on. What if we use a_0 in place of s_3 in our iteration? In other words, we would have $s_1 = f(s_0), s_2 = f(s_1), s_3 = a_0, s_4 = f(s_3)$, and so on. That should improve s_3, s_4, and s_5. And once we have s_5 we again have three consecutive fixed point iterations, so we can apply Aitken's delta squared method again. Instead of calculating $s_6 = f(s_5)$, we can get what should be a better approximation by using equation 2.3.4 on s_3, s_4, and s_5. In other words, $s_6 = a_3$, $s_7 = f(s_6)$, $s_8 = f(s_7)$. Again, we have three consecutive fixed point iterations, so $s_9 = a_6$, and so on. This gives the sequence

$$\begin{array}{ccc}
1.75, & 2.107421875, & 1.943559146486222, \\
1.995068424939850, & 2.002459692429676, & 1.998768643123618, \\
1.999997974970982, & 2.000001012513483, & 1.999999493743001, \\
1.999999999999658, & 2.000000000000170, & 1.999999999999914, \\
1.999999999999999, & \ldots &
\end{array}$$

which converges to 2 very quickly compared to $\langle s_n \rangle$. If we consider the calculations of $s_1, s_2, s_4, s_5, s_7, s_8, \ldots$ to be intermediary and focus on the subsequence $s_0, s_3, s_6, s_9, \ldots = s_0, a_0, a_3, a_6, \ldots$ as a sequence itself we have

$$1.75,\ 1.995068424939850,\ 1.999997974970982,\ 1.999999999999658,\ 1.999999999999999, \ldots$$

which converges very rapidly! The construction of this subsequence as a sequence in and of itself is called Steffensen's method and the convergence is quadratic as long as $\langle s_n \rangle$ is convergent. The following is a heuristic argument that Steffensen's method gives quadratic convergence. As seen, the error in s_2 is not significantly different from the error in s_0. But a_0 has an error approximately equal to the square of the error in s_2, so the error in a_0 is approximately the square of the error in s_0. Similarly, the error in s_5 is not significantly different from that in $a_0 = s_3$. But the error in a_1 is approximately the square of the error in s_5, so the error in a_1 is approximately the square of the error in a_0. Similarly, the error in a_{n+1} is approximately the square of the error in a_n.

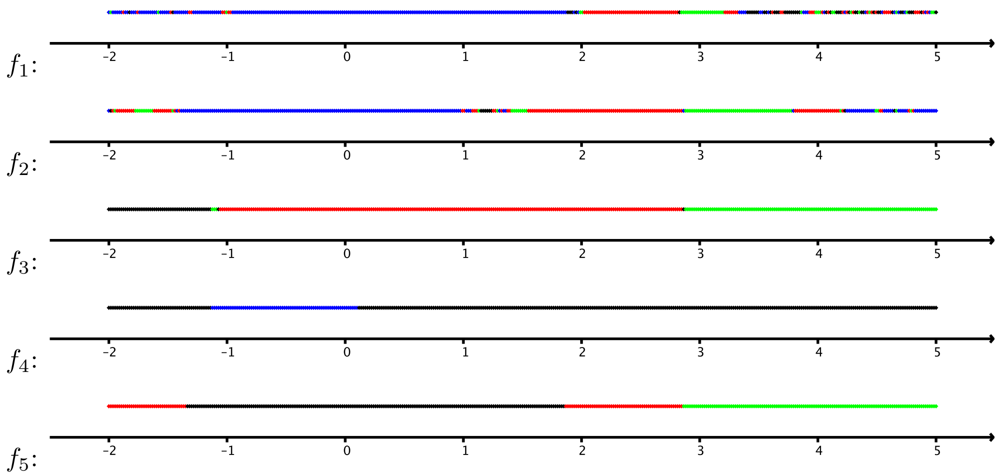
Applying Steffensen's method to the function $f(x) = \cos x$ with $x_0 = 1$, we can accelerate the convergence of the sequence $\langle c_n \rangle$ dramatically. Table 2.4 shows the first few terms of $\langle a_n \rangle$ with some error analysis. The last column of the table indicates that

$$\lim_{n\to\infty} \frac{|a_{n+1} - c|}{|a_n - c|^2} \approx .148$$

and, consequently, that the sequence $\langle a_n \rangle$ converges quadratically.

Finally, we have two ways to get quick convergence from fixed point iteration. One, we simply iterate when the function has derivative zero at the fixed point. Two, we use Steffensen's method.

Figure 2.3.1: Convergence diagrams for 5 functions with the same fixed points—Steffensen's method.



black: does not converge; green: converges to 3; red: converges to $1 + \sqrt{3}$; blue: converges to $1 - \sqrt{3}$

Convergence Diagrams

Speeding up fixed point iteration only takes care of one deficiency of the method. There is still the problem of divergence from fixed points where the derivative of the function has magnitude equal to or greater than 1. Steffensen's method helps. Compare Figure 2.3.1 with Figure 2.2.6. The convergence diagrams for Steffensen's method show convergence over larger intervals of initial values. Moreover, where f_1 and f_2 are concerned, Steffensen's method finds all three fixed points, just as fixed point iteration on f_6 did.

Steffensen's Method (pseudo-code)

Since Steffensen's method is particularly prone to floating-point error, we do a preliminary check for convergence before the Aitken's delta-squared step. This helps prevent large errors or division by zero in Step 4.

```
Assumptions: Fixed point iteration converges to a fixed point of $f$ with initial value $x_0$.
Input: Initial value $x_0$; function $f$; desired accuracy $tol$; maximum number of iterations $N$.
Step 1: For $j = 1 \ldots N$ do Steps 2-6:
    Step 2: Set $x_1 = f(x_0)$; $x_2 = f(x_1)$
    Step 3: If $|x_2 - x_1| \leq tol$ then return $x_2$
    Step 4: Set $x = x_0 - \frac{(x_1 - x_0)^2}{x_2 - 2x_1 + x_0}$
    Step 5: If $|x - x_0| \leq tol$ then return $x$;
    Step 6: Set $x_0 = x$;
Step 7: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $x$ near exact fixed point, or message of failure.
```

Key Concepts

Aitken's delta-squared method: If $\langle p_n \rangle$ converges to p linearly, the sequence $\langle a_n \rangle$ defined by $a_n = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}$ converges to p superlinearly.

Fixed Point Error Bound: Let f be a differentiable function with fixed point $\hat{x}$ and let $[a, b]$ be an interval containing $\hat{x}$. If $|f'(x)| \leq M < 1$ for all $x \in [a, b]$ and $f([a, b]) \subseteq [a, b]$, then for any initial value $x_0 \in [a, b]$, fixed point iteration, with $x_{k+1} = f(x_k)$ for all $k \geq 0$, gives an approximation of $\hat{x}$ with absolute error no more than $M^k |x_0 - \hat{x}|$.

Fixed Point Iteration Order of Convergence: Suppose f is a function with fixed point $\hat{x}$ and $f'(\hat{x})$ exists. Let $x_0, x_1, x_2, \ldots$ be a sequence derived from fixed point iteration ($x_{k+1} = f(x_k)$ for all $k \geq 1$) such that $\lim_{k\to\infty} x_k = \hat{x}$ and $x_k \neq \hat{x}$ for all $k = 0, 1, 2, \ldots$. Then the sequence $\langle x_n \rangle$ converges linearly to $\hat{x}$ if $f'(\hat{x}) \neq 0$ and at least quadratically if $f'(\hat{x}) = 0$.

Steffensen's method: A modification of fixed point iteration where every third term is calculated using Aitken's delta-squared method.

Superlinear convergence: If the sequence $p_0, p_1, p_2, \ldots$ converges to p and $\lim_{k\to\infty} \dfrac{|p_{k+1} - p|}{|p_k - p|} = 0$, then the sequence is said to converge superlinearly.

Superquadratic convergence: If the sequence $p_0, p_1, p_2, \ldots$ converges to p and $\lim_{k\to\infty} \dfrac{|p_{k+1} - p|}{|p_k - p|^2} = 0$, then the sequence is said to converge superquadratically.

Exercises

1. Supply the proof that $x_k \in [a, b]$ for all $k \geq 0$ in proposition 5.

2. Show that
$$\frac{p_{n+2}p_n - p_{n+1}^2}{p_{n+2} - 2p_{n+1} + p_n}$$
and
$$p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}$$
are algebraically equivalent.

3. Write a function that implements Steffensen's method.

4. Write a program (`.m` file) that uses a `while` loop and the `disp()` command to output the first 10 powers of 5 starting with 5^0.

5. Write a program (`.m` file) that uses a `while` loop, an array, and the `disp()` command to find the values of $f(n) = \dfrac{2^{2^n} - 2}{2^{2^n} + 3}$ for $n = 0, 1, 2, 4, 6, 10$. [S]

6. Write a program (`.m` file) that uses a `while` loop, an array, and the `disp()` command to find the values of $f(n) = \dfrac{2n}{\sqrt{n^2 + 3n}}$ for $n = 0, 2, 5, 10, 100, 1000, 20000$.

7. The function $g(x) = \sqrt[3]{5 - 3x}$ satisfies the hypotheses of proposition 5 over the interval $[1, 1.3]$. Find a bound on the number of iterations required to find the fixed point to within 10^{-5} accuracy starting with initial value x_0 of your choice.

8. Fixed point iteration on the function $g(x) = \sqrt[3]{x^2 + x}$ will converge to approximately 1.618033988749895 for any x_0 in $[0.5, 3.5]$. [A]

   (a) Find a bound on the number of iterations it will take to achieve 10^{-4} accuracy with $x_0 = 2.5$.

   (b) How many iterations does it actually take to achieve 10^{-4} accuracy with $x_0 = 2.5$?

9. Let $f(x) = \frac{3x^2 - 1}{6x + 4}$. In exercise 10 of section 2.2, you were asked to show that f has a unique fixed point on $[-4, -0.9]$. [S]

   (a) Find a bound on the number of iterations required to approximate the fixed point to with 10^{-11} accuracy using fixed point iteration with any initial value in $[-4, -0.9]$.

   (b) Use fixed point iteration with $x_0 = -4$ to find an approximation to the fixed point that is accurate to within 10^{-11}. The fixed point is $x = -1$.

   (c) Compare the bound to the actual number of iterations needed.

10. Let $g(x) = \pi + 0.5\sin(x/2)$. In exercise 11 of section 2.2, you were asked to show that g has a unique fixed point on $[0, 2\pi]$.

    (a) Find a bound on the number of iterations required to achieve 10^{-2} accuracy using fixed point iteration with any initial value in $[0, 2\pi]$.

    (b) Use fixed-point iteration with $x_0 = 0$ to find an approximation to the fixed point that is accurate to within 10^{-2}. The fixed point is $x = ???$.

    (c) Compare the bound to the actual number of iterations needed.

11. Calculate two iterations of Steffensen's method for $g(x) = \sqrt[3]{x^2 + x}$ with $x_0 = 2.5$. [A]

12. Use Steffensen's method to find the root of $g(x) = x^4 - 2x^3 - 4x^2 + 4x + 4$ in $[2, 3]$ accurate to five siginificant digits. [A]

13. Compute a_0, a_1, and a_2 of Aitken's delta-squared method for the sequence in problem 2 on page 21. Since the sequence has an undefined term at $n = 1$, start the sequence $\langle \frac{n+1}{n-1} \rangle$ with $n = 2$. In other words, consider the sequence in problem 2 on page 21 to be $3, 2, \frac{5}{3}, \frac{3}{2}, \frac{7}{5} \ldots$ so $p_0 = 3$, $p_1 = 2$, $p_2 = \frac{5}{3}$, and so on.

14. The following sequences are linearly convergent. Generate the first five terms of the sequence $\langle a_n \rangle$ using Aitken's delta-squared calculation.

    (a) $p_0 = 0.5$, $p_n = (2 - e^{p_{n-1}} + p_{n-1}^2)/3$ for $n \geq 1$ [S]

    (b) $p_0 = 0.75$, $p_n = \sqrt{e^{p_{n-1}}/3}$ for $n \geq 1$

15. Use Aitken's delta squared method to find $p = \lim_{n\to\infty} p_n$ accurate to 3 decimal places.
$$\begin{aligned} p_n = \{&-2,\ -1.85271,\ -1.74274,\ -1.66045,\\ &-1.59884,\ -1.55266,\ -1.51804,\\ &-1.49208,\ -1.47261,\ \ldots\} \end{aligned}$$

16. The sequence $\langle a_n \rangle$ of question 13 converges faster than does the sequence in problem 2 on page 21. If you were to apply Aitken's delta-squared method to the sequence $\langle a_n \rangle$, would you expect the convergence to be even faster? Explain. [A]

17. Recall from calculus that $\lim_{n\to\infty} n \sin\left(\frac{1}{n}\right) = 1$. Therefore, if we let $p_n = n \sin\left(\frac{1}{n}\right)$, then the sequence $\langle p_1, p_2, p_3, \ldots \rangle \approx \langle .84147, .95885, .98158, \ldots \rangle$ converges to 1, albeit very slowly. Generate the first three terms of the sequence $\langle a_n \rangle$ using Aitken's delta-squared calculation. Does it seem to be approaching 1 faster than does $\langle p_n \rangle$?

18. Fixed point iteration applied to $f(x) = \sin(x)$ with $x_0 = 1$ takes $29,992$ iterations to reach a number below 0.01 on its way to the fixed point 0. Incidentally, $x_{29992} \approx 0.099999$. How many iterations does it take Steffensen's method with $x_0 = 1$ to reach a number below 0.01? Comment. [S]

19. Let $f(x) = 1 + (\sin x)^2$ and $p_0 = 1$. Find a_1 and a_2 of Steffensen's method with a calculator. [A]

20. Compute the first three iterations of Steffensen's method applied to $g(x) = (\sqrt{2})^x$ using $p_0 = 3$.

21. Steffensen's method is applied to a function $f(x)$ using $p_0 = 1$. If $f(f(p_0)) = 3$ and $a_1 = 0.75$, what is $f(p_0)$? [A]

22. Find the fixed point of $f(x) = x - 0.002(e^x \cos(x) - 100)$ in $[5, 6]$ using Steffensen's method. [A]

23. In question 22 you found a fixed point $\hat{x}$. For what function $g(x)$ is $\hat{x}$ a root?

24. C Write a `while` loop that outputs the numbers $1, .5, .25, .125, .0625, .03125, .015625, \ldots$ until it reaches a number below 10^{-4}.

2.4 Newton's Method

In section 2.3 we addressed some of the deficiency in fixed point iteration, but delayed deep discussion of the mysterious function f_6 of the root finding investigation on page 37. The time has come to discuss f_6 in some detail. We start with some number crunching. Recall that $f_6(x) = \frac{2x^3-5x^2-6}{3x^2-10x+4}$ and let $x_0 = 4$. Proceeding with fixed point iteration,

$$\begin{aligned}
x_1 = f_6(x_0) &= 3.5\\
x_2 = f_6(x_1) &\approx 3.217391304347826\\
x_3 = f_6(x_2) &\approx 3.072749058541597\\
x_4 = f_6(x_3) &\approx 3.013730618589344\\
x_5 = f_6(x_4) &\approx 3.000683798275568\\
x_6 = f_6(x_5) &\approx 3.000001860777997\\
x_7 = f_6(x_6) &\approx 3.000000000013848.
\end{aligned}$$

You can see two things. The sequence $x_0, x_1, x_2, \ldots$

1. is converging to (the fixed point) 3; and
2. it looks like the convergence is quadratic since, starting with x_4 to x_5, the number of significant digits is roughly doubling with each iteration.

In the analysis in section 2.3 on page 42, we found that fixed point iteration converges quadraticly (or better) only when the derivative at the fixed point is zero. These observations should lead you to believe $f_6'(3) = 0$. Let's check. First, the derivative $f_6'(x) = \frac{6x^4-40x^3+74x^2-4x-60}{(3x^2-10x+4)^2}$ (you should verify this). Evaluating the numerator at the fixed point, $x = 3$, we get $6(3)^4 - 40(3)^3 + 74(3)^2 - 4(3) - 60 = 486 - 1080 + 666 - 12 - 60 = 0$. So we have convergence to a fixed point where the derivative of the function is zero, and we indeed have that convergence is quadratic.

Starting with $x_0 = 2$, fixed point iteration on f_6 converges to $1 + \sqrt{3}$, and starting with $x_0 = -1$, fixed point iteration converges to $1 - \sqrt{3}$. You should be able to verify this from the convergence diagram in Figure 2.2.6 or from calculating the first several iterations for each yourself. What you do not get from the convergence diagram is the speed of convergence. For that, you need to look at the iterates. You should do so. Does convergence look quadratic in these cases too? Answer on page 56.

From the convergence diagram, we see that fixed point iteration will converge for virtually any initial value, and all three fixed points can be estimated by fixed point iteration. Moreover, from our calculations, it looks like convergence is quadratic for all three. It's hard to ask for more from a function. Fast convergence to any fixed point! So whence did f_6 come?

Suppose $g(x)$ is differentiable and $g(\hat{x}) = 0$ so g has a root at $\hat{x}$. Consider $f(x) = x - \frac{g(x)}{g'(x)}$. $\hat{x}$ is a fixed point of f as long as $g'(\hat{x}) \neq 0$:

$$f(\hat{x}) = \hat{x} - \frac{g(\hat{x})}{g'(\hat{x})} = \hat{x} - \frac{0}{g'(\hat{x})} = \hat{x}.$$

Moreover, as long as g has a second derivative near $\hat{x}$,

$$\begin{aligned}
f'(\hat{x}) &= 1 - \frac{g'(\hat{x}) \cdot g'(\hat{x}) - g(\hat{x})g''(\hat{x})}{g'(\hat{x}) \cdot g'(\hat{x})}\\
&= 1 - 1 + \frac{0 \cdot g''(\hat{x})}{g'(\hat{x}) \cdot g'(\hat{x})}\\
&= 0.
\end{aligned}$$

From these calculations, we conclude if $g(x)$ is twice differentiable, $g(\hat{x}) = 0$ and $g'(\hat{x}) \neq 0$, then fixed point iteration of $f(x)$ with initial value in a neighborhood of $\hat{x}$ will converge quadratically to $\hat{x}$. What a great way to turn a root finding problem into a fixed point problem!

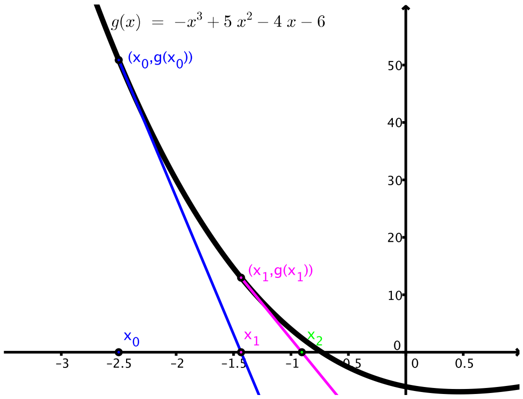
Now is a good time to recall that f_6 was just one of 6 candidate functions designed to find the roots of $g(x) = -x^3 + 5x^2 - 4x - 6$ by fixed point iteration. Indeed, $g'(x) = -3x^2 + 10x - 4$ and

$$\begin{aligned}
x - \frac{g(x)}{g'(x)} &= x - \frac{-x^3 + 5x^2 - 4x - 6}{-3x^2 + 10x - 4}\\
&= \frac{2x^3-5x^2-6}{3x^2-10x+4}\\
&= f_6(x).
\end{aligned}$$

Using fixed point iteration on $f_6(x) = x - \frac{g(x)}{g'(x)}$ to find roots of $g(x)$, as done here, is called Newton's method.

A Geometric Derivation of Newton's Method

The following figure shows how to compute the first two iterations of Newton's method on $g(x) = -x^3+5x^2-4x-6$ with initial value $x_0 = -2.5$ geometrically.



To compute x_1, the tangent line to g at $(x_0, g(x_0))$ is drawn and its intersection with the x-axis is x_1. Similarly, the tangent line to g at $(x_1, g(x_1))$ is drawn and its intersection with the x-axis is x_2. And so on. For example, $(x_0, g(x_0)) = (-2.5, 50.875)$ and $g'(x_0) = g'(-2.5) = -47.75$. Hence, the "rise" $(0-50.875)$ over the "run" $(x_1+2.5)$ between $(-2.5, 50.875)$ and $(x_1, 0)$ must equal -47.75. We thus have $\frac{-50.875}{x_1+2.5} = -47.75$ so

$$x_1 = \frac{-50.875}{-47.75} - 2.5 \approx -1.43455497382199.$$

In symbols, the "rise" $(-g(x_0))$ over the "run" $(x_1 - x_0)$ must equal $g'(x_0)$. In other words,

$$\begin{aligned}
\frac{-g(x_0)}{x_1 - x_0} &= g'(x_0) \Rightarrow \\
\frac{-g(x_0)}{g'(x_0)} &= x_1 - x_0 \Rightarrow \\
x_1 &= x_0 - \frac{g(x_0)}{g'(x_0)}.
\end{aligned}$$

Similar calculation shows $x_2 = x_1 - \frac{g(x_1)}{g'(x_1)}$, and more generally $x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}$. This recurrence relation describes Newton's method—iterating the function $f(x) = x - \frac{g(x)}{g'(x)}$.

Newton's Method (pseudo-code)

Unlike Steffensen's method, the denominator appearing in Newton's method is not expected to approach zero as the iterates converge, so generally there is much less trouble with stability of the calculation and no intermediate checks are done before computing one iteration from the previous.

```
Assumptions: $g$ is twice differentiable. $g$ has a root at $\hat{x}$. $x_0$ is in a neighborhood $(\hat{x}-\delta, \hat{x}+\delta)$ where the
    magnitude of $f'(x) = 1 - \frac{g'(x)\cdot g'(x) - g(x)g''(x)}{g'(x)\cdot g'(x)}$ is less than one.
Input: Initial value $x_0$; function $g$ and its derivative $g'$; desired accuracy $tol$; maximum number of iterations
    $N$.
Step 1: For $j = 1 \ldots N$ do Steps 2-4:
    Step 2: Set $x = x_0 - \frac{g(x_0)}{g'(x_0)}$;
    Step 3: If $|x - x_0| \le tol$ then return $x$;
    Step 4: Set $x_0 = x$;
Step 5: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $x$ near exact fixed point, or message of failure.
```

Table 2.5: The secant method applied to $g(x) = -x^3 + 5x^2 - 4x - 6$ with $x_0 = 5$ and $x_1 = x_0 + g(x_0) = -21$.

n	x_n	$\|3 - x_n\|$
0	5	$2(10)^0$
1	-21	$2.4(10)^1$
2	4.9415730337078	$1.941(10)^0$
3	4.8869924815972	$1.886(10)^0$
4	4.0502898397912	$1.050(10)^0$
5	3.7088949488497	$7.088(10)^{-1}$
6	3.412824115541	$4.128(10)^{-1}$
7	3.232292913133	$2.322(10)^{-1}$
8	3.1141957095727	$1.141(10)^{-1}$
9	3.0465011115969	$4.650(10)^{-2}$
10	3.0132833760752	$1.328(10)^{-2}$
11	3.0020189248976	$2.018(10)^{-3}$
12	3.0001014520965	$1.014(10)^{-4}$
13	3.0000008128334	$8.128(10)^{-7}$
14	3.0000000003297	$3.297(10)^{-10}$

Secant Method

The greatest weakness of Newton's method is the requirement that g' be known and used in the calculation. The derivative is not always accessible or manageable or even known, though. In such a case, it is better to use Steffensen's method or the secant method. The secant method is derived by replacing the g' of Newton's method with a difference quotient. In order for this to make any sense, though, we will need to restate Newton's method in terms of x_n. In Newton's method we are iterating $f(x) = x - \frac{g(x)}{g'(x)}$ so $x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}$.

Now suppose you have a function g and some iterate x_{n-1}. That is enough to locate one point on the graph of g, namely $(x_{n-1}, g(x_{n-1}))$. But we need another point in order to form a difference quotient (the slope of the line through two points). So suppose we have a second value, x_n, near x_{n-1}. Then $\frac{g(x_n)-g(x_{n-1})}{x_n - x_{n-1}} \approx g'(x_n)$ so we can substitute $\frac{g(x_n)-g(x_{n-1})}{x_n - x_{n-1}}$ for $g'(x_n)$ in Newton's method. This yields the secant method, $x_{n+1} = x_n - g(x_n)/\left(\frac{g(x_n)-g(x_{n-1})}{x_n - x_{n-1}}\right)$, which simplifies to

$$x_{n+1} = x_n - g(x_n)\frac{x_n - x_{n-1}}{g(x_n) - g(x_{n-1})}. \tag{2.4.1}$$

Notice this is not quite a fixed point iteration scheme. Each iteration depends on the previous *two* values, not one. The analysis we've done so far does not apply, but there's hope that convergence will be fast since this method is a reasonable approximation of Newton's method near a root, assuming g is differentiable near there. Table 2.5 provides evidence that the secant method indeed converges quickly. In the particular case of $g(x) = -x^3 + 5x^2 - 4x - 6$ with $x_0 = 5$ and $x_1 = x_0 + g(x_0) = -21$, it takes a while to settle in, but after the first 8 iterations or so, convergence is very fast. Not quite quadratic, but superlinear for sure.

Crumpet 11: The secant method converges with order $\frac{1+\sqrt{5}}{2}$.

Suppose g is a function with root $\hat{x}$, $g'(\hat{x}) \neq 0$, $g''(\hat{x}) \neq 0$, and $g'''(x)$ exists in a neighborhood of $\hat{x}$. Let $x_0, x_1, x_2, \ldots$ be a sequence derived from the secant method ($x_{n+1} = x_n - g(x_n)\frac{x_n - x_{n-1}}{g(x_n)-g(x_{n-1})}$ for all $k \geq 2$) such that $\lim\limits_{k\to\infty} x_k = \hat{x}$. Define $e_n = x_n - \hat{x}$ so $x_n = \hat{x} + e_n$. Making this substitution into 2.4.1 we have

$$e_{n+1} = e_n - g(\hat{x} + e_n)\frac{e_n - e_{n-1}}{g(\hat{x} + e_n) - g(\hat{x} + e_{n-1})}. \tag{2.4.2}$$

Taylor's theorem allows $g(\hat{x} + e_k) = g(\hat{x}) + e_k g'(\hat{x}) + \frac{1}{2}e_k^2 g''(\hat{x}) + O(e_k^3)$. Noting that $g(\hat{x}) = 0$ and substituting into 2.4.2,

$$\begin{aligned}
e_{n+1} &= e_n - (e_n - e_{n-1})\frac{e_n g'(\hat{x}) + \frac{1}{2}e_n^2 g''(\hat{x}) + O(e_n^3)}{(e_n - e_{n-1})g'(\hat{x}) + \frac{1}{2}(e_n^2 - e_{n-1}^2)g''(\hat{x}) + O(e_{n-1}^3)} \\
&= e_n - \frac{e_n + \frac{e_n^2 g''(\hat{x})}{2g'(\hat{x})} + O(e_n^3)}{1 + \frac{(e_n+e_{n-1})g''(\hat{x})}{2g'(\hat{x})} + \frac{O(e_{n-1}^3)}{(e_n-e_{n-1})}} \\
&= \frac{e_n\left(1 + \frac{(e_n+e_{n-1})g''(\hat{x})}{2g'(\hat{x})} + \frac{O(e_{n-1}^3)}{(e_n-e_{n-1})}\right) - \left(e_n + \frac{e_n^2 g''(\hat{x})}{2g'(\hat{x})} + O(e_n^3)\right)}{1 + \frac{(e_n+e_{n-1})g''(\hat{x})}{2g'(\hat{x})} + \frac{O(e_{n-1}^3)}{(e_n-e_{n-1})}} \\
&= \frac{e_n e_{n-1}\frac{g''(\hat{x})}{2g'(\hat{x})} + \frac{e_n}{e_n-e_{n-1}}O(e_{n-1}^3) + O(e_n^3)}{1 + \frac{(e_n+e_{n-1})g''(\hat{x})}{2g'(\hat{x})} + \frac{O(e_{n-1}^3)}{(e_n-e_{n-1})}}. \qquad (2.4.3)
\end{aligned}$$

Using equality 2.4.3 to find a value α for which $\lim_{n\to\infty}\frac{|\hat{x}-x_{n+1}|}{|\hat{x}-x_n|^\alpha} = \lambda \neq 0$, we have

$$\begin{aligned}
\lim_{n\to\infty}\frac{|\hat{x}-x_{n+1}|}{|\hat{x}-x_n|^\alpha} &= \lim_{n\to\infty}\frac{|e_{n+1}|}{|e_n|^\alpha} \\
&= \lim_{n\to\infty}\left|\frac{e_n^{1-\alpha}e_{n-1}\frac{g''(\hat{x})}{2g'(\hat{x})} + \frac{e_n^{1-\alpha}}{e_n-e_{n-1}}O(e_{n-1}^3) + O(e_n^{3-\alpha})}{1 + \frac{(e_n+e_{n-1})g''(\hat{x})}{2g'(\hat{x})} + \frac{O(e_{n-1}^3)}{(e_n-e_{n-1})}}\right| \\
&= \lambda \neq 0.
\end{aligned}$$

But $\lim_{n\to\infty} e_n = \lim_{n\to\infty} e_{n-1} = 0$. Hence, $\lim_{n\to\infty} e_n^{1-\alpha}e_{n-1}$ must not be 0 or divergent, for if it were, $\lim_{n\to\infty}\frac{|\hat{x}-x_{n+1}|}{|\hat{x}-x_n|^\alpha}$ would be 0 or divergent, respectively. Consequently, there is a positive constant C such that $\lim_{n\to\infty}|e_n^{1-\alpha}e_{n-1}| = \lim_{n\to\infty}|e_{n+1}^{1-\alpha}e_n| = C \Rightarrow \lim_{n\to\infty}|e_{n+1}e_n^{1/(1-\alpha)}| = C^{1/(1-\alpha)}$. Now we have

$$\lim_{n\to\infty}\frac{|e_{n+1}|}{|e_n|^\alpha} = \lambda \neq 0 \text{ and } \lim_{n\to\infty}\frac{|e_{n+1}|}{|e_n|^{1/(\alpha-1)}} = C^{1/(1-\alpha)} \neq 0.$$

Since the order of convergence of a sequence is unique (Exercise 20 of section 1.3) it must be that $\alpha = 1/(\alpha-1)$ or $\alpha^2 - \alpha - 1 = 0$. The quadratic formula supplies the desired result.

So far we have only applied Newton's method and the secant method to the cubic polynomial $g(x) = -x^3 + 5x^2 - 4x - 6$, a task not strictly necessary. The rational roots theorem, a basic tool from pre-calculus, would give you the roots exactly. The method would have you check $\pm1, \pm2, \pm3$, and ±6 as possible roots of g. Assuming you did your checks by synthetic division, your work might look something like this:

$$\begin{array}{r|rrrr}
3 & -1 & 5 & -4 & -6 \\
 & & -3 & 6 & 6 \\
\hline
 & -1 & 2 & 2 & \boxed{0}
\end{array}$$

meaning $g(x) = (x-3)(-x^2+2x+2)$. The other two roots would then come from the quadratic formula applied to $-x^2+2x+2$ and would be $\frac{-2\pm\sqrt{4+8}}{-2} = 1\pm\sqrt{3}$.

Crumpet 12: Solving the cubic

The solutions of the quadratic equation $ax^2+bx+c=0$ are given by the well-known quadratic equation. Less well-known, and significantly more involved, is any formula for the solutions of the cubic equation $ax^3+bx^2+cx+d=0$. One method of solution follows. First, we let

$$\begin{aligned}
p &= \frac{3ac-b^2}{3a^2} \quad \text{and} \\
q &= \frac{2b^3-9abc+27a^2d}{27a^3}.
\end{aligned}$$

Then we set

$$w^3 = -\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}.$$

Third, we set w_1, w_2, and w_3 to the three possible (complex) values of w. Finally, the three solutions of $ax^3 + bx^2 + cx + d = 0$ are

$$x_i = w_i - \frac{p}{3w_i} - \frac{b}{3a}, \quad i = 1, 2, 3.$$

This is essentially the method of Cardano, published in the 16^{th} century!

For example, to solve the equation $-x^3 + 5x^2 - 4x - 6 = 0$, we start with

$$\begin{aligned} p &= \frac{3(-1)(-4) - 5^2}{3(-1)^2} = -\frac{13}{3} \quad \text{and} \\ q &= \frac{2 \cdot 5^3 - 9(-1)(5)(-4) + 27(-1)^2(-6)}{27(-1)^3} = \frac{92}{27}. \end{aligned}$$

Then

$$\begin{aligned} w^3 &= -\frac{92}{2 \cdot 27} - \sqrt{\frac{92^2}{4 \cdot 27^2} - \frac{13^3}{27^2}} \\ &= -\frac{46}{27} - \frac{\sqrt{92^2 - 4 \cdot 13^3}}{54} \\ &= -\frac{46}{27} - \frac{\sqrt{-324}}{54} \\ &= -\frac{46}{27} - \frac{i}{3}. \end{aligned}$$

In polar form, $w^3 = \frac{13\sqrt{13}}{27} e^{i(\tan^{-1}(9/46) - \pi)}$ so we may set $w_1 = \frac{\sqrt{13}}{3} e^{i(\tan^{-1}(9/46)-\pi)/3}$, one of the cube roots of w^3. Unfortunately, finding the angle $(\tan^{-1}(9/46) - \pi)/3$ exactly amounts to solving a cubic equation! However, with a calculator in hand, one can get the approximation -0.982793723247329, which in the end will be good enough. So, the real part of w_1 is approximately $\frac{\sqrt{13}}{3} \cos(-0.982793723247329) \approx .6666666666666667$ and the imaginary part is approximately $\frac{\sqrt{13}}{3} \sin(-0.982793723247329) \approx -1$. w_1 is suspiciously close to $\frac{2}{3} - i$. And we can check, $\left(\frac{2}{3} - i\right)^3 = \left(\frac{2}{3}\right)^3 + 3\left(\frac{2}{3}\right)^2(-i) + 3 \cdot \frac{2}{3}(-i)^2 + (-i)^3 = \frac{8}{27} - \frac{12}{9}i - 2 + i = -\frac{46}{27} - \frac{1}{3}i$. Therefore, $w_1 = \frac{2}{3} - i$ and we let $w_2 = \left(\frac{2}{3} - i\right)\left(-\frac{1}{2} + \frac{\sqrt{3}}{2}i\right) = \frac{3\sqrt{3}-2}{6} + \frac{3+2\sqrt{3}}{6}i$ and $w_3 = \left(\frac{2}{3} - i\right)\left(-\frac{1}{2} + \frac{\sqrt{3}}{2}i\right) = \frac{-3\sqrt{3}-2}{6} + \frac{3-2\sqrt{3}}{6}i$. Finally,

$$\begin{aligned} x_1 &= w_1 + \frac{13}{9w_1} + \frac{5}{3} = w_1 + \frac{13\overline{w_1}}{9|w_1|^2} + \frac{5}{3} = w_1 + \overline{w_1} + \frac{5}{3} = 3 \\ x_2 &= w_2 + \frac{13}{9w_2} + \frac{5}{3} = w_2 + \frac{13\overline{w_2}}{9|w_2|^2} + \frac{5}{3} = w_2 + \overline{w_2} + \frac{5}{3} = \sqrt{3} + 1 \\ x_3 &= w_3 + \frac{13}{9w_3} + \frac{5}{3} = w_3 + \frac{13\overline{w_3}}{9|w_3|^2} + \frac{5}{3} = w_3 + \overline{w_3} + \frac{5}{3} = -\sqrt{3} + 1 \end{aligned}$$

For an equation you most likely did not see in pre-calculus, or calculus for that matter, consider

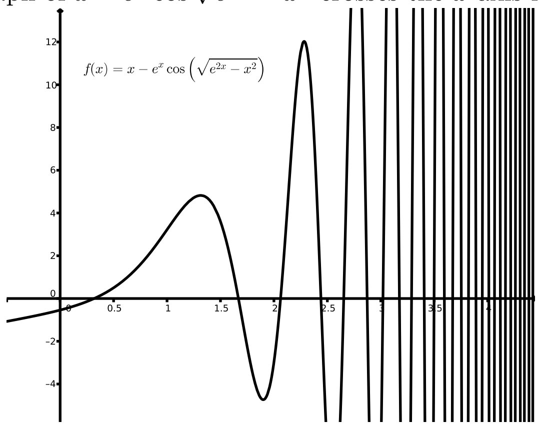
$$x - e^x \cos\sqrt{e^{2x} - x^2} = 0.$$

You might try to solve this equation exactly, with a pencil and paper, but you would soon run into a dead end. This equation can not be solved explicitly. The best you can hope for is to approximate the solutions with a numerical method. To get some idea what we are in for, look at the graph of $x - e^x \cos\sqrt{e^{2x} - x^2}$ in Figure 2.4.1. The function oscillates wildly, and only oscillates more wildly as x increases. The graph crosses the x-axis 29 times on the interval from 0 to 4.5 so has 29 roots there! They are

$$.3181315052047641,\ 1.668024051576096,\ 2.062277729598284,$$
$$2.439940377216816,\ 2.653191974038697, \ldots$$

and can be found by Newton's method with initial values $0, 1.5, 2, 2.4, 2.6, \ldots$. Can you find the next root? Answer on page 56.

Figure 2.4.1: The graph of $x - e^x \cos\sqrt{e^{2x} - x^2}$ crosses the x-axis infinitely many times.



Secant Method (pseudo-code)

A straightforward implementation of the secant method can easily be inefficient due to the number of times g appears in formula on page 51. The pseudo-code below takes great care not to compute each value of g more than once. If it seems more complicated than necessary, this is likely the source of the complication.

```
Assumptions: $g$ has a root at $\hat{x}$. $g$ is differentiable in a neighborhood of $\hat{x}$. $x_0$ and $x_1$ are sufficiently close to $\hat{x}$.
Input: Initial values $x_0$ and $x_1$; function $g$; desired accuracy $tol$; maximum number of iterations $N$.
Step 1: Set $y_0 = g(x_0)$; $y_1 = g(x_1)$
Step 2: For $j = 1 \dots N$ do Steps 3-5:
    Step 3: Set $x = x_1 - y_1 \frac{x_1 - x_0}{y_1 - y_0}$;
    Step 4: If $|x - x_1| \leq tol$ then return $x$;
    Step 5: Set $x_0 = x_1$; $y_0 = y_1$; $x_1 = x$; $y_1 = g(x_1)$
Step 6: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $x$ near exact fixed point, or message of failure.
```

Seeded Secant Method (pseudo-code)

The greatest drawback to the secant method is the necessity of two initial values. They should be near one another, but how near, and how do you determine? These are tough questions, and the answers are complicated at best. One reasonable approach is to let $x_1 = x_0 + g(x_0)$. Assuming x_0 is near a root, $g(x_0)$ will be small, so x_1 will be near x_0. Taking this approach relieves the user from the burden of selecting a second initial value. There are times when such automated selection is not desirable, so both methods have their place. This method only works well when the initial approximation is good.

```
Assumptions: $g$ has a root at $\hat{x}$. $g$ is differentiable in a neighborhood of $\hat{x}$. $x_0$ is sufficiently close to $\hat{x}$.
Input: Initial value $x_0$; function $g$; desired accuracy $tol$; maximum number of iterations $N$.
Step 1: Set $y_0 = g(x_0)$; $x_1 = x_0 + y_0$; $y_1 = g(x_1)$
Step 2: For $j = 1 \dots N$ do Steps 3-5:
    Step 3: Set $x = x_1 - y_1 \frac{x_1 - x_0}{y_1 - y_0}$;
    Step 4: If $|x - x_1| \leq tol$ then return $x$;
    Step 5: Set $x_0 = x_1$; $y_0 = y_1$; $x_1 = x$; $y_1 = g(x_1)$
Step 6: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $x$ near exact fixed point, or message of failure.
```

Key Concepts

Rational Roots Theorem: If the polynomial $p(x) = a_0 + a_1 x + \cdots + a_k x^k$ has rational coefficients, then any rational roots of p are in the set $\left\{\frac{n}{d} : n \text{ is a factor of } a_0 \text{ and } d \text{ is a factor of } a_k\right\}$.

Synthetic division: A method for calculating the quotient of a polynomial by a monomial. Example on page 52.

Newton's method: A root finding method that generally converges to a root of $g(x)$ quadratically, but requires the use of the derivative. In this method, x_0 is chosen and $x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}$ is computed for each $n > 0$.

Secant method: A root finding method that generally converges to a root of $g(x)$ with order approximately 1.618, but does not require the use of the derivative. In this method, x_0 and x_1 are chosen and $x_{n+1} = x_n - g(x_n)\frac{x_n - x_{n-1}}{g(x_n) - g(x_{n-1})}$ is computed for each $n > 0$.

Seeded secant method: A modification of the secant method where x_0 is chosen and $x_1 = x_0 + g(x_0)$.

Exercises

1. Write code that implements Newton's method as a function.

2. Write code that implements the secant method as a function.

3. Write code that implements the seeded secant method as a function.

4. Use your secant method function from question 2 with a tolerance of 10^{-5} to find a solution of

   (a) $e^x + 2^{-x} + 2\cos x - 6 = 0$ using $1 \le x_0 \le 2$.

   (b) $\ln(x-1) + \cos(x-1) = 0$ using $1.3 \le x_0 \le 2$.

   (c) $2x\cos x - (x-2)^2 = 0$ using $2 \le x_0 \le 3$. [A]

   (d) $2x\cos x - (x-2)^2 = 0$ using $3 \le x_0 \le 4$. [A]

   (e) $(x-2)^2 - \ln x = 0$ using $1 \le x_0 \le 2$.

   (f) $(x-2)^2 - \ln x = 0$ using $e \le x_0 \le 4$.

5. Repeat exercise 4 using your Newton's method code from question 1. [A]

6. Repeat exercise 4 using your seeded secant method code from question 3. [A]

7. Repeat exercise 4 using a tolerance of 10^{-10}. Taking this new value as the exact value, did using a tolerance of 10^{-5} give a result accurate to within 10^{-5} of the exact value? [A]

8. Let $g(x) = \frac{100}{x^2}\sin\left(\frac{10}{x}\right)$ and $x_0 = 1.25$. Find x_1 and x_2 of Newton's method. [S]

9. Let $g(x) = 2\ln(1+x^2) - x$. Find x_{14} using Newton's method with

   (a) $x_0 = 5$

   (b) $x_0 = 1.2$ [A]

10. Let $g(x) = 2\ln(1+x^2) - x$. Find x_2 and x_3 using the secant method with

    (a) $x_0 = 5$ and $x_1 = 6$ [S]

    (b) $x_0 = 1$ and $x_1 = 2$

11. Compare the secant method and Newton's method based on questions 4 and 5. Which finds roots in fewer iterations? Which one fails least often? Which is better?

12. Compute the first three iterations of Newton's method applied to $g(x) = x - (\sqrt{2})^x$ with $x_0 = 3$.

13. Find a value of x_0 for which Newton's method will fail to converge to a root of $g(x) = 2 + x - e^x$.

14. Explain why Newton's method fails to converge for the the function $g(x) = x^2 + x + 1$ with $x_0 = 1$.

15. Let $g(x) = \dfrac{2\ln(1+x^2) - x}{1+x^2}$. Using Newton's method to find a root of $g(x)$ with $x_0 = 5$ yields $x_{14} = 8.6624821192$ and with $\tilde{x}_0 = 1.2$ yields $\tilde{x}_{14} = 0$. Compare the values of x_{14} and $\tilde{x}_{14}$ with the fourteenth iterations from question 9 and explain any similarities or differences. [A]

16. Let $g(x) = e^{3x} - 27x^6 + 27x^4e^x - 9x^2e^{2x}$ and let $p_0 = 4$. Find p_{10} using Newton's method. HINT: $g'(x) = 3e^{3x} - 18(x+x^2)e^{2x} + 27(x^4+4x^3)e^x - 162x^5$. [A]

17. Newton's method does not introduce spurious solutions. Suppose $f(x) = x - \frac{g(x)}{g'(x)}$ and $g'(\hat{x}) \neq 0$. Prove that $\hat{x}$ is a root of g if and only if $\hat{x}$ is a fixed point of f. Hint: one direction is proven in the text of this section.

18. The polynomial $g(x) = x^4 + 2x^3 - x - 3$ has a root $\hat{x} \approx 1.097740792$. Find the largest neighborhood (a, b) of $\hat{x}$ such that Newton's method converges to $\hat{x}$ for any initial value $x_0 \in (a, b)$. [S]

19. Use Newton's method to find a negative solution of
    $$0 = 12x^4 - 13x^3 + 7x^2 + x - 130$$
    accurate to the nearest 10^{-4}. What initial value did you use? How many iterations did it take?

20. Consider the function $g(x) = e^{6x} + 3(\ln 2)^2e^{2x} - (\ln 8)e^{4x} - (\ln 2)^3$. Compute enough iterations of Newton's method with $x_0 = 0$ to approximate a zero of g with tolerance 0.0002. Construct the Aitken's delta squared sequence $\langle a_n \rangle$. Is the order of convergence improved? [A]

21. As with Newton's method, the secant method can easily be described geometrically: Draw the line through

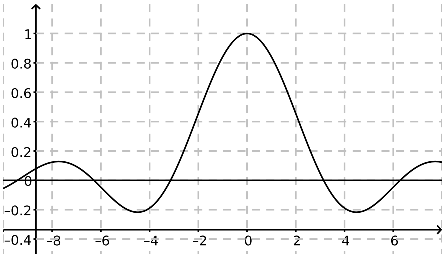
the two points $(x_0, f(x_0))$ and $(x_1, f(x_1))$. Find the intersection of this line with the x-axis. The x-coordinate of the intersection is x_2. Find x_3 by intersecting the line through $(x_1, f(x_1))$ and $(x_2, f(x_2))$ with the x-axis. And so on. Graph the polynomial $p(x) = x^3 - 3x + 3$, and demonstrate the first iteration of the secant method graphically for $x_0 = -1$ and $x_1 = -2$. [S]

22. Suppose you are using the secant method with $x_0 = 1$ and $x_1 = 1.1$ to find a root of $f(x)$.

    (a) Find x_2 given that $f(1) = 0.3$ and $f(1.1) = 0.23$.

    (b) Create a sketch (graph) that illustrates the calculation. HINT: x_2 will be located where the line through $(x_0, f(x_0))$ and $(x_1, f(x_1))$ crosses the x-axis.

23. Use the graph of g to answer the following questions. g has roots at $-2\pi, -\pi, \pi$, and 2π. [A]



    (a) To which root will Newton's method converge if $x_0 = 2.5$?

    (b) What will happen if $x_0 = 0$?

    (c) Find a positive integer value of x_0 for which Newton's method will converge to 2π.

    (d) Find a negative value of x_0 for which Newton's method will converge to 2π.

24. Graph the polynomial $p(x) = x^3 - 3x + 3$, and demonstrate Newton's method graphically for $x_0 = -1$.

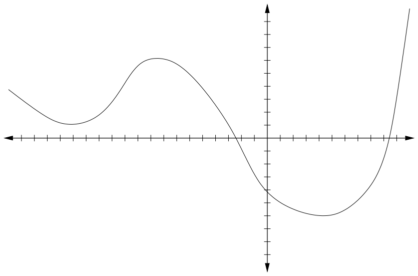
25. Use your code from question 2 to find a root of the function in the interval of question 2 on page 28 to within 10^{-8}. Compare your answer to that from question 4 on page 28. [A]

26. The sum of two numbers is 20. If each number is added to its square root, the product of the two sums is 172.2. Determine the two numbers to within 10^{-4} of their exact values. [S]

27. Find an example of a situation in which Newton's method will fail on the second iteration (i.e., x_1 may be calculated but x_2 may not). [S]

28. Let $h(x) = 2.2x^3 - 6.6x^2 + 4.4x$ and let $g(x) = h^{\circ 3}(x)$. That is, $g(x) = h(h(h(x)))$. Approximate a root of $g'(x)$.

29. For what values of x_0, approximately, will Newton's method converge to -2.5?



30. For the function shown in question 29, find x_2 and x_3 for the secant method with $x_0 = -10$ and $x_1 = 6$.

31. Let
$$f(x) = 10 - \int_0^x \frac{e^t}{1+t}\,dt.$$
Approximate the positive root of f. [A]

32. Of the root finding methods we have surveyed so far (Bisection, Fixed Point, Newton's, Secant, and Steffensen's), which one do you feel is the best? Why?

Answers

Quadratic convergence?

n	x_n	x_n
0	2	-1
1	2.5	$-.7647058823529411$
2	2.666666666666667	$-.7326286052763475$
3	2.722222222222227	$-.7320509933083684$
4	2.731741086881274	$-.7320508075688965$
5	2.732050478023325	
6	2.732050807568503	
$\vdots$	$\vdots$	$\vdots$
	2.732050807568877	$-.7320508075688772$

The convergence looks quadratic since the number of significant digits of accuracy roughly doubles with the last couple of iterations.

Next root? The next root is approximately 2.872257717171606. This can be found using Newton's method with $x_0 = 2.81$, for example. Note this computation is very sensitive to initial conditions because there are so many roots near one another. Starting with $x_0 = 2.8$, for example, leads to the root at 9.662623060421268!

2.5 More Convergence Diagrams

The cubic function $g(x) = 1 - x^3$ has one real root, 1. But it also has two complex roots. If you have studied complex analysis, you probably know what the other two are. And even if you have not studied complex analysis, you can figure them out by basic techniques of pre-calculus. Since 1 is a root, you can use synthetic division to deflate the polynomial:

$$\begin{array}{r|rrrr} 1 & -1 & 0 & 0 & 1 \\ & & -1 & -1 & -1 \\ \hline & -1 & -1 & -1 & \boxed{0} \end{array}$$

This division shows that $g(x) = (x-1)(-x^2 - x - 1)$, so the other two roots are the solutions of the equation $-x^2 - x - 1 = 0$, thus deflating the problem to a quadratic. The solutions are $\frac{1 \pm \sqrt{1-4}}{-2} = -\frac{1}{2} \pm i\frac{\sqrt{3}}{2}$. By the way, you may also recognize $1 - x^3$ as one of the special forms of polynomials, the difference of cubes.

Of course this is all fascinating, but what does this have to do with numerical analysis? What may surprise you is that fixed point iteration (and, therefore, Newton's method), the secant method, and Steffensen's method can all be used to find complex roots just as well as real ones! In fact, the algorithms need no modification! The programming language used to implement the methods, of course, does need to be able to handle complex number arithmetic.

First, finding a root of $g(x) = 1 - x^3$ and finding a fixed point of $f(x) = 1/x^2$ are equivalent. Why? Answer on page 64. Setting $x_0 = -1 + i$ and applying Newton's method and the secant method to $g(x) = 1 - x^3$, and Steffensen's method to $f(x) = 1/x^2$ we get the following:

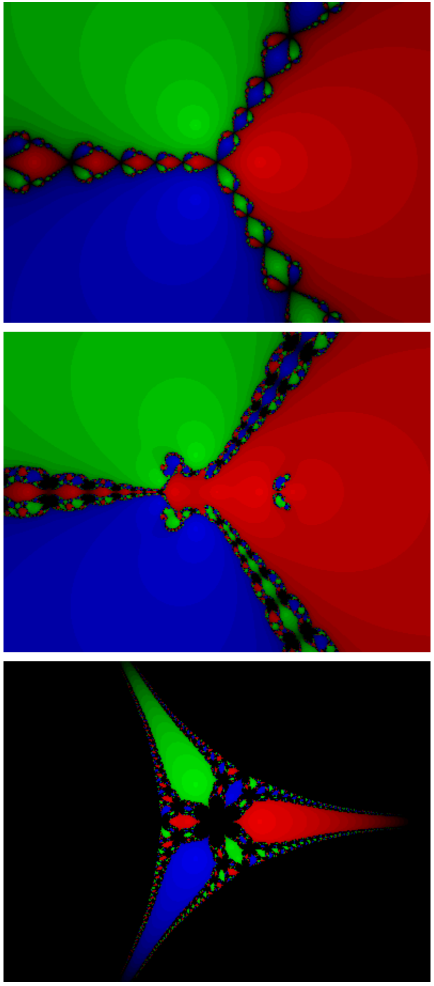
	x_i		
i	Steffensen's	Secant	Newton's
0	$-1+i$	$-1+i$	$-1+i$
1	$-0.85+0.8i$	$-0.66666666+0.83333333i$	$-0.66666666+0.83333333i$
2	$-0.60313824+0.67770639i$	$-0.55034016+0.82376444i$	$-0.50869191+0.84109987i$
3	$-0.39846066+0.84671567i$	$-0.49763752+0.85554014i$	$-0.49932999+0.86626917i$
4	$-0.51660491+0.84998590i$	$-0.49932718+0.86627140i$	$-0.49999991+0.86602490i$
5	$-0.49910537+0.86543351i$	$-0.50000774+0.86602504i$	$-0.50000000+0.86602540i$
6	$-0.50000228+0.86602568i$	$-0.49999999+0.86602540i$	
7	$-0.50000000+0.86602540i$	$-0.50000000+0.86602540i$	
8	$\vdots$	$\vdots$	$\vdots$

Each sequence quickly converges to the complex root $-\frac{1}{2} + \frac{\sqrt{3}}{2}i$. And this is not a fluke or a contrived example. Generally, these methods work just as well in the complex plane as they do on the real line. One can find real roots starting with complex numbers too. If we change the initial value x_0 to $1+i$, Newton's method converges to 1, for example.

Having expanded our view of the methods to include complex numbers, there is a new type of convergence diagram to consider. We can now look at convergence patterns for the three methods over a host of initial values in the complex plane, not just the real line. Figure 2.5.1 shows convergence diagrams for Newton's method with $g(x) = 1 - x^3$, the seeded secant method with $g(x) = 1 - x^3$, and Steffensen's method with $f(x) = 1/x^2$. Each diagram covers the part of the complex plane with real parts in $[-5, 5]$ and imaginary parts in $[-3.75, 3.75]$. The top left corner of each diagram represents initial value $-5 + 3.75i$ and the bottom right corner represents initial value $5 - 3.75i$. The center of each diagram represents the initial value 0. The colors correspond to the three roots, red to 1, green to $-\frac{1}{2} + \frac{\sqrt{3}}{2}i$, and blue to $-\frac{1}{2} - \frac{\sqrt{3}}{2}i$. Black corresponds to failure to converge. The different intensities of red, green, and blue correspond to the number of iterations the method took to converge. The greater the intensity, the fewer iterations. We can see that for $x_0 = 5 - 3.75i$, Newton's method and the seeded secant method both converge to $-\frac{1}{2} + \frac{\sqrt{3}}{2}i$, because the upper right hand corner of each diagram is colored green. Steffensen's method, on the other hand, fails to converge to any root if begun with $x_0 = 5 - 3.75i$, evidenced by the blackness in the upper right hand corner of the convergence diagram.

The dwell represents the maximum number of iterations allowed, so actually the black dots represent initial values for which convergence was not achieved within a number of iterations equal to or less than the dwell. That's different from claiming the method does not converge at all for these initial values. There's a chance that some of the blackened initial values would still lead to convergence if allowed more iterations.

Figure 2.5.1: Convergence diagrams over the complex plane.



From top to bottom:
Newton's method with

$$g(x) = 1 - x^3$$

and dwell 20;
seeded secant method with

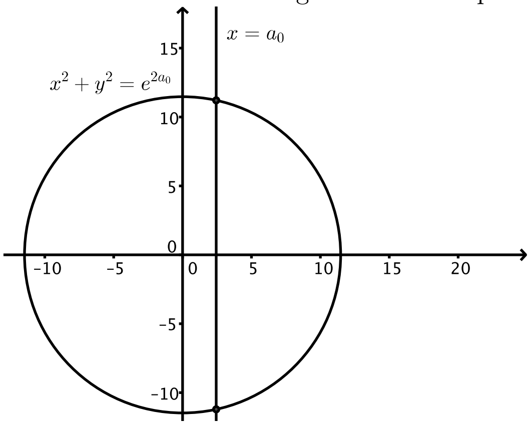
$$g(x) = 1 - x^3$$

and dwell 40;
Steffensen's method with

$$f(x) = \frac{1}{x^2}$$

and dwell 40.
Each diagram covers the part of the complex plane with real parts in $[-5, 5]$ and imaginary parts in $[-3.75, 3.75]$.

Figure 2.5.2: A vertical line and its image under the exponential function.



Two things are very striking about these convergence diagrams. First, the seeded secant method and Newton's method converge for a much larger set of initial values than does Steffensen's method. This is, at least in part, due to the function chosen. For other functions, there may be a fixed point scheme for which Steffensen's method converges on large sets of initial values too. Second, the patterns of colors are extremely intricate, even fractal in nature. Predicting to which root a method will converge for a given initial value, and indeed whether it will converge at all, are very difficult questions! And this analysis is done on a rather benign (simple) function.

Consider now a much more complicated problem—finding the roots of $g(z) = e^z - z$ or, equivalently, finding the fixed points of $f(z) = e^z$. A graph of $f(z)$ (over the real numbers) will quickly convince you that there are no real number solutions. It will take some thought to determine the nature of any complex solutions.

To that end, fix a real number a_0 and consider the vertical line in the complex plane, $L_{a_0} = \{a_0 + ib : b \in \mathbb{R}\}$. The image of L_{a_0} under the exponential function is a circle with radius e^{a_0} centered at the origin. Indeed, $e^{a_0+ib} = e^{a_0}e^{ib} = e^{a_0}(\cos b + i \sin b)$. Thus b parameterizes the circle about the origin with radius e^{a_0}. Now, suppose L_{a_0} contains a fixed point, $\hat{z} = a_0 + i\hat{b}$, of the exponential function, $f(z) = e^z$. Then $\hat{z} = f(\hat{z})$, or $a_0 + i\hat{b} = e^{a_0}(\cos\hat{b} + i\sin\hat{b})$. We conclude that the line and the circle intersect at the fixed point. Every fixed point of f is necessarily an intersection of the line L_{a_0} with the circle C_{a_0} for some a_0. Figure 2.5.2 shows a representative example. In fact, the diagram shows an interesting case: $x = a_0 \approx 2.439940377216816$. The coordinates of the two intersections are

$$(2.439940377216816, \pm 11.2098911414971).$$

The interesting thing is

$$e^{2.439940377216816+11.2098911414971i} \approx 2.439940377216816 - 11.2098911414971i$$

and

$$e^{2.439940377216816-11.2098911414971i} \approx 2.439940377216816 + 11.2098911414971i.$$

The two points are images of one another under the exponential function! What we have found here are called periodic points. If we let $z_1 = 2.439940377216816 - 11.2098911414971i$ and $z_2 = 2.439940377216816 + 11.2098911414971i$, then $e^{z_1} = z_2$ and $e^{z_2} = z1$. Hence, if we iterate $z_2 = f(z_1)$, $z_3 = f(z_2)$, $z_4 = f(z_3)$, $z_5 = f(z_4)$, and so on, the sequence $z_1, z_2, z_3, z_4, \ldots$ actually looks like

$$z_1, z_2, z_1, z_2, z_1, z_2, \ldots.$$

The sequence just flops back and forth between z_1 and z_2 in a periodic fashion. We call such values period 2 points. They are not fixed points of $f(z)$ but they are fixed points of $f(f(z))$!

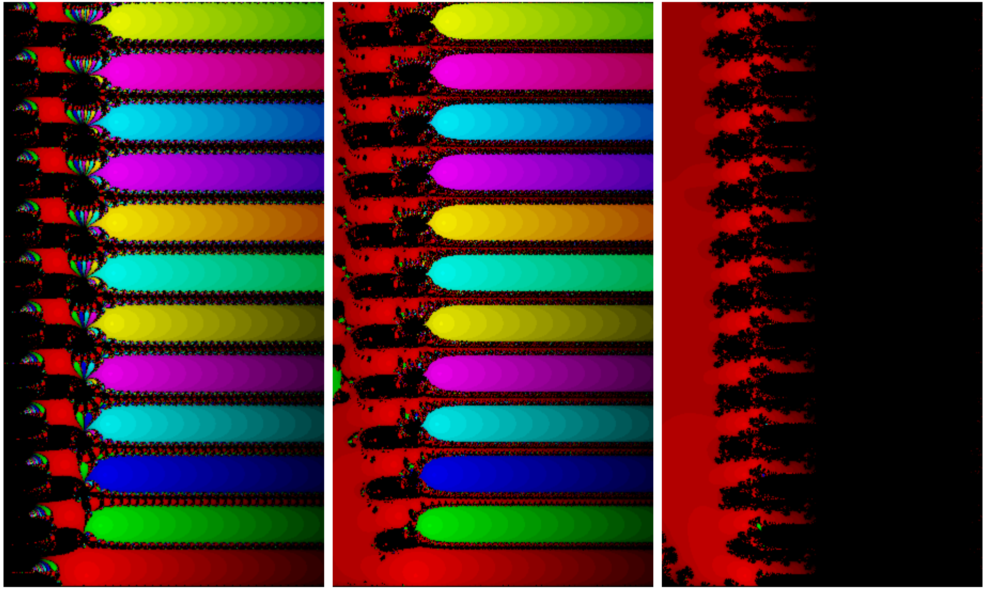
Crumpet 13: Periodic points.

If a sequence $\langle p_n \rangle$ has the form

$$p_1, p_2, \ldots, p_k, p_1, p_2, \ldots, p_k, p_1, \ldots, \qquad k > 1$$

then we say p_1 is a period k point (and $p_2, p_3, \ldots, p_k$ are too!).

Figure 2.5.3: More convergence diagrams over the complex plane.



From left to right: Newton's method with $g(z) = z - e^z$ and dwell 20; secant method with $g(z) = z - e^z$ and dwell 40; Steffensen's method with $f(z) = e^z$ and dwell 40. Each diagram covers the part of the complex plane with real parts in $[-10, 30]$ and imaginary parts in $[0, 73]$.

On the other hand, $\hat{z} = 2.062277729598284 + 7.588631178472513i$ is (approximately) a fixed point of $f(z)$ since

$$e^{2.062277729598284+7.588631178472513i} = 2.062277729598284 + 7.588631178472513i.$$

Moreover, the conjugate of $\hat{z}$, $\overline{\hat{z}} = 2.0622377729598284 - 7.588631178472513i$ is also a fixed point. Verify it with a calculator or a computer!

Generally, if $\hat{z}$ is a fixed point of e^z then so is $\overline{\hat{z}}$:

$$\hat{z} = e^{\hat{z}} \implies \overline{\hat{z}} = \overline{e^{\hat{z}}} = e^{\overline{\hat{z}}}.$$

So if we find one fixed point, we actually have found two, the fixed point and its conjugate.

We're ready to get back to considering intersections of L_{a_0} and C_{a_0}. Assume $a_0 + ib$ is a fixed point of e^z. Then $a_0 + ib = e^{a_0+ib} = e^{a_0}(\cos b + i \sin b)$, so

$$\begin{aligned} a_0 &= e^{a_0} \cos b \\ b &= e^{a_0} \sin b \end{aligned} \tag{2.5.1}$$

Now, because $a_0 + ib$ is a point of intersection, it is on C_{a_0}, so $a_0^2 + b^2 = e^{2a_0} \Rightarrow b = \pm\sqrt{e^{2a_0} - a_0^2}$. Finally, substituting $b = \sqrt{e^{2a_0} - a_0^2}$ into 2.5.1, we find an intersection point will be a fixed point if and only if

$$\begin{aligned} a_0 &= e^{a_0} \cos \sqrt{e^{2a_0} - a_0^2} \\ &\text{and} \\ \sqrt{e^{2a_0} - a_0^2} &= e^{a_0} \sin \sqrt{e^{2a_0} - a_0^2}. \end{aligned} \tag{2.5.2}$$

You should pause long enough to consider why it is not necessary to substitute $b = -\sqrt{e^{2a_0} - a_0^2}$ into 2.5.1. Hint: make the substitution and simplify. You should find out that the two equations you get are equivalent to those in 2.5.1.

For example, $2.439940377216816 - 11.2098911414971i$ and $2.062277729598284 + 7.588631178472513i$ both satisfy the first equation of 2.5.2, but $2.439940377216816 - 11.2098911414971i$ does not satisfy the second while $2.062277729598284 + 7.588631178472513i$ does. So, as observed earlier, $2.439940377216816 - 11.2098911414971i$ is not a fixed point but $2.062277729598284 + 7.588631178472513i$ is.

Do you recognize the first equation of 2.5.2? We first saw it on page 53 in section 2.4. As noted there, the smallest five solutions are

$$.3181315052047641,\ 1.668024051576096,\ 2.062277729598284,$$
$$2.439940377216816,\ 2.653191974038697, \ldots$$

The values 2.062277729598284 and 2.439940377216816 provided the examples for this discussion. What about the other three values in this list? Do they give fixed points of the exponential function? Period two points? Something else? Take a moment to investigate. Answers are on page 64. Using the computer to investigate 2.062277729598284, which we know is a fixed point:

```
> a0=2.062277729598284
> b=sqrt(exp(2*a0)-a0^2)
> exp(a0+I*b)
ans =  2.06227772959828 + 7.58863117847251i
```

verifies that $e^{a_0+ib} = a_0 + ib$ for $a_0 = 2.062277729598284$, at least to machine precision. The exact value of the fixed point is not known, but that is the nature of numerical analysis.

Figure 2.5.3 shows convergence to 12 of the fixed points of e^z, one for each of the 12 different colors. The coordinates of each fixed point can be approximated by locating the spot of greatest intensity within each colored band.

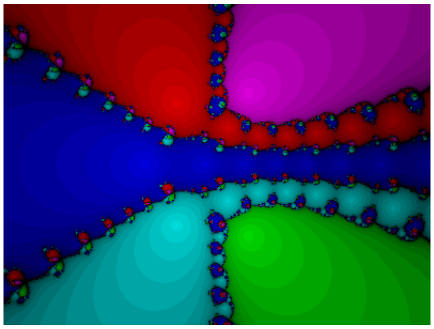
As was done in Figure 2.5.3, convergence diagrams for the secant method can be created by setting $x_1 = x_0 + \delta$ for some small number δ. It does not matter whether δ is real or complex. Selecting x_1 automatically this way allows the diagram to show convergence or divergence based on x_0 alone, just as is done for the other convergence diagrams. You will notice that the convergence diagram for the secant method and the convergence diagram for Newton's method are quite similar. For sufficiently small δ, this will be the case in general. The secant method convergence diagram and the Newton's method convergence diagram for the same function over the same region will look very much the same. The only significant difference will be the number of iterations needed for convergence. The secant method will need more iterations to converge.

Exercises

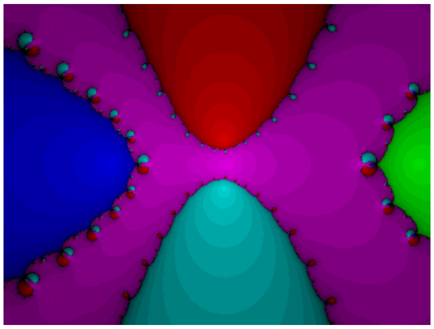
1. Match the function with its Newton's method convergence diagram. The real axis passes through the center of each diagram, and the imaginary axis is represented, but is not necessarily centered. [S]

$$\begin{aligned}
f(x) &= 56 - 152x + 140x^2 - 17x^3 - 48x^4 + 9x^5 \\
g(x) &= (x^2)(\ln x) + (x-3)e^x \\
h(x) &= 1 + 2x + 3x^2 + 4x^3 + 5x^4 + 6x^5 \\
l(x) &= (\ln x)(x^3 + 1)
\end{aligned}$$

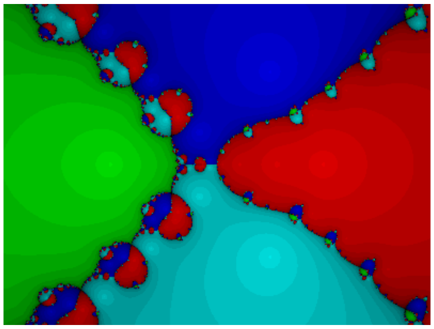
(a)



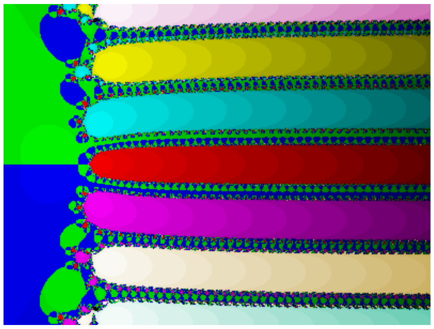
(b)



(c)



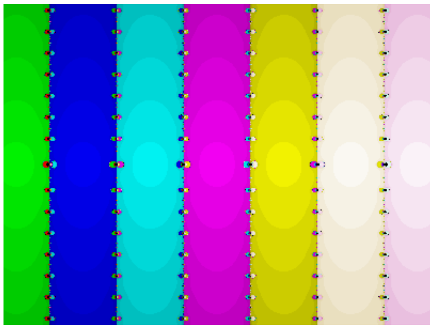
(d)



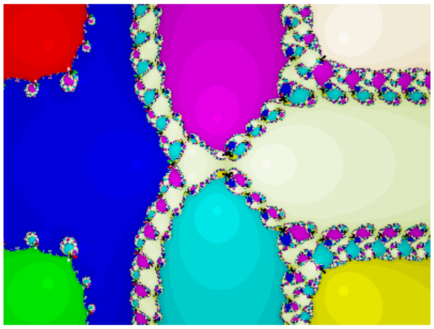
2. Match the function with its Newton's method convergence diagram. The real axis passes through the center of each diagram, and the imaginary axis is represented, but is not necessarily centered. [A]

$$\begin{aligned} f(x) &= \sin x \\ g(x) &= \sin x - e^{-x} \\ h(x) &= e^{x} + 2^{-x} + 2\cos x - 6 \\ l(x) &= x^{4} + 2x^{2} + 4 \end{aligned}$$

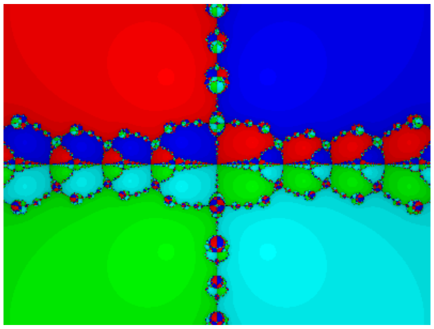
(a)



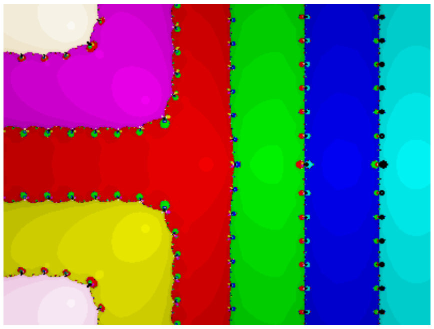
(b)



(c)



(d)



3. Find a polynomial that has the following roots and no others.

(a) $-7, 2, 1 \pm 5i$

(b) $-7, 2, 1 + 5i$

(c) $-4, -1, 2, \pm 2i$ [S]

(d) $-4, -1, 2, 2i$ [S]

(e) $0, -1 \pm i, 1 \pm i$

(f) $-3 + i, -2 - i, -3i, 1 - 2i$

4. Create Newton's method convergence diagrams for the polynomials of question 3. Make sure you capture a region that shows at least a small area converging to each root.

5. The functions $f(x) = e^x$ and $g(x) = \frac{1}{x^2+1}$ have no roots, real or complex. Find at least two others that also have no roots.

6. Let $f(x) = \frac{x^2-7x+10}{2} + \sin(3x)$.

   (a) Find all the real roots of f. This is not a polynomial, so deflation will not work. Instead, graph the function and use Newton's method to find the real roots accurate to 10^{-8}. There are four of them.

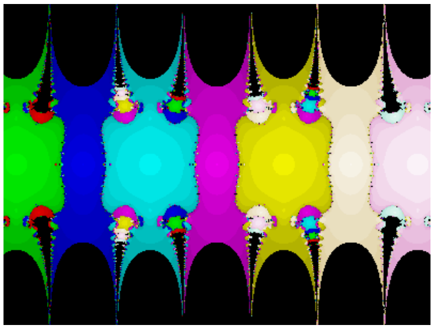
   (b) Create a Newton's method convergence diagram for f to see if there are any complex roots. If so, use Newton's method to approximate them. Use the convergence diagram to help you choose initial values.

   (c) Can you find all the roots of f?

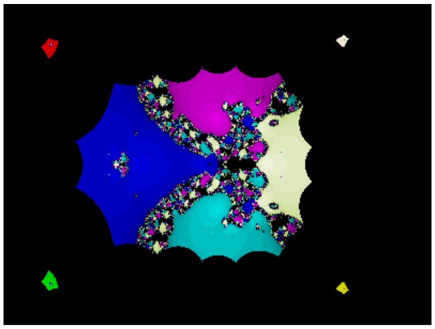
7. Match the function with its seeded secant method convergence diagram. The real axis passes through the center of each diagram, and the imaginary axis is represented, but is not necessarily centered. [S]

$$\begin{aligned} f(x) &= \sin x \\ g(x) &= \sin x - e^{-x} \\ h(x) &= e^x + 2^{-x} + 2\cos x - 6 \\ l(x) &= 56 - 152x + 140x^2 - 17x^3 - 48x^4 + 9x^5 \end{aligned}$$

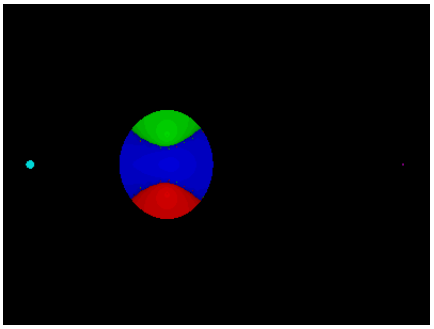
(a)



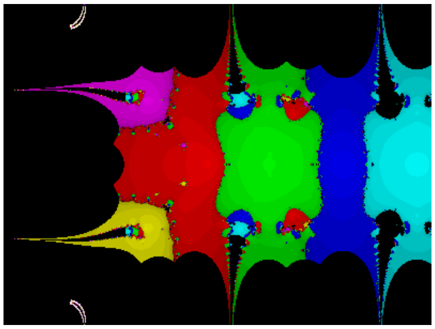
(b)



(c)

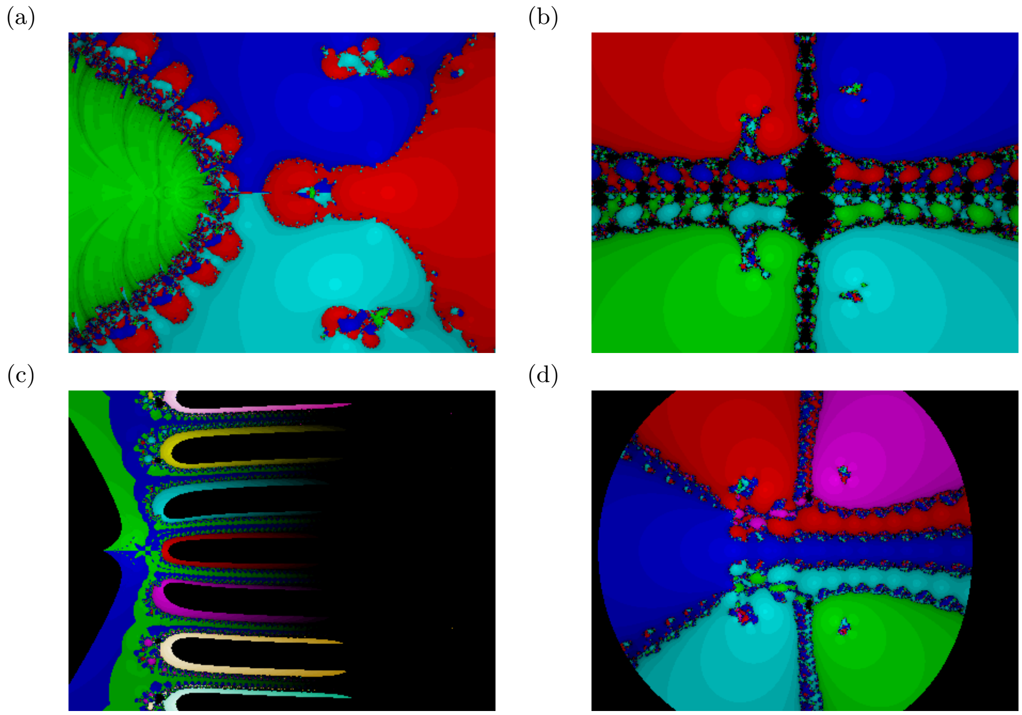


(d)



8. Match the function with its seeded secant method convergence diagram. The real axis passes through the center of each diagram, and the imaginary axis is represented, but is not necessarily centered. [A]

$$\begin{aligned} f(x) &= x^4 + 2x^2 + 4 \\ g(x) &= (x^2)(\ln x) + (x-3)e^x \\ h(x) &= 1 + 2x + 3x^2 + 4x^3 + 5x^4 + 6x^5 \\ l(x) &= (\ln x)(x^3 + 1) \end{aligned}$$



9. Create seeded secant method convergence diagrams for the polynomials of question 3. Make sure you capture a region that shows at least a small area converging to each root.

10. The Newton's method convergence diagram for one polynomial is much like the Newton's method convergence diagram for another. Interesting changes in the Newton's method convergence diagrams and seeded secant method convergence diagrams can be achieved by multiplying a polynomial by a non-polynomial function with no roots. Create Newton's method and seeded secant method convergence diagrams for products of functions in question 3 with functions in question 5.

11. Discuss the relative strengths and weaknesses of Newton's method, the secant method, and the seeded secant method.

Answers

Why equivalent? The equations $g(x) = 0$ and $f(x) = x$ have exactly the same solutions. $g(x) = 0 \Leftrightarrow 1 - x^3 = 0 \Leftrightarrow 1 = x^3 \Leftrightarrow \frac{1}{x^2} = x \Leftrightarrow f(x) = x$.

Nature of roots? .3181315052047641 is a fixed point of the exponential function:

```
> a0=.3181315052047641;
> b=sqrt(exp(2*a0)-a0^2)
> exp(a0+I*b)
ans =  0.318131505204764 + 1.337235701430689i
```

1.668024051576096 is a period two point of the exponential function:

```
> a0=1.668024051576096;
> b=sqrt(exp(2*a0)-a0^2)
> exp(a0+I*b)
ans =  1.66802405157609 - 5.03244706448616i
```

2.653191974038697 is a fixed point of the exponential function:

```
> a0=2.653191974038697;
> b=sqrt(exp(2*a0)-a0^2)
> exp(a0+I*b)
ans =   2.65319197403878 + 13.949208334533191
```

2.6 Roots of Polynomials

Synthetic division revisited

You may recall using the rational roots theorem and synthetic division to find roots of polynomials of degree 3 or more in algebra. The process was something like this. You made a list of possible roots based on the rational roots theorem. You checked each one using synthetic division until you either found a root or ran out of candidates. It is possible that was as far as your class took the process, but there is more to say.

Suppose we have a polynomial $p(x)$ and a number t. Synthetic division gives coefficients of $q(x)$ such that $p(x) = q(x) \cdot (x - t) + p(t)$. For example, the synthetic division

$$\begin{array}{r|rrrr}
\multicolumn{1}{c}{\overbrace{}^{t}} & \multicolumn{4}{c}{\overbrace{}^{p(x)}}\\
-3 & -4 & 2 & 3 & -6\\
 & & 12 & -42 & 117\\ \hline
 & -4 & 14 & -39 & \boxed{111}\\
\multicolumn{1}{c}{} & \multicolumn{3}{c}{\underbrace{}_{q(x)}} & \multicolumn{1}{c}{\underbrace{}_{p(t)}}
\end{array}$$

tells us that $p(x) = -4x^3 + 2x^2 + 3x - 6 = (-4x^2 + 14x - 39)(x + 3) + 111$. While it is a small burden to evaluate the expression $-4x^3 + 2x^2 + 3x - 6$ when $x = -3$, it is no burden at all to evaluate $(-4x^2 + 14x - 39)(x + 3) + 111$ when $x = -3$. The $(x + 3)$ factor is zero, so it doesn't matter to what $(-4x^2 + 14x - 39)$ evaluates. The product is zero and $(-4x^2 + 14x - 39)(x + 3) + 111$ evaluates to 111. Therefore, $p(-3) = 111$. Synthetic division gives a quick way to evaluate a polynomial. The number at the end of the division is the value of the polynomial at the value of the divisor.

More generally, here is a dissection of the division of $p(x) = a_0 + a_1x + \cdots + a_nx^n$ by $x - t$ using synthetic division:

$$\begin{array}{c|ccccc}
t & a_n & a_{n-1} & a_{n-2} & \cdots & a_0\\
 & & a_nt & a_n(a_nt + a_{n-1}) & \cdots & a_n(\cdots a_n(a_n(a_nt + a_{n-1}) + a_{n-2}) + \cdots + a_1)\\ \hline
 & a_n & a_nt + a_{n-1} & a_n(a_nt + a_{n-1}) + a_{n-2} & \cdots & \boxed{p(t)}
\end{array}$$

Beginning with t in the upper left corner, we end up with $p(t)$ in the lower right corner. It is not only when the number in the lower right corner is zero do we find something of interest. Every synthetic division gives something of interest! The number in the bottom right corner is $p(t)$ whether it turns out to be zero or not. And there is more.

The numbers a_n, $a_nt + a_{n-1}$, $a_n(a_nt + a_{n-1}) + a_{n-2}$, and so on, appearing in the bottom row of the synthetic division give the coefficients of the quotient, $q(x)$. Every synthetic division gives a decomposition of the polynomial into quotient and remainder. Thus, with every synthetic division, we get an equivalent expression of the form $q(x) \cdot (x - t) + p(t)$. There is still more.

Differentiating the equation $p(x) = q(x) \cdot (x - t) + p(t)$ with respect to x gives

$$p'(x) = q'(x) \cdot (x - t) + q(x).$$

Hence, $p'(t) = q'(t) \cdot (t - t) + q(t) = q(t)$. So, not only do the numbers in the bottom row give the coefficients of the quotient, they double as coefficients appropriate for evaluating $p'(t)$. Returning to the previous example, if we desire to calculate $p'(-3)$, we simply continue the synthetic division as in

$$\begin{array}{r|rrrr}
-3 & -4 & 2 & 3 & -6\\
 & & 12 & -42 & 117\\ \hline
-3 & -4 & 14 & -39 & \boxed{111}\\
 & & 12 & -78 & \\ \hline
 & -4 & 26 & \boxed{-117} &
\end{array}$$

and find out $p'(-3) = -117$. The procedure of calculating $p(t)$ and $p'(t)$ by simultaneous synthetic divisions is known as Horner's method and is especially convenient for use in Newton's method. If we were trying to find a root of $p(x) = -4x^3 + 2x^2 + 3x - 6$ with initial approximation $x_0 = -3$ we would have, at this point, $x_1 = x_0 - \frac{p(x_0)}{p'(x_0)} = -3 - \frac{111}{-117} \approx -2.05128$. Yet there is more.

Finding all the roots of polynomials

When we happen upon a root of the polynomial $p(x)$, the result of the synthetic division, $p(x) = q(x)(x-t)+p(t)$, reduces to $p(x) = q(x)(x-t)$ since t is a root, meaning $p(t) = 0$. In this case, we have a factorization of $p(x)$. The rest of the roots of p are exactly the roots of q, so having found one root, we have reduced the problem of finding roots of p to (a) noting the root we have found plus (b) finding the roots of the polynomial q, a polynomial of one degree less than that of p. In this way, we have deflated the problem of finding the n roots of the n^{th} degree polynomial p to finding the $n-1$ roots of the $(n-1)$-degree polynomial q. Taking it a step further, when we have found a root of q, we can use synthetic division to reduce the problem again. We (a) note the root of q and (b) continue searching for roots of the quotient, an $(n-2)$-degree polynomial. We continue this way, deflating the problem by one degree each time we find a root until we have reduced the problem to a 2^{nd} degree polynomial. At this point, we have a quadratic polynomial and can use the quadratic equation to find the last two roots.

For example, -1.18985 is (approximately) a root of $p(x) = -4x^3+2x^2+3x-6$. Synthetic division of $p(x)$ by $(x+1.18985)$ gives

$$\begin{array}{r|rrrr} -1.18985 & -4 & 2 & 3 & -6 \\ \hline & & 4.7594 & -8.04267 & 6.00002 \\ \hline & -4 & 6.7594 & -5.04267 & \boxed{0.00002} \end{array}$$

The (near) zero in the box at the bottom-right indicates that -1.18985 is approximately a root. There is no appreciable remainder upon division of $-4x^3+2x^2+3x-6$ by $x+1.18985$. Moreover, the numbers $-4, 6.7594, -5.04267$ in the bottom row give the coefficients of $q(x)$. Thus, we find from this division that $-4x^3+2x^2+3x-6 =\approx (-4x^2+6.7594x-5.04267)(x+118985)$. We can now find the other two roots by locating the roots of $q(x) = -4x^2+6.7594x-5.04267$. Using the quadratic formula, they are

$$\frac{-6.7594 \pm \sqrt{6.7594^2 - 4(-4)(-5.04267)}}{-8} \approx .84493 \pm .73944i.$$

Our process will lead us to finding n roots of any n^{th} degree polynomial. It is important to note that some of these roots may be complex and some of them may be repeated.

Crumpet 14: The Fundamental Theorem of Algebra

The process of finding one root of a given polynomial, deflating, and finding another mirrors quite closely the mathematical theorems of algebra. The Fundamental Theorem of Algebra states that every polynomial with complex coefficients and degree at least one has a complex root. Thus our search for a root is not in vain! We can then write our polynomial in factored form and continue. The Fundamental Theorem says that there is again a root of the deflated polynomial. And if we keep track of all the roots as we find them, we end up writing our polynomial in the form

$$p(x) = a(x-r_1)^{e_1}(x-r_2)^{e_2}\cdots(x-r_k)^{e_k}, \tag{2.6.1}$$

where a is a nonzero constant, $r_1, r_2, \ldots, r_k$ are the k distinct complex roots, and $e_1, e_2, \ldots, e_k$ are the so-called (positive integer) multiplicities of the roots. From this form, we see that the degree of the polynomial equals the sum of the multiplicities, $e_1+e_2+\cdots+e_k$. This is what we mean when we say the number of roots, counting multiplicity, is equal to the degree of the polynomial. Thus when searching for the roots of a polynomial of degree n, we know we are looking for n roots, but not necessarily n distinct roots. Some of them may be repeated and the repetitions are accounted for in the multiplicities. To formalize the claim in equation 2.6.1, we have the follwing theorem.

Theorem 6. *(Fundamental Factorization Theorem) If $n \geq 1$ and p is a degree n polynomial, then*

$$p(x) = a(x-r_1)^{e_1}(x-r_2)^{e_2}\cdots(x-r_k)^{e_k}$$

for some constant $a \neq 0$, roots $r_1, r_2, \ldots, r_k$, and positive integer exponents $e_1, e_2, \ldots, e_k$ where

$$\sum_{j=1}^{k} e_j = n.$$

Proof. Suppose $n = 1$ so $p(x)$ takes the form $ax + b$ with $a \neq 0$. Then $p(x) = a(x - (-\frac{b}{a}))^1$ and thus takes the required form. Now suppose all polynomials of some degree $n \geq 1$ take the required form and let p be a polynomial of degree $n + 1$. By the Fundamental Theorem of Algebra, p has a root. Call it ρ. Then $x - \rho$ is a factor of p so p can be written as $p(x) = (x - \rho) \cdot q(x)$ for some polynomial q of degree n. By the inductive hypothesis, we have that q takes the required form, so

$$p(x) = (x - \rho) \cdot a(x - r_1)^{e_1}(x - r_2)^{e_2} \cdots (x - r_k)^{e_k}$$

where $e_1 + e_2 + \cdots + e_k = n$. If ρ is distinct from $r_1, r_2, \ldots, r_k$, then p takes the form

$$p(x) = a(x - r_1)^{e_1}(x - r_2)^{e_2} \cdots (x - r_k)^{e_k}(x - \rho)^1.$$

If ρ equals one of $r_1, r_2, \ldots, r_k$, say r_j, then p takes the form

$$p(x) = a(x - r_1)^{e_1}(x - r_2)^{e_2} \cdots (x - r_j)^{e_j+1} \cdots (x - r_k)^{e_k}.$$

In either case, p takes the required form and the proof is complete. □

Pseudo-pseudo-code for this procedure might look something like this:

```
Assumptions: $p$ is a polynomial of degree $n > 2$.
Input: Polynomial $p(x)$; tolerance $tol$; maximum number of iterations $N$.
Step 1: For $i = 1$ to $n - 2$ do Steps 2-5:
    Step 2: Find a root $x_0$ of $p(x)$ [using $tol$, $N$, and some root-finding method];
    Step 3: If error trying to find $x_0$ then
        return "Method failed. Root of degree $n - i + 1$ not found.";
    Step 4: Factor $p(x)$ as $q(x) \cdot (x - x_0)$;
    Step 5: Set $x_i = x_0$; $p(x) = q(x)$;
Output: Approximate roots.
```

To refine the pseudo-pseudo-code into pseudo-code, we will use Newton's method, assisted by Horner's method, in Step 2. The usual drawback of Newton's method, the requirement that the derivative be known and calculated, is but a small inconvenience when Horner's method is employed. But how do we represent polynomials in a computer program so that we can accomplish Steps 4 and 5? The same way we implement code to execute Horner's method. Pseudo-code for Horner's method, with an array:

```
Assumptions: $p$ is a polynomial of degree $n \geq 1$.
Input: array $[c]$ of coefficients of $p(x) = c_1 + c_2x + c_3x^2 + \cdots + c_{n+1}x^n$; $x_0$.
Step 1: Set $y = c_{n+1}$; $z = c_{n+1}$;
Step 2: For $j = n, n - 1, \ldots, 2$ do Step 3
    Step 3: Set $y = x_0y + c_j$; $z = x_0z + y$;
Step 4: Set $y = x_0y + c_1$;
Output: $y = p(x_0)$ and $z = p'(x_0)$.
```

As in synthetic division, there is no need to retain the variable to various exponents. Only the coefficients are needed to define a polynomial. So, in the program, a polynomial is represented by an array of numbers. Putting together our pseudo-pseudo code, Newton's method and Horner's method into a single program, we have a method for finding all the roots of a polynomial:

```
Assumptions: $p$ is a polynomial of degree $n > 2$ and $c_1$, the constant coefficient of $p$, is nonzero.
Input: array $[c]$ of coefficients of $p(x) = c_1 + c_2x + c_3x^2 + \cdots + c_{n+1}x^n$; tolerance $tol$; maximum number of
    iterations $N$; initial value $x_0$.
Step 1: Set $m = n$;
Step 2: For $i = 1$ to $n - 2$ do Steps 3-13:
    Step 3: Set $k = 0$; Set $x = x_0$;
    Step 4: While $|x - x_0| > tol$ or $k = 0$ do Steps 5-12:
        Step 5: If $k = N$ then return "Method failed. Not all roots found."
        Step 6: Set $x_0 = x$;
        Step 7: Set $d_m = c_{m+1}$; $z = c_{m+1}$;
        Step 8: For $j = m, m-1, \ldots, 2$ do Step 9
            Step 9: Set $d_{j-1} = x_0 d_j + c_j$; $z = x_0 z + d_{j-1}$;
        Step 10: Set $y = x_0 d_1 + c_1$;
        Step 11: Set $x = x_0 - \frac{y}{z}$;
        Step 12: Set $k = k + 1$;
    Step 13: Set $r_i = x$; $[c] = [d]$; $m = m - 1$;
Step 14: Set $D = \sqrt{c_2^2 - 4c_1c_3}$; $s_1 = -c_2 + D$; $s_2 = -c_2 - D$;
Step 15: If the real part of $c_2$ is negative, then set $r_{n-1} = \frac{s_1}{2c_3}$ and $r_n = \frac{2c_1}{s_1}$; else set $r_{n-1} = \frac{s_2}{2c_3}$ and $r_n = \frac{2c_1}{s_2}$;
Output: Array $[r_1, r_2, \ldots, r_n]$ of approximate roots.
```

Steps 4 through 12 implement Newton's method to find a single root, using Horner's method in Steps 7 through 10 to calculte the value of the polynomial and its derivative at x_0. Care is taken to calculate and store the coefficients $[d]$ of the quotient for easy referral in Step 13. It is assumed that the square root calculated in Step 14 is the principle branch of the complex square root. Steps 14 and 15 utilize an alternate form of the quadratic formula that avoids the subtraction of nearly equal quantities so much as possible.

Crumpet 15: Alternate Quadratic Formula

When the roots of $p(x) = ax^2 + bx + c$ are small, the numerator of the quadratic formula, $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$, is necessarily small. In this case, it is best to match the signs of $-b$ and $\pm\sqrt{b^2 - 4ac}$ in order to avoid subtracting quantities of nearly equal value. Choosing the sign of the square root term this way gives one of the roots as accurately as possible, but leaves the other root undetermined. Multiplying both numerator and denominator by the conjugate of the numerator gives an alternate expression of the quadratic formula:

$$\begin{aligned} \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b \mp \sqrt{b^2 - 4ac}}{-b \mp \sqrt{b^2 - 4ac}} &= \frac{b^2 - (b^2 - 4ac)}{2a(-b \mp \sqrt{b^2 - 4ac})} \\ &= \frac{4ac}{2a(-b \mp \sqrt{b^2 - 4ac})} \\ &= \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}. \end{aligned}$$

Expanding, we have

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{2c}{-b - \sqrt{b^2 - 4ac}}$$

and

$$\frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{2c}{-b + \sqrt{b^2 - 4ac}}.$$

However, there is little that can be done at this point if zero happens to be a double root. In this instance, both c_1 and c_2 will be zero or nearly zero, making both s_1 and s_2 very small. This is why the set of assumptions includes the stipulation $c_1 \neq 0$. This ensures that zero is not a root of p.

Newton's method and polynomials

There is one more issue to address regarding the use of Newton's method for finding roots of polynomials. For a polynomial with real coefficients, if x_0 is real, so will be x_1, and x_2, and every successive iteration! There will be no hope of finding complex roots. This is not a problem if the polynomial has at most two complex roots. The real roots will be found and the resulting quadratic will hold the two complex roots. The complex roots will be uncovered by the quadratic formula. In general, though, we can not count on a polynomial having at most two complex roots. Our method should work for polynomials with arbitrarily many complex roots, including the case when all roots are complex.

The fix is not difficult, with one proviso. Mathematically, Newton's method and Horner's method work just as well with complex numbers as they do with real numbers. As long as the programming language you are using can handle complex numbers, just begin with a complex (not purely real) initial approximation x_0, and complex roots will be found! Even so, it is possible that all the real roots are found first and what remains will be a polynomial with more than two complex roots and no real roots. This is where the inaccuracy of floating point arithmetic is actually helpful! Neither the coefficients nor the value of x_0 will be purely real due to round-off error. The complex roots will generally be found.

Müller's Method

Another very fast method for finding roots of equations is Müller's method . In principle, it is very much like the secant method. With the secant method, two initial approximations p_0 and p_1 are made. The secant line through the points $(p_0, f(p_0))$ and $(p_1, f(p_1))$ is drawn and its intersection with the x-axis gives p_2. With Müller's method, three initial approximations p_0, p_1, and, p_2 are needed. The parabola through the points $(p_0, f(p_0))$, $(p_1, f(p_1))$, and $(p_2, f(p_2))$ is drawn and its intersection with the x-axis gives p_3. There are a couple of issues to deal with, however. First, if the parabola so drawn crosses the x-axis at all, it crosses it twice. We need to choose one of the zeros for p_3. Second, it is possible the parabola will not cross the x-axis at all.

Solving the problem of which root to choose is simple. We assume the approximation p_2 is better than the others, so we choose the root that is closest to p_2. Actually, that solves the second "problem" too. Even when the parabola does not cross the x-axis, it has zeros. They are complex. And we do not worry about that. We simply take the complex root that is closest to p_2. This has the nice advantage that even when the coefficients of $p(x)$ are all real and p_0, p_1, and, p_2 are all real, and all the roots of $p(x)$ are complex, it will find a complex root.

As to the business of finding the parabola passing through $(p_0, f(p_0))$, $(p_1, f(p_1))$, and $(p_2, f(p_2))$, we will seek a parabola $P(x)$ of the form

$$P(x) = a(x - p_2)^2 + b(x - p_2) + c.$$

Making the substitutions $x = p_i$ and $P(x) = f(p_i)$ leads to the three equations

$$\begin{aligned} f(p_0) &= a(p_0 - p_2)^2 + b(p_0 - p_2) + c \\ f(p_1) &= a(p_1 - p_2)^2 + b(p_1 - p_2) + c \\ f(p_2) &= c \end{aligned}$$

So we find out immediately that $c = f(p_2)$ and we must solve the simultaneous equations

$$\begin{aligned} f(p_0) - f(p_2) &= a(p_0 - p_2)^2 + b(p_0 - p_2) \\ f(p_1) - f(p_2) &= a(p_1 - p_2)^2 + b(p_1 - p_2) \end{aligned}$$

for a and b. The solution is

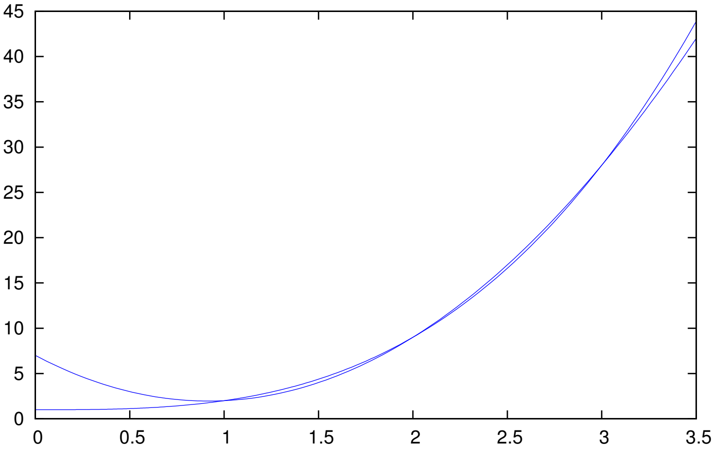
$$\begin{aligned} b &= \frac{(p_0 - p_2)^2(f(p_1) - f(p_2)) - (p_1 - p_2)^2(f(p_0) - f(p_2))}{(p_0 - p_2)(p_1 - p_2)(p_0 - p_1)} \\ a &= \frac{(p_1 - p_2)(f(p_0) - f(p_2)) - (p_0 - p_2)(f(p_1) - f(p_2))}{(p_0 - p_2)(p_1 - p_2)(p_0 - p_1)}. \end{aligned}$$

Now plugging a, b, and c into the quadratic formula gives us roots $x = p_2 - \frac{2}{b \pm \sqrt{b^2 - 4ac}}$. To choose the one closest to p_2, we compare $|b + \sqrt{b^2 - 4ac}|$ with $|b - \sqrt{b^2 - 4ac}|$ and use the larger. This gives us the smallest value for $|x - p_2|$, the distance of the root from p_2.

For example, we will use Müller's method with $p_0 = 1$, $p_1 = 2$, and $p_2 = 3$ to find a root of $f(x) = x^3 + 1$. We calculate

$$\begin{aligned}
\delta_0 &= f(p_0) - f(p_2) = 2 - 28 = -26 \\
\delta_1 &= f(p_1) - f(p_2) = 9 - 28 = -19 \\
h_0 &= p_0 - p_2 = -2 \\
h_1 &= p_1 - p_2 = -1 \\
h_2 &= p_0 - p_1 = -1
\end{aligned}$$

so we get $c = 28$, $b = \frac{h_0^2\delta_1 - h_1^2\delta_0}{h_0h_1h_2} = \frac{4(-19)-1(-26)}{-2} = 25$, and $a = \frac{h_1\delta_0 - h_0\delta_1}{h_0h_1h_2} = \frac{-1(-26)-(-2)(-19)}{-2} = 6$. A close look at the graphs of $f(x)$ and $P(x) = 6x^2 + 25x + 28$ shows that they do meet three times (at the required points), and that $P(x)$ does not have real roots:



.

$b \pm \sqrt{b^2 - 4ac} = 25 \pm \sqrt{625 - 672} = 25 \pm i\sqrt{47}$. Since $|25 + i\sqrt{47}| = |25 - i\sqrt{47}|$, it does not matter which root we take. Selecting $p_3 = p_2 - \frac{2c}{b-\sqrt{b^2-4ac}}$, we get $p_3 = 3 - \frac{56}{25-i\sqrt{47}} = \frac{11}{12} - \frac{\sqrt{47}}{12}i$. Continuing this process gives the iterates $0.75238 - 0.75810i,\ 0.57069 - 0.84288i, \ldots, 0.50000 - 0.86603i$, converging to $\frac{1}{2} - \frac{\sqrt{3}}{2}i$.

Crumpet 16: Orders of convergence

The order of convergence of Müller's method to a simple root (one that is not repeated) is

$$\left(\frac{\sqrt{11}}{3\sqrt{3}} + \frac{19}{27}\right)^{\frac{1}{3}} + \frac{4}{9\left(\frac{\sqrt{11}}{3\sqrt{3}} + \frac{19}{27}\right)^{\frac{1}{3}}} + \frac{1}{3} \approx 1.839286755214161$$

and to a double root,

$$\left(\frac{\sqrt{139}}{24\sqrt{3}} + \frac{8}{27}\right)^{\frac{1}{3}} + \frac{7}{36\left(\frac{\sqrt{139}}{24\sqrt{3}} + \frac{8}{27}\right)^{\frac{1}{3}}} + \frac{1}{6} \approx 1.233751928528259.$$

The method of Laguerre converges to a simple root with order 3.

References [23, 26]

The following chart summarizes the relative strengths and weaknesses of Newton's method, the secant method, and Müller's method.

	Newton's	Secant	Müller's
Initial values needed	1	2	3
Derivative needed?	Yes	No	No
Order of Convergence[5]	2	≈ 1.618	≈ 1.839
Automatic discovery of complex roots?	No	No	Yes
Simplified in the case of polynomials?	Yes	No	No

Key Concepts

Synthetic division: A method for dividing a polynomial $p(x)$ by a monomial $(x - x_0)$ using only addition, multiplication, and the coefficients of p. The process is identical to evaluating a polynomial by nesting. Synthetic division simply provides an organizational tool so that nesting can be accomplished simply with pencil and paper.

Horner's method: A method where the value of a polynomial and its derivative at a single point are calculated simultaneously via synthetic division.

Müller's method: A root-finding method similar to the secant method where instead of using a secant line a parabola is used.

Deflation: The method of replacing a polynomial $p(x)$ by the product of a monomial $(x - x_0)$ and a polynomial $q(x)$ of degree one less than that of the original polynomial.

Exercises

1. C Write a function that calculates the roots of a quadratic function using the alternate quadratic formula when appropriate. The first line of your function should be

   ```
   function [r1,r2] = quadraticRoots(a,b,c)
   ```

   where `r1` and `r2` are the roots of $p(x) = ax^2 + bx + c$. This way, the values `r1` and `r2` are returned by the function in an array. The function is called like this:

   ```
   [s,t]=quadraticRoots(1,2,3),
   ```

   setting `s` to the value of one of the roots and `t` to the other. Test your code well by comparing outputs of your function to hand/calculator computations.

2. C Write a function that implements Horner's method. The first line of your function should be

   ```
   function [p,pprime] = horner(x0,c)
   ```

   where `c` is an array containing the coefficients of the polynomial, `x0` is the number at which to evaluate it, `p` is the value of the polynomial at `x0`, and `pprime` is the value of the derivative of the polynomial at `x0`. This way, the values `p` and `pprime` are returned by the function in an array. The function is called like this:

   ```
   [y,yy]=horner(-2,[5,4,3,2,1]),
   ```

   setting `y` to the value of the polynomial and `yy` to the value of its derivative. Test your code well by comparing outputs of your function to hand/calculator computations.

3. C Write a function that implements Newton's method with Horner's method. The first line of your function should be

   ```
   function x = newtonhorner(c,x0,tol,N)
   ```

   where `c` is an array containing the coefficients of the polynomial, `x0` is the initial value, `tol` is the tolerance, and `N` is the maximum number of iterations before giving up. The code should be similar to code you wrote to implement Newton's method before, but this code will only work for polynomials. Inside your `newtonhorner` function, DO NOT write Horner's method code. Just call the `horner` function you wrote in question 2. Test your code well by comparing outputs of your function to outputs from the code you wrote in question 1 on page 55.

4. C Complete the code for the deflate function begun here.

   ```
   % This function will deflate a polynomial
   % given a root.
   % INPUT: coefficients c of the polynomial;
   %        a root r of the polynomial.
   % OUTPUT: coefficients d of the deflated
   %         polynomial.
   function d = deflate(c,r)

   end%function
   ```

5. C Write a function implementing Müller's method.

6. Use Horner's method/synthetic division to find $g(2)$ and $g'(2)$. Do not use a computer.

   (a) $g(x) = 3x^3 + 12x^2 - 13x - 8$ [S]

   (b) $g(x) = -7 + 8x - 3x^2 + 5x^3 - 2x^4$ [A]

7. Use Horner's method to calculate $g(-2)$ and $g'(-2)$ where $g(x) = 4x^4 - 5x^3 + 6x - 7$. Do not use a computer.

8. Use your work from question 6 to help execute two iterations of Newton's method using a pencil, paper, calculator, and Horner's method/synthetic division. Use initial value $x_0 = 2$. [S] [A]

9. Use your work from question 7 to help execute two iterations of Newton's method using a pencil, paper, calculator, and Horner's method/synthetic division. Use initial value $x_0 = -2$.

10. Compute x_2 of Newton's method by hand (using Horner's method/synthetic division) for $f(x) = x^3 + 4x - 8$ starting with $x_0 = 0$.

11. Find x_2 of Newton's method by hand (using Horner's method/synthetic division) for $f(x) = x^4 - 2x^3 - 4x^2 + 4x + 4$ using $x_0 = 2$.

12. Using Horner's method as an aid, and not using your calculator, find the first iteration of Newton's method for the function $f(x) = 2x^3 - 10x + 1$ using $x_0 = 2$.

13. Demonstrate two iterations of Newton's method (using Horner's method/synthetic division) applied to $f(x) = 5x^3 - 2x^2 + 7x - 3$ with $p_0 = 1$ by hand.

14. Find all the roots of the polynomial as follows. Use Newton's method with tolerance 10^{-5} to approximate a root of the polynomial. You may use your `newtonhorner` function from question 3. Then use synthetic division to deflate the polynomial one degree. Do not use a computer for deflation. Then use Newton's method with tolerance 10^{-5} to approximate a root of the deflated polynomial. Then use synthetic division to deflate the deflated polynomial one degree. Repeat until the deflated polynomial is quadratic. Once this happens, use the quadratic formula (or alternate quadratic formula) to find the last two roots.

    (a) $g(x) = x^4 + 6x^3–59x^2 + 144x–144$ [S]

    (b) $g(x) = -280 + 909x - 154x^2 - 178x^3 + 54x^4 + 9x^5$ [A]

15. Find all the roots of the polynomial as follows. Use Newton's method with tolerance 10^{-5} to approximate a root of the polynomial. You may use your `newtonhorner` function from question 3. Then use synthetic division to deflate the polynomial one degree. You may use your `deflate` function from question 4 for deflation. Then use Newton's method with tolerance 10^{-5} to approximate a root of the deflated polynomial. Then use synthetic division to deflate the deflated polynomial one degree. Repeat until the deflated polynomial is quadratic. Once this happens, use the quadratic formula to find the last two roots. You may use your `quadraticRoots` function from question 1 for solving the quadratic.

    (a) $g(x) = x^4 - 2x^3 - 12x^2 + 16x - 40$ [S]

    (b) $g(x) = 56 - 152x + 140x^2 - 17x^3 - 48x^4 + 9x^5$ [A]

16. For each root you found in question 14 except the first one, use it as an initial approximation in Newton's method with tolerance 10^{-5} to see if you can refine your roots. Do they change? [S] [A]

17. $f(x) = x^3 - 1.255x^2 - .9838x + 1.2712$ has a root at $x = 1.12$.

    (a) Use Newton's method with an initial approximation $x_0 = 1.13$ to attempt to find this root. Explain what happens.

    (b) Find all the roots of $f(x)$.

18. About 800 years ago John of Palermo challenged mathematicians to find a solution of the equation $x^3 + 2x^2 + 10x = 20$. In 1224, Fibonacci answered the call in the presence of Emperor Frederick II. He approximated the only real root using a geometric technique of Omar Khayyam (1048-1131), arriving at the estimate

$$1 + 22\left(\frac{1}{60}\right) + 7\left(\frac{1}{60}\right)^2 + 42\left(\frac{1}{60}\right)^3 + 33\left(\frac{1}{60}\right)^4 + 4\left(\frac{1}{60}\right)^5 + 40\left(\frac{1}{60}\right)^6.$$

    How accurate was his approximation?

    Reference [5, pg. 96 ex. 10]

19. C Calculate the value of the polynomial at the given value of x in two different ways. (i) Use your `horner` function from question 2; and (ii) use an `inline()` function. Then (iii) compare the two results using the `==` operator.

    (a) $p(x) = x^4 - 2x^3 - 12x^2 + 16x - 40$ at $x = \sqrt{3}$ [S]

    (b) $q(x) = 56 - 152x + 140x^2 - 17x^3 - 48x^4 + 9x^5$ at $x = \pi/2$ [A]

    (c) $r(x) = x^6 + 11x^4 - 34x^3 - 130x^2 - 275x + 819$ at $\frac{1-\sqrt{5}}{2}$ [A]

    (d) $s(x) = 5x^{10} + 3x^8 - 46x^6 - 102x^4 + 365x^2 + 1287$ at $\frac{1}{e}$

20. Write a function that uses your functions from questions 1, 3, and 4 to find all the roots of a polynomial. Test your function well on polynomials of various degrees for which you know the roots. You may base your function on the pseudo-code on page 67, but your code should be significantly simpler since you are calling functions instead of writing their code. [A]

21. Use your code from question 20 to find all the solutions of the equation. [A]

    (a) $x^5 + 11x^4 - 34x^3 - 130x^2 - 275x + 819 = 0$

    (b) $5x^5 + 3x^4 - 46x^3 - 102x^2 + 365x + 1287 = 0$

22. Find all the roots of $g(x) = 25x^3 - 105x^2 + 148x - 174$.

23. Recall that there are some similarities between the secant method and Müller's method. They each require multiple initial approximations. They each involve calculating the zero of some function passing through these initial points. They both give superlinear convergence to simple roots. And, of course, they are both root finding methods. Let's tweak the idea in the following way. To find roots of g, start as with the secant method, using two approximations, x_0 and x_1. Then, instead of using the zero of a line through $(x_0, g(x_0))$ and $(x_1, g(x_1))$, find the function of the form

$$h(x) = ax^3 + b$$

    passing through $(x_0, g(x_0))$ and $(x_1, g(x_1))$. Let x_2 be the zero of h. Then repeat with x_1 and x_2 to get x_3, and so on.

(a) Let $g(x) = 2\ln(1+x^2) - x$, $x_0 = 5$ and $x_1 = 6$. Find x_2 using this method.

(b) Find a formula for x_2 given any function $g(x)$ and any initial conditions x_0 and x_1. Your formula should be in terms of x_0, x_1, $g(x_0)$, and $g(x_1)$.

(c) Find a general formula for x_n in terms of x_{n-2}, x_{n-1}, $g(x_{n-2})$, and $g(x_{n-1})$).

(d) Write a function that implements this method and prints out each iteration.

(e) Use your function to decide whether the order of convergence for this method is linear or superlinear.

24. Pick a function whose root(s) you know exactly. Use Müller's method to find one of the roots. Use three consecutive iterations to estimate the order of convergence.

25. The errors in three consecutive iterations of Müller's method are shown in the table. Use this information to estimate the order of convergence.

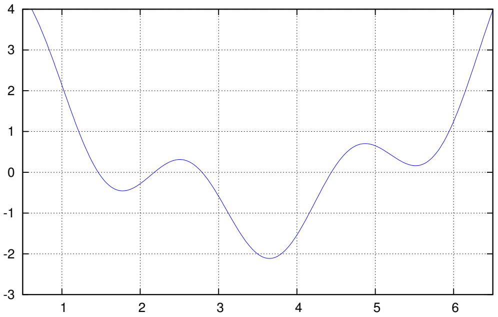
n	$\lvert x_n - x\rvert$
12	$1.53627(10)^{-349}$
13	$1.67365(10)^{-642}$
14	$1.83922(10)^{-1181}$

26. The graph of $f(x)$ is shown. Find distinct sets of values p_0, p_1, and p_2 for which Müller's method

(a) will lead to a complex value for p_3.

(b) will lead to the root at $x \approx 4.4$.

(c) will lead to the root at $x \approx 2.8$.



27. The function shown in question 26 is $f(x) = \frac{x^2-7x+10}{2} + \sin(3x)$. Use this information to test your conjectures in question 26.

2.7 Bracketing

Bisection is called a bracketed root-finding method. A root is known to lie within a certain interval. Each iteration reduces the size of the interval and maintains the guarantee the root is within. At each step of the algorithm, the root is known to be between the latest estimate and one of the previous. These bounds form a bracket around the root. As the algorithm proceeds, the bracket decreases in size until it is smaller than some tolerance, at which point the root is known to be close and the algorithm stops.

The problem with bisection is its linear order of convergence. Compared to superlinear methods like the secant method and Newton's method, the bisection method just creeps along. But the bisection method has something the secant method and Newton's method do not—certainty of convergence. Yes, the secant method and Newton's method are fast when they converge, but there is no guarantee they will converge at all.

Methods combining the virtues of the bisection method (guaranteed convergence) and some higher order method (speed) are called safeguarded methods. They are guaranteed to converge and can do so quickly when the root is near. Any superlinear method may be bracketed, producing a safeguarded method.

Bracketing

Bracketing means maintaining an interval in which a root is known to lie. Bracketing is used in the bisection method. With each iteration, the root is known to lie between the two latest approximations. Bracketing is not used in the secant method nor Newton's method. There is no guarantee a root remains near the latest approximations.

It is not difficult, however, to combine the bisection method with the secant method or Newton's method, or any other high order method for that matter, to form a hybrid method where the root remains bracketed and there is a chance for fast convergence. In such a method, a candidate for the next iteration is computed according to the high order method. If this candidate lies within the bracket, it becomes the next iteration. If the candidate lies outside the bracket, the bisection method is used to compute the next iteration instead.

Bracketed secant method, better known as the method of false position or regula falsi, provides an elementary example. In fact, the high order method (the secant method) always produces a value inside the bracket, so checking that point is not necessary. Where false position and the secant method differ is choosing which of the previous two iterations to keep. In the secant method, it is always the latest iteration which is kept for the next. In false position, the latest iteration which maintains a bracket about the root is kept for the next whether that iteration is the latest or not. Bracketed Newton's method provides a slightly more advanced example because it is entirely possible an iteration of Newton's method will land outside the bracket.

Take the function $g(x) = 3 - x - \sin(x)$ over the interval $[2, 3]$. f is continuous on $[2, 3]$, and $g(2) \approx 0.09$ and $g(3) \approx -0.14$ have opposite signs. Thus $[2, 3]$ brackets a root of g, so let $x_0 = 2$ and $x_1 = 3$. The table shows the computation of the next iteration for bracketed secant method and bracketed Newton's method.

	x_0	x_1	candidate x_2	x_2
bracketed secant	2	3	$x_1 - g(x_1)\frac{x_1-x_0}{g(x_1)-g(x_0)} \approx 2.3912$	2.3912
bracketed Newton's	2	3	$x_1 - \frac{g(x_1)}{g'(x_1)} \approx -11.101$	2.5

In bracketed secant, the candidate x_2 is accepted, but in bracketed Newton's method, the candidate x_2 is outside the bracket so it is discarded and x_2 according to the bisection method (2.5) is taken instead.

To set up the next iteration, $g(x_2)$ is calculated. Since $g(x_2)$ is negative in both methods, the old x_1, which was 3, is discarded and $x_0 = 2$ is "upgraded" to x_1 in order to maintain the bracket. This way, g has opposite signs at $x_1$and x_2. The following table demonstrates this decision process plus the computation of the next iteration.

	$g(x_2)$	x_1	x_2	candidate x_3	x_3
bracketed secant	-0.073141	2	2.3912	$x_2 - g(x_2)\frac{x_2-x_1}{g(x_2)-g(x_1)} \approx 2.2165$	2.2165
bracketed Newton's	-0.098472	2	2.5	$x_2 - \frac{g(x_2)}{g'(x_2)} \approx 2.0048$	2.0048

Can you fill in x_4 based on the values in the following table? Notice the old x_1 must be "upgraded" in bracketed secant but not in bracketed Newton's. Why? Answers on page 81.

	$g(x_3)$	x_2	x_3	candidate x_4	x_4
bracketed secant	-0.015215	2	2.2165	$x_3 - g(x_3)\frac{x_3-x_2}{g(x_3)-g(x_2)} \approx 2.1854$	?
bracketed Newton's	0.087906	2.5	2.0048	$x_3 - \frac{g(x_3)}{g'(x_3)} \approx 2.1565$	?

The next 5 iterations of each method are given here in case you would like to try your hand at computing a few. And now is a good time to do so. These values were computed using the subsequent computer code.

	bracketed	
	secant	Newton's
x_5	2.18062942638407	2.17925592233708
x_6	2.17988957044102	2.17975682599184
x_7	2.17977718322867	2.17975706647997
x_8	2.17976012038625	2.17975706648003
x_9	2.17975753008587	2.17975706648003

False position (bracketed secant method) code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Written by Dr. Len Brin           20 May 2014 %
% Purpose: Implementation of the Method of      %
%        False Position.                        %
% INPUT: function g; initial values a and b;    %
%        tolerance TOL; maximum iterations N    %
% OUTPUT: approximation x and number of         %
%        iterations i; or message of failure    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,i] = falsePosition(g,a,b,TOL,N)
  i=1;
  A=g(a);
  B=g(b);
  while (i<N)
    b
    x=b-B*(b-a)/(B-A);
    if (abs(x-b)<TOL)
      return
    end%if
    X=g(x);
    if ((B<0 && X>0) || (B>0 && X<0))
      a=b; A=B;
    end%if
    b=x; B=X;
    i=i+1;
  end%while
  x="Method failed---maximum number of iterations reached";
end%function
```

Bracketed Newton's method code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Written by Dr. Len Brin           20 May 2014 %
% Purpose: Implementation of bracketed Newton's %
%        method.                                %
% INPUT: function g; its derivative gp; initial %
%        values a and b; tolerance TOL; maximum %
%        iterations N                           %
% OUTPUT: approximation x and number of         %
%        iterations i; or message of failure    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,i] = bracketedNewton(g,gp,a,b,TOL,N)
  i=1;
  A=g(a);
  B=g(b);
  while (i<N)
    b
    x=b-B/gp(b);
    if (x<min([a,b]) || x>max([a,b]))
      x=b+(a-b)/2;
    end%if
    if (abs(x-b)<TOL)
      return
    end%if
    X=g(x);
    if ((B<0 && X>0) || (B>0 && X<0))
      a=b; A=B;
    end%if
    b=x; B=X;
    i=i+1;
  end%while
  x="Method failed---maximum number of iterations reached";
end%function
```

`falsePosition.m` and `bracketedNewton.m` may be downloaded at the companion website.

The code for bracketed secant method and bracketed Newton's method are very similar. In fact, they are nearly identical. There are only two differences besides the commentary at the beginning. Where bracketed secant has the line `x=b-B*(b-a)/(B-A);`, bracketed Newton's has the line `x=b-B/gp(b);`. This is the essential difference between the two as this is where the high order method is executed. The only other difference is that bracketed Newton's includes three lines where it checks whether `x` lands within the bracket and executes one step of the bisection method if not:

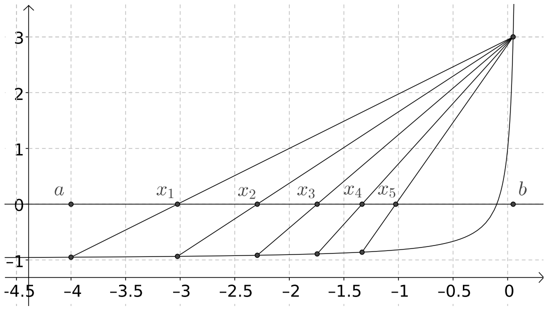
```
    if (x<min([a,b]) || x>max([a,b]))
      x=b+(a-b)/2;
    end%if
```

Actually, we could add these three lines to the bracketed secant method and it would run just the same. It is impossible for the secant method to produce a value of `x` outside the bracket, so the bisection step would never be executed. The only essential difference between the two functions is the execution of the high order method.

We can use this observation to create a sort of blueprint for bracketing any high order method. Steffensen's, Müller's (as long as the approximation stays real), or Sidi's (section 3.2), for example, can be bracketed this way. The following pseudo-pseudo-code represents such a blueprint, giving guidance on how to safeguard a high order method by combining it with bisection.

```
Assumptions: $g$ is continuous on $[a, b]$. $g(a)$ and $g(b)$ have opposite signs.
Input: Interval $[a, b]$; function $g$; desired accuracy $tol$; maximum number of iterations $N$; any other variables,
    like $g'$ in the case of Newton's method, needed to iterate the superlinear method.
Step 1: Set $A = g(a)$; $B = g(b)$; $i = 2$;
Step 2: Initialize any other variables needed for superlinear();
Step 3: While $i < N$ do Steps 4-10:
    Step 4: Set $x = \text{superlinear}(a, b, g, \ldots)$;
    Step 5: If $(x - a)(x - b) > 0$ then $x = b + \frac{a-b}{2}$;
    Step 6: If $|x - b| < tol$ then return $x$
    Step 7: Set $X = g(x)$;
    Step 8: If $BX < 0$ then set $a = b$; $A = B$;
    Step 9: Set $b = x$; $B = X$; $i = i + 1$;
    Step 10: Update any other variables needed for superlinear();
Step 11: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $m$ within $tol$ of exact root, or message of failure.
```

Figure 2.7.1: A troublesome function for the bracketed secant method.



As motivation for the need to develop bracketed versions of other high order methods, consider the particularly problematic function $g(x) = \frac{1+10x}{1-10x}$. It has a root at $-\frac{1}{10}$, but the bracketed secant method can be very slow to converge to this root. Figure 2.7.1 illustrates this slow convergence beginning with the bracket $[a, b] = [-4, .05]$. With this unfortunate choice of bracket, the method takes 45 iterations to achieve 10^{-5} accuracy. A smarter algorithm would not only check that each iterate lands within the brackets, but would also check to see that the high order method is making quick progress toward the root. If it detected that convergence was slow, say slower than bisection would be, it would take a bisection step instead. Note that bracketed Newton's method does not have a significant problem with this function. Given the same initial bracket, it converges to within 10^{-5} of the root in only 10 iterations (the first 4 of which are bisection steps). Alas, Newton's method requires use of the derivative. A fast bracketed root-finding method that does not require knowledge of the derivative would be quite useful.

In the early 1970s, Richard Brent built upon the work of van Wijngaarden and Dekker to produce a bracketed method that combines bisection, the secant method, and inverse quadratic interpolation, all the while checking to make sure the high order method is making sufficiently quick progress toward a root. The result is what is now known as Brent's method [3]. It does not require knowledge of the derivative. It is fast. It is guaranteed to converge. Consequently, it is a popular all-purpose method for finding a root within a bracket when the derivative is not accessible. The full details of Brent's method will not be presented here, but a significant step toward that method will. The method presented here is similar to the MATLAB function `fzero` [22].

Inverse Quadratic Interpolation

You may recall, in Müller's method, three initial approximations, say a, b, and, c are needed. The parabola through the points $(a, g(a))$, $(b, g(b))$, and $(c, g(c))$ is drawn and its intersection with the x-axis gives the next iteration. The key elements of this method, the process of fitting a quadratic function to the three points, is called interpolation. Thus Müller's method could just as well be called the "quadratic interpolation method".

As you may have guessed, the method of inverse quadratic interpolation is similar. Instead of fitting a quadratic function to the points $(a, g(a))$, $(b, g(b))$, and $(c, g(c))$, the roles of x and y are reversed. A quadratic function is fitted to the points $(g(a), a)$, $(g(b), b)$, and $(g(c), c)$ instead. Since x is a function of y in this case, the quadratic will cross the x-axis exactly once, when $y = 0$. Evaluating the quadratic at 0 gives the next iteration. Figure 2.7.2 shows quadratic interpolation and inverse quadratic interpolation on the same set of three points. In quadratic interpolation, y is treated as a function of x. In inverse quadratic interpolation, x is treated as a function of y. Inverse quadratic interpolation avoids the main complication of quadratic interpolation—calculating its x-axis crossings. In quadratic interpolation, the quadratic may cross the x-axis twice or not at all! Either way, some choice needs to be made at every step, and the roots of the quadratic involve the quadratic formula. In inverse quadratic interpolation, the quadratic is guaranteed to cross the x-axis exactly once, and finding the crossing is just a matter of evaluating the quadratic at 0. That is, $y = 0$. Remember, the quadratic gives x as a function of y.

Referring back to the derivation of Müller's method on page 69, forcing the parabola to pass through the points (a, A), (b, B), and (c, C), and swapping the roles of x and y, a formula for the inverse parabola, q, just falls out:

$$q(y) = q_0(y - B)^2 + q_1(y - B) + q_2$$

where

$$\begin{aligned}
q_2 &= b \\
q_1 &= \frac{(A-B)^2(c-b)-(C-B)^2(a-b)}{(A-B)(C-B)(A-C)} \\
q_0 &= \frac{(C-B)(a-b)-(A-B)(c-b)}{(A-B)(C-B)(A-C)}.
\end{aligned}$$

Crumpet 17: Quadratic interpolation order of convergence

The method of inverse quadratic interpolation has order of convergence about 1.84 under reasonable assumptions. If the function whose root is being determined has three continuous derivatives in a neighborhood of the root, the latest three approximations are sufficiently close, and the root is simple, then the order of convergence is the real solution of

$$\alpha^3-\alpha^2-\alpha-1=0.$$

We can use inverse quadratic interpolation to approximate it!

```
>> format('long')
>> f=inline('x^3-x^2-x-1')
f = f(x) = x^3-x^2-x-1
>> [res,i]=inverseQuadratic(f,1,2,10^-12,100)
res =  1.83928675521416
i =  8
```

The exact solution is

$$\alpha=\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{19}{27}\right)^{\frac{1}{3}}+\frac{4}{9\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{19}{27}\right)^{\frac{1}{3}}}+\frac{1}{3}.$$

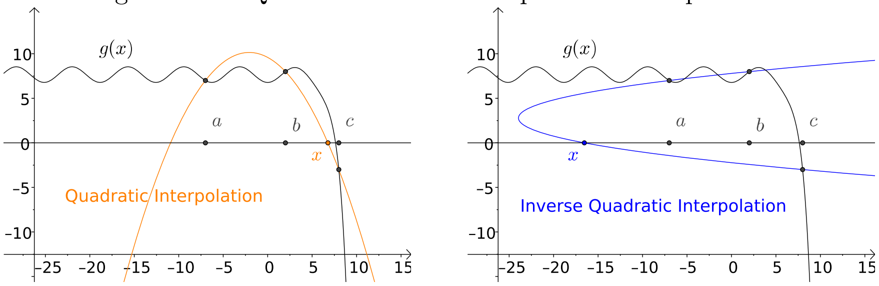
You may recognize this as the order of convergence for Müller's method. Indeed, any quadratic interpolation method converges to a simple root with this order.

Reference [29]

The x-axis crossing is, therefore,

$$\begin{aligned}
x &= q(0) \\
&= B^2q_0-Bq_1+q_2 \\
&= B^2\frac{(C-B)(a-b)-(A-B)(c-b)}{(A-B)(C-B)(A-C)}-B\frac{(A-B)^2(c-b)-(C-B)^2(a-b)}{(A-B)(C-B)(A-C)}+b \\
&= \frac{\left[B^2(C-B)+B(C-B)^2\right](a-b)-\left[B^2(A-B)+B(A-B)^2\right](c-b)}{(A-B)(C-B)(A-C)}+b \\
&= \frac{\left[-B^2C+BC^2\right](a-b)-\left[-B^2A+BA^2\right](c-b)}{(A-B)(C-B)(A-C)}+b \\
&= \frac{BC(C-B)(a-b)-BA(A-B)(c-b)}{(A-B)(C-B)(A-C)}+b \\
&= b+\frac{\frac{B}{A}(\frac{C}{B}-1)(a-b)-\frac{A}{C}(1-\frac{B}{A})(c-b)}{(1-\frac{B}{A})(\frac{C}{B}-1)(\frac{A}{C}-1)} \\
&= b+\frac{\frac{A}{C}(1-\frac{B}{A})(c-b)-\frac{B}{A}(\frac{C}{B}-1)(a-b)}{(\frac{B}{A}-1)(\frac{B}{C}-1)(\frac{A}{C}-1)}.
\end{aligned}$$

Figure 2.7.2: Quadratic and inverse quadratic interpolation.



To make the compuation of x a little more programmer-friendly, some new variables are introduced. Let

$$r = \frac{B}{A} - 1, \quad s = \frac{C}{B} - 1, \quad t = \frac{A}{C} - 1$$

so

$$x = b - \frac{r(t+1)(c-b) + s(r+1)(a-b)}{rst}. \tag{2.7.1}$$

Inverse quadratic interpolation can be bracketed just like any other high order method. But it does present an interesting question that not all high order methods do. Three points are necessary for a quadratic interpolation, so when they are used to produce the next iteration, a fourth point is generated. Of the four points, the computer needs to decide which two will become the next bracket, and which point should be the third needed for the next interpolation. But we are getting ahead of ourselves.

Each iteration begins with three points, $(a, g(a))$, $(b, g(b))$, and $(c, g(c))$ where a and b bracket a root and c is a third point. For the first iteration, only the bracket is given. c is set equal to a. For every iteration, the signs of $g(a)$ and $g(b)$ are checked to ensure that a and b bracket a root. If they are opposite, the method proceeds. If they are the same, that means $g(b)$ and $g(c)$ must have opposite signs, so a is set equal to c. Next, the absolute values of $g(a)$ and $g(b)$ are checked. If $|g(a)| < |g(b)|$, the labels of a and b are switched and c is set equal to the new value of a. After these initial checks, the computation of the next iteration begins with assurance that a root lies between a and b; b is likely the best estimate of the root to date; and c is likely the worst estimate of the root to date.

If $c = a$ after the initial checks and possible relabeling, then quadratic interpolation is impossible. The next iteration is generated by the secant method (linear interpolation) instead. If $c \neq a$ after the initial checks and possible relabeling, a candidate for the next iteration, x, is calculated according to inverse quadratic interpolation. If the candidate lies within the bracket, it is accepted as the next iteration. If it lies outside the bracket, a step of the bisection method is used instead. In either case, c is set equal to b and b is set equal to x. For bracketed inverse quadratic interpolation, this completes one iteration. The method is then repeated until a sufficiently good approximation is found.

In the best-case scenario, inverse quadratic interpolation is used at every step and convergence is superlinear with order about 1.84. In the worst-case scenario, one of the high order methods is used at every step, but the function is pathological and convergence is slow, possibly even slower than bisection. Slow convergence is rare, though, and the actual order of convergence can not be pinned down in general. The method switches between methods of different orders. The best we can say is it is usually fast.

Bracketed inverse quadratic interpolation code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Written by Dr. Len Brin            21 May 2014 %
% Purpose: Implementation of bracketed inverse   %
%        quadratic interpolation method.         %
% INPUT: function g; initial values a and b;     %
%        tolerance TOL; maximum iterations N0    %
% OUTPUT: approximation x and number of          %
%        iterations i; or message of failure     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,i] = bracketedInverseQuadratic(g,a,b,TOL,N0)
  i=1;
  A=g(a);
  B=g(b);
  c=a; C=A;
  while (i<N0)
    b
    if (B*A>0)
      a=c; A=C;
    end%if
    if (abs(A) < abs(B))
      c=b; C=B;
      b=a; B=A;
      a=c; A=C;
    end%if
    if (a==c)
      x=(b*A-a*B)/(A-B);
    else
      r=B/A-1; s=C/B-1; t=A/C-1;
      p=(t+1)*r*(c-b)+(r+1)*s*(a-b);
      q=t*s*r;
      x=b-p/q;
    end%if
    if (x<min([a,b]) || x>max([a,b]))
      x=b+(a-b)/2;
    end%if
    if (abs(x-b)<TOL)
      disp(" ");
      return
    end%if
    c=b; C=B;
    b=x; B=g(b);
    i=i+1;
  end%while
  x="Method failed---maximum number of iterations reached";
end%function
```

Applying the bracketed inverse quadratic interpolation method to the problematic function $g(x) = \frac{1+10x}{1-10x}$ over the interval $[-4, .05]$ yields the result within 10^{-5} accuracy in only 11 iterations. The method took only 1 iteration more than bracketed Newton's without requiring knowledge of the derivative of g! `bracketedInverseQuadratic.m` may be downloaded at the companion website.

Stopping

In all of our root-finding methods, the algorithm stops when the difference between consecutive iterations is less than some tolerance. This criterion is based on the assumption that the error will be no more than this difference. And that is a safe assumption for any method that is converging superlinearly when it quits. Indeed, it is even safe for the linearly converging bisection method where the difference between consecutive iterations is exactly the theoretical bound on the error.

The criterion is not safe when a superlinear method is used far enough from a root that superlinear convergence is not observed. This is exactly what happens in figure on page 77. The difference between consecutive iterations is actually larger than the absolute error at every step. This is an unusual situation, but it can happen.

The criterion is also not safe when a method is linearly convergent with a limiting convergence constant $\lambda > \frac{1}{2}$. However, linearly convergent methods should never be used on their own as there is always a faster alternative.

There is one more important consideration regarding stopping. Stopping when the difference between consecutive iterations is less than some tolerance is dependent on the absolute error. When roots could be very small or very large, it is perhaps better to use a criterion based on relative error. Instead of stopping when $|x_{n+1} - x_n| < tol$, for example, we would instead stop when $|x_{n+1} - x_n| < tol \cdot |x_{n+1}|$.

Key Concepts

Bracketing: Iteratively refining an interval, also known as the bracket, in which a root is known to lie until it is small beyond some tolerance.

Inverse quadratic interpolation: A quadratic in y is fit to three consecutive approximations of a root. The intersection of the quadratic with the x-axis becomes the next iteration.

Bracketed secant method: A combination of the secant method and bisection method employing bracketing. At each iteration, if the secant method produces a value inside the current bracket, it becomes the next iteration. Otherwise bisection is used to produce the next iteration.

False position: Another name for the bracketed secant method.

Regula falsi: Another name for the bracketed secant method.

Bracketed Newton's method: A combination of Newton's method and the bisection method employing bracketing. At each iteration, if Newton's method produces a value inside the current bracket, it becomes the next iteration. Otherwise bisection is used to produce the next iteration.

Bracketed inverse quadratic interpolation: A combination of inverse quadratic interpolation, the secant method, and bisection employing bracketing. At each iteration, if inverse quadratic interpolation produces a value inside the current bracket, it becomes the next iteration. Otherwise either the secant method or bisection is used to produce the next iteration.

Exercises

1. Use the bracketed secant method (false position) to find a root in the indicated interval, accurate to within 10^{-2}.
   (a) $f(x) = 3 - x - \sin x$; $[2, 3]$ [A]
   (b) $g(x) = 3x^4 - 2x^3 - 3x + 2$; $[0, 1]$
   (c) $g(x) = 3x^4 - 2x^3 - 3x + 2$; $[0, 0.9]$ [S]
   (d) $h(x) = 10 - \cosh(x)$; $[-3, -2]$
   (e) $f(t) = \sqrt{4 + 5\sin t} - 2.5$; $[-600, -500]$ [A]
   (f) $g(t) = \frac{3t^2 \tan t}{1-t^2}$; $[3490, 3491]$
   (g) $h(t) = \ln(3\sin t) - \frac{3t}{5}$; $[1, 2]$
   (h) $f(r) = e^{\sin r} - r$; $[-20, 20]$ [S]
   (i) $g(r) = \sin(e^r) + r$; $[-3, 3]$
   (j) $h(r) = 2^{\sin r} - 3^{\cos r}$; $[1, 3]$ [A]
2. Repeat question 1 using bracketed Newton's method. [S] [A]
3. Repeat question 1 using the secant method. Compare your answer with that of false position. [S] [A]
4. Repeat question 1 using Newton's method. Compare your answer with that of bracketed Newton's method. [S] [A]
5. Repeat question 1 using the computer and a tolerance of 10^{-6}. [S] [A]
6. Repeat question 2 using the computer and a tolerance of 10^{-6}. [S] [A]
7. Repeat question 1 using the computer, bracketed inverse quadratic interpolation, and a tolerance of 10^{-6}. [S] [A]
8. Compare the results of questions 5, 6, and 7. [A]
9. Write a bracketed Steffensen's method function. **REMARK:** Steffensen's method is a fixed point finding method. It solves the equation $f(x) = x$, not $f(x) = 0$. So a proper bracket $[a, b]$ is one for which $(f(a) > a$ and $f(b) < b)$ or $(f(a) < a$ and $f(b) > b)$. Geometrically, this means the points $(a, f(a))$ and $(b, f(b))$ are on opposite sides of the line $f(x) = x$, analogous to a root-finding bracket where the two points are on opposite sides of the line $f(x) = 0$.
10. Use your code from question 9 to repeat question 1 using the computer, bracketed Steffensen's method, and a tolerance of 10^{-6}. Given that you are looking for a root of $g(x)$, use $f(x) = g(x) + x$ in your call to Steffensen's method. [S] [A]
11. Compare the results of questions 7 and 10. [A]
12. Rewrite the `inverseQuadraticInterpolation` function so that it stops when the (approximated) relative error is less than the tolerance.
13. Use your code from question 12 to repeat question 1 with a tolerance of 10^{-6}. [S] [A]
14. Compare the results of questions 7 and 13. [A]

Answers

x_4**:** In both methods, the candidate x_4 is accepted since in each case, x_4 is within the bracket formed by x_2 and x_3. So, for bracketed secant, $x_4 = 2.1854$, and for bracketed Newton's, $x_4 = 2.1565$. x_1 is upgraded to x_2 in bracketed secant because $g(x_3)$ is negative. $g(x_2)$ and $g(x_3)$ must have opposite signs in order to maintain the bracket. x_1 is not upgraded in bracketed Newton's because $g(x_3)$ is positive.

Chapter 3

Interpolation

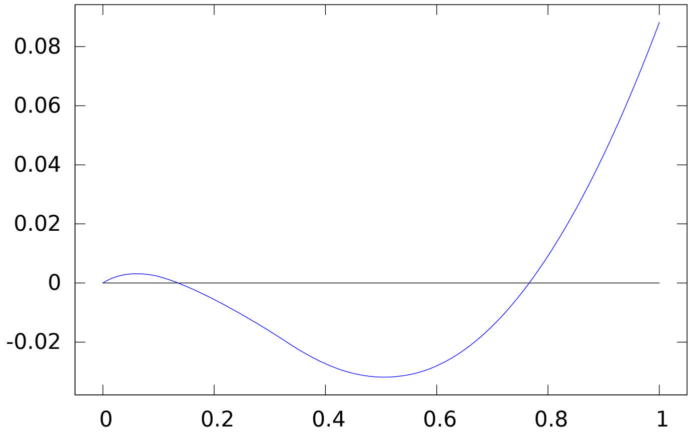
3.1 A root-finding challenge

We open this chapter by combining its content with that of the previous chapter. In the present chapter, we will discuss interpolating functions (functions whose graphs must contain a prescribed set of points) and interpolation (the exercise of finding such a function). In the previous chapter, we discussed approximating roots of functions by numerical computation. Putting these ideas together in the present section, we present an interpolating function, which we will call f, and challenge the reader to find all 6 roots of f, f', and a particular antiderivative of f as accurately and efficiently as possible. Graphs of the three functions and the definition of f follow. Should you accept the challenge, be prepared to use all of what you know about root-finding with computer code. This problem is not easily solved!

If you would like to get right to it, you can skip most of the content of this section. Use the three graphs and the computer code as a starting point to find the roots of F, f, and, f'. The rest of the material is here to help you understand the definition and construction of the functions, but is not prerequisite to taking the challenge.

The function f and its antiderivative

The function



,

which we will call F, could easily be mistaken for a cubic or higher degree polynomial, but it is far from so nice. First, its domain is the interval $[0, 1]$, so the graph shown is the entire graph. Second, it has but two derivatives. Third, its definition is a touch unusual. More on that soon.

What we have here is the antiderivative of a fractal interpolating function. An interpolating function is a function that contains a set of prescribed points. This one happens to be fractal in nature, thus a *fractal* interpolating function. The fractal interpolating function, f, passes through

$$(0, .123), \ (.33, -.123), \ \text{and} \ (1, .5) \tag{3.1.1}$$

in such a way that the graph shown is that of its antiderivative. The unusual nature of the definition of F is derived from the unusual nature of the definition of f:

$$f(x) = \begin{cases} f_1 + c_1\frac{x}{\alpha} + d_1 f\left(\frac{x}{\alpha}\right), & 0 \le x \le \alpha \\ f_2 + c_2\frac{x-\alpha}{1-\alpha} + d_2 f\left(\frac{x-\alpha}{1-\alpha}\right), & \alpha \le x \le 1 \end{cases}$$

where

$$f_1 = \frac{8979}{100000},\ c_1 = -\frac{34779}{100000},\ d_1 = \frac{27}{100}$$
$$f_2 = -\frac{75891}{550000},\ c_2 = \frac{317391}{550000},\ d_2 = \frac{67}{550}$$
$$\alpha = \frac{33}{100}.$$

Crumpet 18: Fractal Interpolating Functions

Fractal interpolating functions are not restricted to passing through three points. Actually, three is the minimum. In general, for $n \ge 3$, suppose $x_1 < x_2 < \cdots < x_n$. The linear fractal interpolating function (there are other types of fractal interpolating functions) passing through each of the points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

and having domain $[x_1, x_n]$ is defined by the linear transformations

$$L_i\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & 0 \\ c_i & d_i \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}, \quad i = 1, 2, \ldots, n-1.$$

The a_i, c_i, e_i, and f_i are calculated based on the requirement that the function interpolate the given points. In particular, we require

$$L_i\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \text{ and } L_i\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}.$$

The d_i are free parameters with the restriction $|d_i| < 1$. It is a straightforward algebraic exercise to show

$$\begin{aligned} a_i &= \frac{x_{i+1} - x_i}{x_n - x_1} \\ c_i &= \frac{y_{i+1} - y_i - d_i(y_n - y_1)}{x_n - x_1} \\ e_i &= x_i - a_i x_1 \\ f_i &= y_i - c_i x_1 - d_i y_1. \end{aligned}$$

In concert, the L_i define the function f, each L_i responsible for the subset $[x_i, x_{i+1}]$ of the domain. $L_i\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i x + e_i \\ c_i x + d_i y + f_i \end{pmatrix}$, so as L_i takes x to $a_i x + e_i$, it simultaneously takes y to $c_i x + d_i y + f_i$. Noting that L_i takes this action on the function f, we must have that $f(a_i x + e_i) = c_i x + d_i f(x) + f_i$ on $[x_1, x_n]$, or equivalently,

$$f(x) = f_i + c_i\left(\frac{x - e_i}{a_i}\right) + d_i f\left(\frac{x - e_i}{a_i}\right) \text{ on } [x_i, x_{i+1}].$$

Putting all the pieces together, f is defined by

$$f(x) = \begin{cases} h_1(x), & x_1 \le x \le x_2 \\ h_2(x), & x_2 \le x \le x_3 \\ & \vdots \\ h_{n-1}(x), & x_{n-1} \le x \le x_n \end{cases}$$

where

$$h_i(x) = f_i + c_i\left(\frac{x - e_i}{a_i}\right) + d_i f\left(\frac{x - e_i}{a_i}\right).$$

Consequently, $F(x) = \int_{x_1}^{x} f(t)\,dt$ is defined by

$$F(x) = \begin{cases} \int_{x_1}^{x} h_1(t)dt, & x_1 \le x \le x_2 \\ F(x_2) + \int_{x_2}^{x} h_2(t)dt, & x_2 \le x \le x_3 \\ & \vdots \\ F(x_{n-1}) + \int_{x_{n-1}}^{x} h_{n-1}(t)dt, & x_{n-1} \le x \le x_n \end{cases}$$

without qualification, and $f'(x)$ is defined by

$$f'(x) = \begin{cases} h_1'(x), & x_1 \le x \le x_2 \\ h_2'(x), & x_2 < x \le x_3 \\ & \vdots \\ h_{n-1}'(x), & x_{n-1} < x \le x_n \end{cases}$$

as long as f' exists! If $\left|\frac{d_i}{a_i}\right| < 1$ for all i, then the derivative will exist almost everywhere, but will generally be discontinuous. If we also have $h_i'(x_{i+1}) = h_{i+1}'(x_{i+1})$ for all $i = 1, 2, \ldots, n-2$, then the derivative will exist and will be continuous.

Reference [2, Chapter 6]

The definition of f is self-referential. Its values are defined by, among other terms, values of itself! This makes evaluating the function a bit different from evaluating a typical function. For example, by virtue of the fact that f passes through the points 3.1.1, we must have $f(0) = .123$, $f(.33) = -.123$, and $f(1) = .5$, facts we can check easily enough. According to the definition,

$$f(0) = f_1 + d_1 f(0) = .08979 + .27 f(0)$$

so $f(0)$ is defined in part by itself. We need to solve the equation $f(0) = .08979 + .27f(0)$ to find $f(0)$. Thus we have $f(0) = \frac{.08979}{.73} = .123$, as promised. Again according to the definition,

$$f(1) = f_2 + c_2 + d_2 f(1) = -\frac{75891}{550000} + \frac{317391}{550000} + \frac{67}{550} f(1).$$

Solving for $f(1)$, we have $f(1) = \frac{-\frac{75891}{550000} + \frac{317391}{550000}}{1 - \frac{67}{550}} = \frac{1}{2}$, as promised. Since $\alpha = .33$, the definition actually gives two ways to calculate $f(.33)$. According to the first part of f,

$$\begin{aligned} f(.33) = f(\alpha) &= f_1 + c_1 + d_1 f(1) \\ &= \frac{8979}{100000} - \frac{34779}{100000} + \frac{27}{100} \cdot \frac{1}{2} \\ &= -.123. \end{aligned}$$

Now is a good time to verify that $f(\alpha) = -.123$ according to the second part of f as well. Try it! Calculating other values of f can be a bit more challenging, but there are still a few that are not so bad. $\alpha^2 < \alpha$ and $\alpha + (1-\alpha)\alpha > \alpha$, so

$$\begin{aligned} f(\alpha^2) &= f_1 + c_1\alpha + d_1 f(\alpha) \\ &= \frac{8979}{100000} - \frac{34779}{100000} \cdot \frac{33}{100} + \frac{27}{100} \cdot \left(-\frac{123}{1000}\right) \\ &= -.0581907 \\ f(\alpha + (1-\alpha)\alpha) &= f_2 + c_2\alpha + d_2 f(\alpha) \\ &= -\frac{75891}{550000} + \frac{317391}{550000} \cdot \frac{33}{100} + \frac{67}{550} \cdot \left(-\frac{123}{1000}\right) \\ &= \frac{2060703}{55000000} \\ &= .037467\overline{327} \end{aligned}$$

With a similar level of difficulty, you can now calculate

$$f(\alpha^3),\ f(\alpha(\alpha+(1-\alpha)\alpha)),\ f(\alpha+(1-\alpha)\alpha^2),\\ \text{and } f(\alpha+(1-\alpha)(\alpha+(1-\alpha)\alpha)).$$

Answers on the next page. More generally, once you have calculated $f(x)$ for some value x, you can then calculate $f(\alpha x)$ and $f(\alpha+(1-\alpha)x)$ from it.

Now that we have a handle on f, we define F by $F(x) = \int_0^x f(t)\,dt$ for all $x \in [0,1]$. Integrating $f(x)$ we have

$$F(x) = \begin{cases} f_1 x + \frac{c_1 x^2}{2\alpha} + \alpha d_1 F\left(\frac{x}{\alpha}\right), & 0 \le x \le \alpha \\ F(\alpha) + f_2\,(x-\alpha) + \frac{c_2(x-\alpha)^2}{2(1-\alpha)} + (1-\alpha)d_2 F\left(\frac{x-\alpha}{1-\alpha}\right), & \alpha \le x \le 1 \end{cases}$$

where again both formulas are applicable when $x = \alpha$. Just like f, F is self-referential. We must go through the same process in finding values of F as we did finding values of f. To get started, $F(0) = \alpha d_1 F(0) \Rightarrow (1-\alpha d_1)\cdot F(0) = 0$, but α and d_1 are both less than 1, so $1-\alpha d_1 \neq 0$. Therefore,

$$F(0) = \frac{0}{1-\alpha d_1} = 0.$$

We could have computed this value by integration just as well: $F(0) = \int_0^0 f(t)\,dt = 0$. Now, according to the formula,

$$F(1) = F(\alpha) + (1-\alpha)\left(f_2 + \frac{c_2}{2} + d_2 F(1)\right)$$
$$\text{and}$$
$$F(\alpha) = \alpha\left(f_1 + \frac{c_1}{2} + d_1 F(1)\right),$$

a system of two equations in the two unknowns, $F(\alpha)$ and $F(1)$. Its solution is

$$\begin{aligned} F(\alpha) &= -\frac{121012947}{6081400000} \approx -.01989886325517151 \\ F(1) &= \frac{5361861}{60814000} \approx 0.0881682014009932. \end{aligned}$$

Now that we have the few values, $F(0)$, $F(\alpha)$, and $F(1)$, we can calculate others as before. The values $F(\alpha x)$ and $F(\alpha+(1-\alpha)x)$ will both depend on the value of $F(x)$. So we can compute $F(\alpha^2)$ and $F(\alpha+(1-\alpha)\alpha)$:

$$\begin{aligned} F(\alpha^2) &= f_1\alpha^2 + \frac{c_1\alpha^3}{2} + \alpha d_1 F(\alpha) \\ &= \frac{10678194456039}{6081400000000000} \\ &\approx .001755877668964219 \\ F(\alpha+(1-\alpha)\alpha) &= F(\alpha) + f_2\,(1-\alpha)\,\alpha + \frac{c_2\,(1-\alpha)\,\alpha^2}{2} + (1-\alpha)d_2 F(\alpha) \\ &= -\frac{94196657189979}{3040700000000000} \\ &\approx -.03097860926430723. \end{aligned}$$

Now you can calculate $F(\alpha^3)$, $F(\alpha(\alpha+(1-\alpha)\alpha))$, $F(\alpha+(1-\alpha)\alpha^2)$, and $F(\alpha+(1-\alpha)(\alpha+(1-\alpha)\alpha))$ yourself. Answers on the facing page. You shouldn't worry about calculating these values exactly. That would require a computer algebra system with arbitrary precision and is not really the point. The point is to make sure you understand how to do the calculations. Use a calculator or the computer and the approximate values already calculated.

The derivative of f and more graphs

The function f has a continuous derivative. In fact, the parameters defining f were specifically chosen so the derivative would exist and be continuous. Differentiating f gives us

$$f'(x) = \begin{cases} \frac{c_1}{\alpha} + \frac{d_1}{\alpha} f'\left(\frac{x}{\alpha}\right), & 0 \le x \le \alpha \\ \frac{c_2}{1-\alpha} + \frac{d_2}{1-\alpha} f'\left(\frac{x-\alpha}{1-\alpha}\right), & \alpha \le x \le 1 \end{cases}$$

Figure 3.1.1: Graph of f.

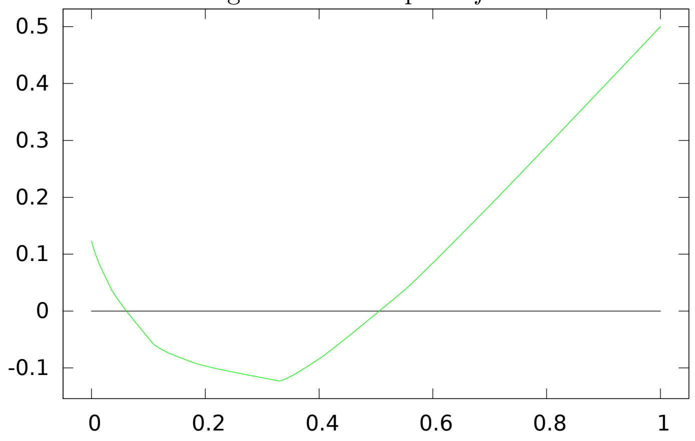
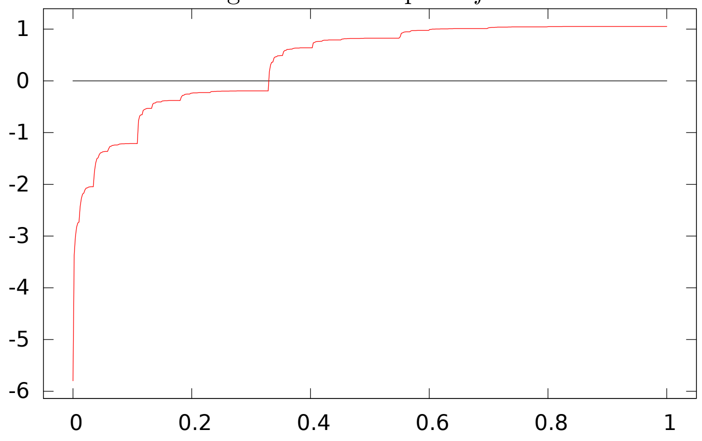


Figure 3.1.2: Graph of f'.



and we can check as before that the definition is consistent when $x = \alpha$:

$$\begin{aligned}
f'(0) = \frac{c_1}{\alpha} + \frac{d_1}{\alpha} f'(0) \Rightarrow f'(0) = \frac{c_1}{\alpha - d_1} = -\frac{11593}{2000} = -5.7965 \\
f'(1) = \frac{c_2}{1-\alpha} + \frac{d_2}{1-\alpha} f'(1) \Rightarrow f'(1) = \frac{c_2}{1-\alpha-d_2} = \frac{105797}{100500} \approx 1.052706467661692 \\
f'(\alpha) = \frac{c_1}{\alpha} + \frac{d_1}{\alpha} f'(1) = -\frac{141949}{737000} \approx -.1926037991858887 \\
f'(\alpha) = \frac{c_2}{1-\alpha} + \frac{d_2}{1-\alpha} f'(0) = -\frac{141949}{737000} \approx -.1926037991858887.
\end{aligned}$$

Other values of f' can be computed as done for f and F. The graphs of f and f' are shown in Figures 3.1.1 and 3.1.2.

That's it. Now see if you can find the roots of the three functions.

Answers

Evaluating f: The following are a few values of f:

$$\begin{array}{rcl}
f(\alpha^3) & \approx & .03620418000000000 \\
f(\alpha(\alpha + (1-\alpha)\alpha)) & \approx & -.09176089063636364 \\
f(\alpha + (1-\alpha)\alpha^2) & \approx & -.08222890363636364 \\
f(\alpha + (1-\alpha)(\alpha + (1-\alpha)\alpha)) & \approx & .1846063473223140.
\end{array}$$

Evaluating F: The following are a few values of F:

$$\begin{aligned}
F(\alpha^3) &\approx .002702687013731212 \\
F(\alpha(\alpha+(1-\alpha)\alpha)) &\approx -.003859289400223274 \\
F(\alpha+(1-\alpha)\alpha^2) &\approx -.02753062961856850 \\
F(\alpha+(1-\alpha)(\alpha+(1-\alpha)\alpha)) &\approx -.01466250212441314.
\end{aligned}$$

3.2 Lagrange Polynomials

A function that is required to have a graph passing through some set of prescribed points is called an interpolating function, and we say that such a function interpolates the prescribed points. Further, the exercise of finding such a function is called interpolation.

In exercise 3a of section 2.5, you are asked to find a polynomial with roots at $-7, 2$, and $1 \pm 5i$ (and no others). The function, therefore, must be a polynomial and have a graph passing through the points

$$(-7,0),\ (2,0),\ (1+5i,0),\ \text{and}\ (1-5i,0). \tag{3.2.1}$$

In retrospect, then, the question could have been phrased as: find a polynomial passing through the points 3.2.1 (and not having any roots besides -7, 2, $1+5i$, and $1-5i$), a question of interpolation. We now expand upon this idea by considering polynomials with graphs passing through points with arbitrary ordinates (not just 0).

We start on familiar ground. The polynomial $p(x) = (x+7)(x-2)$ has roots -7 and 2 so has a graph passing through $(-7,0)$ and $(2,0)$. Suppose we want to modify p so it also passes through $(-1,1)$. That is, we want $p(-7) = 0$, $p(-1) = 1$, and $p(2) = 0$. Beginning with $p(x) = (x+7)(x-2)$, we already have $p(-7) = 0$ and $p(2) = 0$, so really we only need to concentrate on $p(-1) = 1$. As is, $p(-1) = (-1+7)(-1-2) = 6(-3) = -18$, a far cry from 1. But $p(x) = (x+7)(x-2)$ is not the only polynomial passing through $(-7,0)$ and $(2,0)$. Let a be any real number and note that $q(x) = a(x+7)(x-2)$ also passes through $(-7,0)$ and $(2,0)$. If we choose a such that $q(-1) = 1$, we have the desired function:

$$q(-1) = a(-1+7)(-1-2) = -18a = 1 \Rightarrow a = -\frac{1}{18}.$$

$q(x) = -\frac{1}{18}(x+7)(x-2)$ passes through all three of the points, $(-7,0)$, $(2,0)$, and $(-1,1)$. But let us not lose sight of whence this came. $-\frac{1}{18} = \frac{1}{p(-1)}$, so, actually, the desired function can be written as $q(x) = \frac{p(x)}{p(-1)}$. Indeed, $q(-7) = \frac{p(-7)}{p(-1)} = 0$, $q(2) = \frac{p(2)}{p(-1)} = 0$, and $q(-1) = \frac{p(-1)}{p(-1)} = 1$.

Now suppose we want a polynomial passing through $(-7,0)$, $(2,0)$, and $(-1,\sqrt{2})$. As before, we know $p(x) = (x+7)(x-2)$ has the desired roots and $q(x) = \frac{p(x)}{p(-1)}$ has the nice feature that $q(-1) = 1$. We use these two facts to come up with an answer. In fact, without doing any calculation, we know the polynomial

$$l(x) = \frac{p(x)}{p(-1)}\sqrt{2}$$

is the desired function. Take a moment to check that $l(-7) = 0$, $l(2) = 0$, and $l(-1) = \sqrt{2}$, and understand its construction. This idea is the seed for what is called the Lagrange form of interpolating polynomials.

We are now ready to let the ordinates fly! Suppose we would like a polynomial passing through $(-7, y_1)$, $(2, y_2)$, and $(-1, y_3)$. We know the polynomial $p_3(x) = (x+7)(x-2)$ has zeros at -7 and 2, so the polynomial $l_3(x) = \frac{p_3(x)}{p_3(-1)} y_3$ has zeros at -7 and 2 and, conveniently, $l_3(-1) = y_3$. This is a good first step. It has the correct ordinate at -1 and zeros at -7 and 2. Similarly, we can construct the polynomial $p_2(x) = (x+7)(x+1)$ with zeros at -7 and -1, from which we can construct the polynomial $l_2(x) = \frac{p_2(x)}{p_2(2)} y_2$ with zeros at -7 and -1 and, conveniently, $l_2(2) = y_2$. This is a good second step. It has the correct ordinate at 2 and zeros at -7 and -1. Now consider the sum $(l_3 + l_2)$. $l_3(-1) = y_3$ and $l_2(-1) = 0$, so $(l_3 + l_2)(-1) = y_3$. Similarly, $l_3(2) = 0$ and $l_2(2) = y_2$, so $(l_3 + l_2)(2) = y_2$. Moreover, $(l_3 + l_2)(-7) = 0$. We now have a polynomial passing through two of the three required points and having a zero at the abscissa of the third point. If we had a polynomial with the correct ordinate at -7 and zeros at 2 and -1, we could add it to the sum and be done. But this is exactly the type of polynomial we have been constructing! We let $p_1(x) = (x+1)(x-2)$ and $l_1(x) = \frac{p_1(x)}{p_1(-7)} y_1$, and note that l_1 has the correct ordinate at -7 and zeros at 2 and -1, just as we needed. Finally, the desired polynomial is $(l_1 + l_2 + l_3)$. Table 3.1 summarizes the construction.

And now we are ready for complete generalization. Suppose $n \geq 1$ and $x_0, x_1, \ldots, x_n$ are n distinct real numbers. We use the notation $P_n(x)$ for the polynomial of least degree interpolating the points

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n).$$

Setting $p_i(x) = \displaystyle\prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j) = (x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)$, one formula for P_n is

$$L_n(x) = \sum_{i=0}^{n} \frac{p_i(x)}{p_i(x_i)} y_i. \tag{3.2.2}$$

Table 3.1: A polynomial passing through $(-7, y_1)$, $(2, y_2)$, and $(-1, y_3)$.

x	$l_1(x) = \frac{p_1(x)}{p_1(-7)} y_1$	$l_2(x) = \frac{p_2(x)}{p_2(2)} y_2$	$l_3(x) = \frac{p_3(x)}{p_3(-1)} y_3$	$(l_1 + l_2 + l_3)(x)$
-7	y_1	0	0	y_1
2	0	y_2	0	y_2
-1	0	0	y_3	y_3

As written, L_n is called the *Lagrange form* of P_n. For sake of brevity, it is often called the Lagrange interpolating polynomial, or even Lagrange polynomial. However, the interpolating polynomial of least degree by any other name would be but P_n. We will adhere to the practice of calling it the interpolating polynomial of least degree, or use the notation P_n, when the form is unimportant and will add the phrase *Lagrange form*, or use the notation L_n, when it is.

The main use for interpolating polynomials in numerical analysis is to approximate non-polynomial functions in the following way. Suppose we know the value of f at a selection of points. That is, we know $f(x_0) = y_0, f(x_1) = x_1, \ldots, f(x_n) = y_n$ and perhaps not much more. The interpolating polynomial of least degree passing through the $n+1$ points

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

will, by construction, agree with f at $x_0, x_1, \ldots, x_n$ and we can say with some precision how closely this interpolating polynomial agrees with f at other points as well. The values of the interpolating polynomial at these "other points" are what we refer to as approximations of the non-polynomial function.

Setting $a = \min(x_0, \ldots, x_n, x)$ and $b = \max(x_0, \ldots, x_n, x)$, we have the following result. If f has $n+1$ derivatives on (a, b) and $f, f', f'', \ldots, f^{(n)}$ are all continuous on $[a, b]$, then there is a value $\xi_x \in (a, b)$ such that

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n). \tag{3.2.3}$$

Ironically, this result is proven by considering the Lagrange form of an interpolating polynomial in t that is equal to the error at x and equal to zero at each x_i. That polynomial is

$$\Lambda(t) = [P_n(x) - f(x)] \frac{(t - x_0)(t - x_1)\cdots(t - x_n)}{(x - x_0)(x - x_1)\cdots(x - x_n)}.$$

Crumpet 19: Λ

Λ is the (capital) eleventh letter of the Greek alphabet and is pronounced `lam-duh`. The lower case version, λ, appears much more commonly in mathematics and often represents an eigenvalue.

Subtracting this polynomial from the error, $e(t) = P_n(t) - f(t)$, we have a function,

$$g(t) = e(t) - \Lambda(t),$$

that is zero for all $t = x_0, x_1, \ldots, x_n, x$. Since $g, g', \ldots, g^{(n)}$ are all continuous on $[a, b]$ and $g^{(n+1)}$ exists on (a, b), by Generalized Rolle's Theorem, there is a value $\xi_x \in (a, b)$ such that $g^{(n+1)}(\xi_x) = 0$. On the other hand,

$$\begin{aligned} g^{(n+1)}(\xi_x) &= e^{(n+1)}(\xi_x) - \Lambda^{(n+1)}(\xi_x) \\ &= P_n^{(n+1)}(\xi_x) - f^{(n+1)}(\xi_x) - \Lambda^{(n+1)}(\xi_x), \end{aligned}$$

and P_n is a polynomial of degree at most n. Hence, $P_n^{(n+1)}(t) = 0$ for all t and we have $g^{(n+1)}(\xi_x) = -f^{(n+1)}(\xi_x) - \Lambda^{(n+1)}(\xi_x) = 0$. It follows that

$$f^{(n+1)}(\xi_x) = -\Lambda^{(n+1)}(\xi_x).$$

But, Λ is a polynomial of degree $n+1$ in t, so its $(n+1)^{st}$ derivative with respect to t is constant with respect to t. We write Λ as

$$\Lambda(t) = \frac{P_n(x) - f(x)}{(x-x_0)(x-x_1)\cdots(x-x_n)}\left[t^{n+1} + b_n t^n + \cdots + b_0 t^0\right]$$

for some constants $b_n, b_{n-1}, \ldots, b_0$, and consequently,

$$\Lambda^{(n+1)}(t) = \frac{P_n(x) - f(x)}{(x-x_0)(x-x_1)\cdots(x-x_n)} \cdot (n+1)!,$$

and we have, by substitution,

$$f^{(n+1)}(\xi_x) = \frac{f(x) - P_n(x)}{(x-x_0)(x-x_1)\cdots(x-x_n)} \cdot (n+1)!$$

or, equivalently,

$$\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n) = f(x) - P_n(x)$$

as desired.

Figure 3.2.1 shows interpolating polynomials for three different functions. The x-coordinates of the prescribed points are the same for each interpolating polynomial. The x-coordinates are

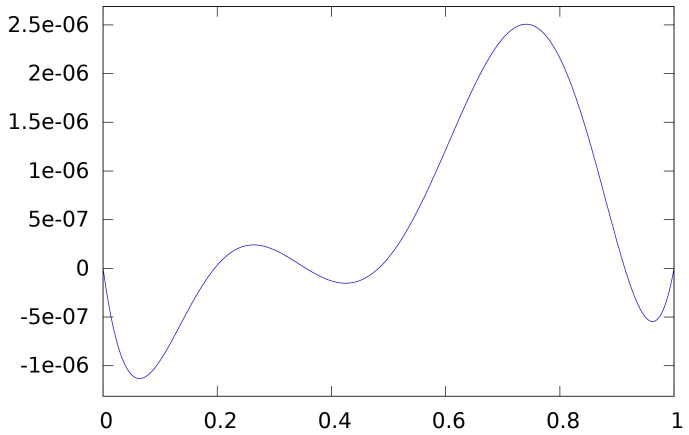
$$0, .1951846177977887, .3554400571592862, .4823905248516196, .9138095996128959, \text{ and } 1.$$

The four numbers between 0 and 1 were selected by a random number generator. The interpolating polynomial closely resembles the function only in the first case. The sixth derivative of f helps explain why.

Our error term,

$$\frac{f^{(6)}(\xi)}{6!}(x-x_0)(x-x_1)\cdots(x-x_5)$$

implies that the sixth derivative of f and the polynomial $h(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_5)}{6!}$ determine how much f and L_6 will differ. By bounding both $\left|f^{(6)}\right|$ and $|h|$ over the interval $[0,1]$, we can get a bound on the difference between f and L_6. The graphs of $f^{(6)}$ are shown in Figure 3.2.1. The graph of h is

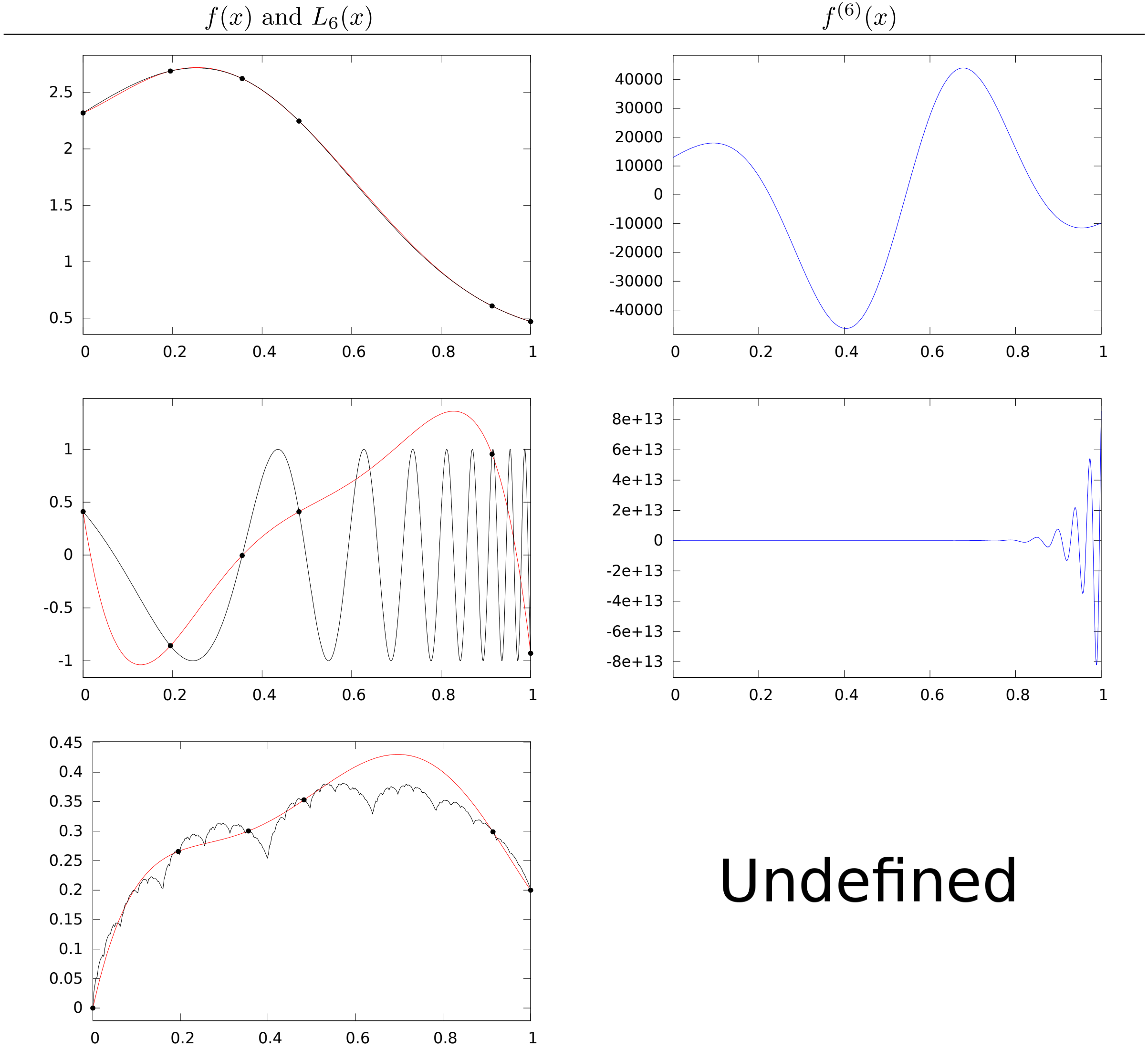


so $\max_{x\in[0,1]}|h(x)|$ occurs around 0.75. We can use a root-finding method applied to h' to find that the maximum of $|h|$ is approximately $h(.7409254943919) \approx 2.506891519629(10)^{-6}$, a relatively small number. On the other hand, for $f(x) = e^{\sin\left((x+1)^2\right)}$, we find $\max_{x\in[0,1]}\left|f^{(6)}(x)\right| \approx f^{(6)}(.6777170541644) \approx 44013.74605321$, a relatively large number. Their product,

$$\max_{x\in[0,1]}|h(x)| \cdot \max_{x\in[0,1]}\left|f^{(6)}(x)\right| \approx .11,$$

gives a bound on the error. The absolute furthest L_6 can be from f over the interval $[0,1]$ is 0.11, a relatively small number. The actual error is considerably smaller, so can barely be noticed in the top left graph of Figure 3.2.1.

Figure 3.2.1: Three interpolating functions. From top to bottom, $e^{\sin\left((x+1)^2\right)}$, $\sin\left(e^{(x+1)^2}\right)$, and a fractal function as defined in section 3.1. f is shown in black and the interpolant, L_6, in red.



For $f(x) = \sin\left(e^{(x+1)^2}\right)$, we find $\max_{x\in[0,1]}\left|f^{(6)}(x)\right| \approx f^{(6)}(1) \approx 8.552147927657737(10)^{13}$, a relatively large number. This time the product,

$$\max_{x\in[0,1]}|h(x)| \cdot \max_{x\in[0,1]}\left|f^{(6)}(x)\right| \approx 2.14393071144600004(10)^{8},$$

is a huge number relative to the values of f. So the theoretical error bound does not predict good results for this interpolation. In fact, it suggests that the interpolation could have been much, much worse! L_6 might have differed from f by over 2 million, a fact that should be worrisome considering f takes values between -1 and 1. An approximation that is off by even 1 is completely useless for this particular f. As it is, we should not be surprised that L_6 is not a good approximation of f since the error term can be quite large. Nonetheless, the method is sound. Failure to approximate f well should not be seen as a flaw in the method, but rather a flaw in its application. If we really wanted to approximate f well, we would need to find a different set of points over which to interpolate.

For the fractal function in the bottom left of Figure 3.2.1, our error estimate is entirely irrelevant. The sixth derivative of f does not exist. In fact, even the first derivative of f does not exist. We have no way to estimate the error except to look at the graphs. And as we see, L_6 again does a very poor job of approximating f. Failure, again, should not be seen as a flaw in the method, but rather in its application. Approximating a function with an interpolating polynomial presumes that the function has sufficient derivatives.

Crumpet 20: Bernstein polynomials

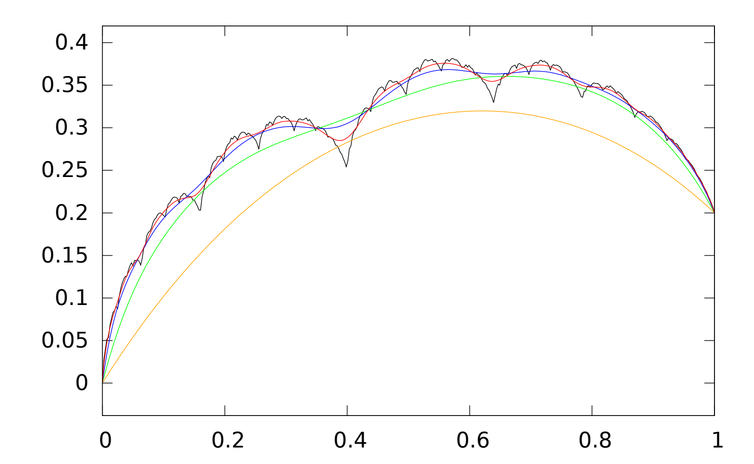
Suppose f is a continuous function on the interval $[0, 1]$, and define the polynomial

$$B_n(x) = \sum_{\nu=0}^{n} \binom{n}{\nu} f\left(\frac{\nu}{n}\right) x^{\nu}(1-x)^{n-\nu}, \quad n = 1, 2, 3, \ldots$$

Then

$$\lim_{n\to\infty} B_n(x) = f(x)$$

uniformly. That is, $\lim_{n\to\infty} \max\{|B_n(x) - f(x)| : x \in [0, 1]\} = 0$. The B_n are Bernstein polynomials. Shown below are B_4, B_{20}, B_{100}, and B_{500} for the fractal function in figure 3.2.1.



An application of interpolating polynomials

Again we find ourselves connecting the content of the previous chapter with that of the current. The secant method is actually an application of interpolating polynomials to root-finding. The secant line whose slope is used to calculate any given iteration can be viewed as an interpolating line! It passes through two points lying on g. Hence, it is an approximation of g.

Having taken this point of view, we can now imagine generalizing the method by using the derivative of a higher degree interpolating polynomial to approximate g' at each step. Such a generalized method, which we will call Sidi's k^{th} degree method [30], is summarized by the formula

$$x_{n+1} = x_n - \frac{g(x_n)}{p'_{n,k}(x_n)}$$

where $p_{n,k}$ is the interpolating polynomial passing through the points

$$(x_n, g(x_n)), (x_{n-1}, g(x_{n-1})), \ldots, (x_{n-k}, g(x_{n-k})).$$

When $k = 1$, this is exactly the secant method. When $k = 2$, this method uses the same parabola as does Müller's method, but in a different way. In Müller's method, the next iteration is found by locating a root of the interpolating polynomial. In this method, the next iteration is found by locating a root of a tangent line to the interpolating polynomial.

As k increases, more initial values are needed, but the order of convergence increases as a benefit. Letting α_k be the order of convergence of Sidi's k^{th} degree method, we have $\alpha_1 = \frac{1+\sqrt{5}}{2} \approx 1.618$, the order of convergence of the secant method, and

$$\alpha_2 \approx 1.839,\ \alpha_3 \approx 1.928,\ \alpha_4 \approx 1.966.$$

For any k, Sidi's method has an order of convergence less than 2 (the order of convergence of Newton's method) but it approaches 2 as k increases.

At this point, you might wonder just how practical such a method might be. After all, calculating a new Lagrange interpolating polynomial and evaluating its derivative at each step can be a cumbersome process. We will take up this issue in the next section.

Neville's Method

The Lagrange form of an interpolating polynomial is as convenient as it gets for a human. With a little care and patience, it is possible to write down such a polynomial without even the aid of a calculator. However, adding points to the interpolation and evaluating the polynomial for non-interpolated points can be cumbersome tasks. Consider a simple example: the polynomial interpolating $f(x) = e^x$ at $x = 0, 1, 2$:

$$\begin{aligned} L_2(x) &= \frac{(x-1)(x-2)}{(0-1)(0-2)}e^0 + \frac{(x-0)(x-2)}{(1-0)(1-2)}e^1 + \frac{(x-0)(x-1)}{(2-0)(2-1)}e^2 \\ &= \frac{(x-1)(x-2)}{2} + \frac{x(x-2)}{-1}e + \frac{x(x-1)}{2}e^2. \end{aligned}$$

Evaluating $L_2(1.5)$, for example, requires either

1. computing the values of the three separate terms, each a quadratic polynomial, and adding:

$$\begin{aligned} L_2(1.5) &= \frac{(1.5-1)(1.5-2)}{2} + \frac{1.5(1.5-2)}{-1}e + \frac{1.5(1.5-1)}{2}e^2 \\ &= -.125 + .75e + .375e^2 \\ &\approx 4.684607408443278 \end{aligned}$$

   or

2. the unpleasant business of simplifying L_2 into a simpler form and then evaluating:

$$\begin{aligned} L_2(x) &= \frac{(x-1)(x-2)}{2} + \frac{x(x-2)}{-1}e + \frac{x(x-1)}{2}e^2 \\ &= \frac{1}{2}(x^2-3x+2) - e(x^2-2x) + \frac{e^2}{2}(x^2-x) \\ &= \left(\frac{1}{2} - e + \frac{e^2}{2}\right)x^2 + \left(-\frac{3}{2} + 2e - \frac{e^2}{2}\right)x + 1 \\ &\approx 1.47624622100628x^2 + 0.242035607452765x + 1 \end{aligned}$$

   so $L_2(1.5) \approx 1.47624622100628(1.5)^2 + 0.242035607452765(1.5) + 1 = 4.684607408443277$.

Method 2 is better if you have more points at which to evaluate, and method 1 is better if you plan to add points of interpolation. However, neither method is particularly convenient. Even less convenient than evaluating the polynomial is the task of requiring another point of interpolation. Previous work is of limited use. And we haven't even begun to discuss the trouble of writing a computer program to automate the calculations. Neville's method can be used to overcome these limitations when the value of the polynomial at a specific point is required.

Neville's method is based on the observation that interpolating polynomials can be constructed recursively. Suppose $P_{k,l}$ is the polynomial of degree at most l interpolating the data

$$(x_k, f(x_k)), (x_{k+1}, f(x_{k+1})), \ldots, (x_{k+l}, f(x_{k+l})).$$

Then, by definition, $P_{0,n}$ is the polynomial of degree at most n interpolating the data

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n)).$$

Moreover, $P_{0,n}$ can be computed using the recursive formula

$$\begin{aligned} P_{i,m+1}(x) &= \frac{(x - x_{i+m+1})P_{i,m}(x) - (x - x_i)P_{i+1,m}(x)}{x_i - x_{i+m+1}} \\ P_{i,0}(x) &= f(x_i), \qquad i = 0, \ldots, n. \end{aligned} \tag{3.2.4}$$

This claim can be checked by noting five things:

1. $P_{i,0}$ is the degree 0 polynomial interpolating the one datum $(x_i, f(x_i))$.

2. $P_{i,m}$ and $P_{i+1,m}$ are polynomials of degree at most m, so $P_{i,m+1}$ is a polynomial of degree at most $m+1$.

3. $P_{i,m+1}(x_i) = \dfrac{(x_i - x_{i+m+1})P_{i,m}(x_i)}{x_i - x_{i+m+1}} = P_{i,m}(x_i) = f(x_i)$.

4. For any $j = i+1, \ldots, i+m$,

$$\begin{aligned} P_{i,m+1}(x_j) &= \frac{(x_j - x_{i+m+1})P_{i,m}(x_j) - (x_j - x_i)P_{i+1,m}(x_j)}{x_i - x_{i+m+1}} \\ &= \frac{(x_j - x_{i+m+1})f(x_j) - (x_j - x_i)f(x_j)}{x_i - x_{i+m+1}} \\ &= \frac{f(x_j)\left[(x_j - x_{i+m+1}) - (x_j - x_i)\right]}{x_i - x_{i+m+1}} \\ &= f(x_j). \end{aligned}$$

5. $P_{i,m+1}(x_{i+m+1}) = \dfrac{-(x_{i+m+1} - x_i)P_{i+1,m}(x_{i+m+1})}{x_i - x_{i+m+1}} = P_{i+1,m}(x_{i+m+1}) = f(x_{i+m+1})$.

A rigorous proof by induction on m, requested in the exercises, should follow closely these notes. Points 1 and 2 establish that $P_{k,l}$ has degree at most l. Points 3 through 5 establish that $P_{k,l}$ interpolates the points $(x_k, f(x_k)), (x_{k+1}, f(x_{k+1})), \ldots, (x_{k+l}, f(x_{k+l}))$. Formula 3.2.4 succinctly summarizes Neville's method.

While Neville's method (formula 3.2.4) can be used to find formulas for interpolating polynomials as in

$$\begin{aligned} P_{0,1}(x) &= \frac{(x - x_1)P_{0,0}(x) - (x - x_0)P_{1,0}(x)}{x_0 - x_1} \\ &= \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1), \end{aligned}$$

it is normally used to find the value of an interpolating polynomial at a specific point. We earlier determined that $L_2(1.5) = 4.684607408443277$ for the polynomial, $L_2(x)$, interpolating $f(x) = e^x$ at $x = 0, 1, 2$. We now find this value using Neville's method. $P_{0,0}(1.5) = f(0) = 1$, $P_{1,0}(1.5) = f(1) \approx 2.718281828459045$, and $P_{2,0}(1.5) = f(2) \approx$

Table 3.2: Neville's method example, calculating $P_{0,2}(1.5)$.

x_i	$P_{i,0} = f(x_i)$	$P_{i,1}$	$P_{i,2}$
0	1	3.577422742688568	4.684607408443278
1	2.718281828459045	5.053668963694848	
2	7.38905609893065		

7.38905609893065. So

$$\begin{aligned}
P_{0,1}(1.5) &= \frac{(1.5 - x_1)P_{0,0}(1.5) - (1.5 - x_0)P_{1,0}(1.5)}{x_0 - x_1} \\
&= \frac{(1.5-1)(1) - (1.5-0)(2.718281828459045)}{0-1} \\
&\approx 3.577422742688568 \\
P_{1,1}(1.5) &= \frac{(1.5 - x_2)P_{1,0}(1.5) - (1.5 - x_1)P_{2,0}(1.5)}{x_1 - x_2} \\
&= \frac{(1.5-2)(2.718281828459045) - (1.5-1)(7.38905609893065)}{1-2} \\
&\approx 5.053668963694848 \\
P_{0,2}(1.5) &= \frac{(1.5 - x_2)P_{0,1}(1.5) - (1.5 - x_0)P_{1,1}(1.5)}{x_0 - x_2} \\
&= \frac{(1.5-2)(3.577422742688568) - (1.5-0)(5.053668963694848)}{0-2} \\
&\approx 4.684607408443278.
\end{aligned}$$

A tabulation of the computation may make it easier to internalize the recursion and imagine how this process might be automated. Table 3.2 shows such a tabulation. The use of this recursive formula may be more difficult than direct computation for a human being, but for a computer, using the recursion is much quicker and simpler as evidenced by a look at the pseudo-code.

```
Assumptions: $P_n(x)$ is the degree at most $n$ polynomial interpolating the data

        $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$

    and the value $P_n(\hat{x})$ is desired.
Input: Value $\hat{x}$; abscissas $x_0, x_1, \ldots, x_n$; ordinates $f(x_0), f(x_1), \ldots, f(x_n)$.
Step 1: For $i = 0 \ldots n$ do Step 2:
    Step 2: Set $P_{i,0} = f(x_i)$;
Step 3: For $j = 1 \ldots n$ do Steps 4-5:
    Step 4: For $i = 0 \ldots n - j$ do Step 5:
        Step 5: Set $P_{i,,j} = \frac{(\hat{x}-x_{i+j})P_{i,j-1}-(\hat{x}-x_i)P_{i+1,j-1}}{x_i-x_{i+j}}$
Output: Table of values, $P$. $P_{0,n}$ holds the desired value, $L_n(\hat{x})$.
```

Uniqueness

There are some subtleties we have thus far glossed over. When we introduced the Lagrange form, we casually stated "L_n is called the *Lagrange form* of P_n", implying that the Lagrange form gives the interpolating polynomial *of least degree* (since P_n is defined as such)! This fact is far from obvious. Nonetheless, we went on as if it were obvious that L_n and P_n were one and the same polynomial. Worse yet, when we came around to discussing Neville's method, we calculated $P_{0,2}(1.5)$ and compared it to $L_2(1.5)$ from earlier with the implication that they should be the same, again as if it were simply given that $P_{0,2}$ and L_2 should be the same polynomial. The following result shows that our blind faith that P_n, L_n, and $P_{0,n}$ amount to different names for the same object was not misplaced (by virtue of the fact that they all interpolate the same data and have degree at most n).

Theorem 7. *The polynomial, P_n, of least degree interpolating the data $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ exists and is unique. Moreover, any interpolating polynomial of degree at most n is equal to P_n.*

Proof. By construction, L_n interpolates the data. Moreover, the degree of L_n is at most n since it is the sum of polynomials p_i each with degree exactly n. Thus P_n exists and has degree at most n [at this point, we must admit that the degree of P_n may be less than that of L_n]. Now suppose q is any polynomial interpolating $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ with degree n or less. Then the polynomial $f = P_n - q$ also has degree n or less. Moreover, $f(x_i) = P_n(x_i) - q(x_i) = y_i - y_i = 0$ for all $i - 0, \dots, n$. Thus f has $n+1$ roots. Alas, the only way f can have $n+1$ roots and have degree n or less is if f is identically 0. Hence, $f(x) = P_n(x) - q(x) = 0$, implying $P_n(x) = q(x)$ for all x. □

Key Concepts

Interpolating function: A function whose graph is required to pass through a set of prescribed points.

Interpolating polynomial: A polynomial whose graph is required to pass through a set of prescribed points.

Interpolating polynomial of least degree: The polynomial of least degree interpolating a given set of $n+1$ data points is unique. We denote this polynomial by P_n.

Interpolating polynomial of degree at most n: The polynomial interpolating $n+1$ distinct points has degree at most n and is equal to the polynomial of least degree interpolating the points.

Generalized Rolle's theorem: Suppose that f has n derivatives on (a, b) and $f, f', f'', \dots, f^{(n-1)}$ are all continuous on $[x_0, x_n]$. If $f(x_0) = f(x_1) = \cdots = f(x_n)$ for some $x_0 < x_1 < \cdots < x_n$, then there exists $\xi \in (a, b)$ such that $f^{(n)}(\xi) = 0$.

Lagrange form of an interpolating polynomial: The Lagrange form, L_n, of the polynomial of degree at most n interpolating the points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ is given by the formula

$$L_n(x) = \sum_{i=0}^{n} \frac{p_i(x)}{p_i(x_i)} y_i,$$

where $p_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j) = (x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)$.

Interpolation error: For P_n, the interpolating polynomial of least degree passing through the $n+1$ points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, there is a value $\xi_x \in (a, b)$ such that

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)(x - x_1) \cdots (x - x_n),$$

assuming f has $n+1$ derivatives on (a, b) and $f, f', f'', \dots, f^{(n)}$ are all continuous on $[a, b]$, and where $a = \min(x_0, \dots, x_n, x)$ and $b = \max(x_0, \dots, x_n, x)$.

Sidi's method: A root-finding method summarized by the formula

$$x_{n+1} = x_n - \frac{f(x_n)}{p'_{n,k}(x_n)}$$

where $p_{n,k}$ is the interpolating polynomial passing through the points

$$(x_n, f(x_n)), (x_{n-1}, f(x_{n-1})), \dots, (x_{n-k}, f(x_{n-k})).$$

Neville's method: A method for computing the interpolating polynomial of least degree or values of it based on the recursive relation

$$\begin{aligned} P_{i,m+1}(x) &= \frac{(x - x_{i+m+1})P_{i,m}(x) - (x - x_i)P_{i+1,m}(x)}{x_i - x_{i+m+1}} \\ P_{i,0}(x) &= f(x_i) \end{aligned}$$

where $P_{k,l}$ is the polynomial of least degree interpolating the data

$$(x_k, f(x_k)), (x_{k+1}, f(x_{k+1})), \dots, (x_{k+l}, f(x_{k+l})).$$

Exercises

1. Write down the Lagrange interpolating polynomial passing through $(1, 2)$, $(1.5, -0.83)$, and $(2.11, -1)$.
2. Find a polynomial that passes through the four points

$$(0, 0),\ (1, 2),\ (4, -3),\ \text{and}\ (10, -1).$$

3. Construct the (at most) quadratic Lagrange Polynomial interpolating the data.

   (a) $(1, 1)$, $(2, 1)$, and $(3, 2)$

   (b) $(0, 10)$, $(30, 58)$, $(1029, -32)$

   (c) $(-10, 10)$, $(20, 58)$, $(1019, -32)$ [S]

   (d)

x	$f(x)$
5	15
200	2
10	15

   (e)

x	$f(x)$
-5	15
-2	2
3	15

4. Suppose the data from question 3 were taken from an appropriately differentiable function f. Use the interpolating polynomial you found in question 3 to estimate $f(1.3)$. [S]
5. Find the estimate in question 4 using Neville's method. [S]
6. Given the following data for $f(x)$, approximate $f(0.3)$ using an interpolating polynomial of degree at most

   (a) 1

   (b) 2

   (c) 3

x	0	1	2	3
$f(x)$	0.8	0.7	0.75	0.5

7. Given the following data for $f(x)$, approximate $f(3)$ using an interpolating polynomial of degree at most [S]

   (a) 1

   (b) 2

   (c) 3

x	2	3.5	4	5
$f(x)$	0.8	0.7	0.75	0.5

8. Use interpolating polynomials of degrees one, two, and three to approximate each of the following:

   (a) $f(0.43)$ if $f(0) = 1$, $f(0.25) = 1.64872$, $f(0.5) = 2.71828$, $f(0.75) = 4.48169$.

   (b) $f(0.18)$ if $f(0.1) = -0.29004986$, $f(0.2) = -0.56079734$, $f(0.3) = -0.81401972$, $f(0.4) = -1.0526302$. [S]

   (c) $f(2.26)$ if $f(1) = 1.654$, $f(1.5) = -2.569$, $f(2) = -1.329$, $f(2.5) = 1.776$. [S]

   (d) $f(11.26)$ if $f(10) = -0.7865$, $f(11) = -1.2352$, $f(12) = -0.8765$, $f(13) = 0.0021$.

9. Let $x_0 = 1$, $x_1 = 1.25$, and $x_2 = 1.6$. Using data at these x_i, construct interpolating polynomials of degrees at most one and at most two and use them to approximate $f(1.4)$. Find the absolute errors.

   (a) $f(x) = \sin \pi x$ [S]

   (b) $f(x) = \sqrt[3]{x-1}$

   (c) $f(x) = e^{2x-4}$

   (d) $f(x) = \ln(10x)$

10. Use formula 3.2.3 to find theoretical error bounds for the approximations in question 9. Compare the bound to the actual error. [S]
11. A Lagrange interpolating polynomial is constructed for the function $f(x) = (\sqrt{2})^x$ using $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, $x_3 = 3$. It is used to approximate $f(1.5)$. Find a bound on the error in this approximation.
12. Find the polynomial referred to in question 11. Then

    (a) use the polynomial to approximate $f(1.5)$; and

    (b) calculate the actual error of this approximation, and compare it to the bound you calculated in question 11.

13. Use Neville's method to find the approximation in question 11.
14. The height of a model rocket is given at several times in the following table. Approximate the height of the rocket at time $t = 0.6$ sec using at least two different sets of points. Comment on which approximation is likely most accurate.

Time (sec)	Height (ft)
0.53238	30.0534
0.56040	32.7929
0.58842	35.4956
0.61644	38.1575

15. The following table results from using Neville's method to approximate $f(0.4)$.

0	1	2.6	$P_{0,2}$	3.016
0.25	2	$P_{1,1}$	2.96	
0.5	$P_{2,0}$	2.4		
0.75	8			

    Determine $f(0.5)$. [A]

16. $L_3(x) = -7x^3 + 57x^2 - 134x + 78$ is the degree (at most) 3 interpolating polynomial for the data in the table. Find ω. [A]

x	0.5	0.8	ω	1.4
y	24.375	3.696	0	-17.088

17. Let $P_3(x)$ be the interpolating polynomial for the data $(0, 0)$, $(0.5, y)$, $(1, 3)$, $(2, 2)$. Find y if the coefficient of x^3 in $P_3(x)$ is 6.
18. Let $f(x) = \sqrt{x - x^2}$ and $P_2(x)$ be the interpolating polynomial on $x_0 = 0$, x_1, and $x_2 = 1$. Find the largest value of x_1 in $(0, 1)$ for which $f(0.5) - P_2(0.5) = -0.25$.

19. The interpolating polynomial on $n+1$ points does not always have degree n. It has degree at most n. Plot the data $(1,1)$, $(2,3)$, $(3,5)$, and $(4,7)$, and make a conjecture as to the degree of the polynomial interpolating these four points. What led you to your conjecture?

20. Use Neville's method to find the polynomial described in question 19. Does it have the degree you expected?

21. Let

$$\begin{aligned} x_j &= 1 - \frac{1}{j+1} \quad \text{for } j = 0,1,2,\ldots \\ f(x) &= 5 + 3x^{2018} \\ P_n(x) &= \text{the interpolating polynomial} \\ &\quad \text{passing through} \\ &\quad (x_0, f(x_0)), \ldots, (x_n, f(x_n)). \end{aligned}$$

Find

$$\lim_{n\to\infty} P_n(1).$$

[A]

22. Let $f(x) = e^{-x}$. Two different numbers are chosen at random from the interval $[0,1]$, say x_0 and x_1. Then the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ are used to get a linear Lagrange interpolation approximation to f over the interval $[0,1]$. Find a bound (good for the entire interval and every pair of points x_0 and x_1) for the error in using this approximation.

23. Supply the inductive proof that $P_{0,n}$ is the polynomial of degree at most n interpolating the data $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$. See notes on page 95.

3.3 Newton Polynomials

In this section, we are interested in an efficient automated process for calculating interpolating polynomials. The Lagrange form of an interpolating polynomial is best suited for pencil and paper calculations, not computer automation. Neville's method is well suited for computing the value of an interpolating polynomial at a particular point, not calculation of the polynomial itself. True, Neville's method *can* be used to calculate the interpolating polynomials themselves, but it lends itself to this task no better than the Lagrange form. Presently, we will discover how the same recursive formula used in Neville's method is used to derive a very efficient, computer-friendly method for calculating interpolating polynomials themselves. The result of the computation is a set of coefficients for the Newton form of a polynomial.

Suppose we have already computed the polynomial $N_n(x)$ interpolating the data

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n)).$$

We now wish to compute the polynomial $N_{n+1}(x)$ interpolating the data

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_{n+1}, f(x_{n+1})),$$

and we would like to recycle the work we have already done (much the same way we could add a point of interpolation in Neville's method and reuse all previous work)! One way to attack the problem is to find a polynomial $q(x)$ such that

$$N_{n+1}(x) = N_n(x) + q(x).$$

If the attack is to be successful, we must have $q(x) = N_{n+1}(x) - N_n(x)$ for all x, and, in particular, $q(x_j) = N_{n+1}(x_j) - N_n(x_j)$ for $j = 0, 1, \ldots, n+1$. But $N_{n+1}(x_j) - N_n(x_j) = f(x_j) - f(x_j) = 0$ for $j = 0, 1, \ldots, n$, and $N_{n+1}(x_{n+1}) - N_n(x_{n+1}) = f(x_{n+1}) - N_n(x_{n+1})$. In other words, we seek the polynomial q interpolating the points

$$(x_0, 0), (x_1, 0), \ldots, (x_n, 0), (x_{n+1}, (f - N_n)(x_{n+1})).$$

Ironically, this is a job for the Lagrange form:

$$\begin{aligned} q(x) &= \frac{(x - x_0)\cdots(x - x_n)}{(x_{n+1} - x_0)\cdots(x_{n+1} - x_n)}(f - N_n)(x_{n+1}) \\ &= \frac{(f - N_n)(x_{n+1})}{(x_{n+1} - x_0)\cdots(x_{n+1} - x_n)}(x - x_0)\cdots(x - x_n). \qquad (3.3.1) \end{aligned}$$

But $\frac{(f-N_n)(x_{n+1})}{(x_{n+1}-x_0)\cdots(x_{n+1}-x_n)}$ is just a constant, so we replace it by a_{n+1} so that we have $q(x) = a_{n+1}(x - x_0)\cdots(x - x_n)$. Of course we can calculate a_{n+1} using the formula $\frac{(f-N_n)(x_{n+1})}{(x_{n+1}-x_0)\cdots(x_{n+1}-x_n)}$, but there is a better way, which we will see shortly. We can also learn from the upcoming computation the most convenient form for N_n.

When $n = 0$, q has the form $a_1(x - x_0)$; when $n = 1$, q has the form $a_2(x - x_0)(x - x_1)$; when $n = 2$, q has the form $a_3(x - x_0)(x - x_1)(x - x_2)$; and so on. Of course $N_0(x) = a_0$ is constant since it is the interpolating polynomial of least degree passing through a single point. So $N_1(x) = N_0(x) + a_1(x - x_0)$ immediately takes the form $a_0 + a_1(x - x_0)$; $N_2(x)$ immediately takes the form $a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$; $N_3(x)$ immediately takes the form $a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$; and so on. This would suggest that the most convenient form for N_{n+1}, the one that requires no simplification, is

$$N_{n+1}(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_{n+1}(x - x_0)\cdots(x - x_n). \qquad (3.3.2)$$

Given in this form, the unknown quantity, a_{n+1}, appears as the coefficient of the x^{n+1} term. Consequently, a_{n+1} is *potentially* the leading coefficient of N_{n+1}. If a_{n+1} were zero, then we would not call it the leading coefficient. We will facilitate the rest of this discussion by introducing the following term. For an interpolating polynomial on $k+1$ points, the coefficient of its x^k term is called its **potential leading coefficient** (even if it happens to be zero). Since this potential leading coefficient is the crux of our problem, we focus attention on determining the potential leading coefficient of any interpolating polynomial.

Here is where the recursive formula

$$\begin{aligned} P_{i,m+1}(x) &= \frac{(x - x_{i+m+1})P_{i,m}(x) - (x - x_i)P_{i+1,m}(x)}{x_i - x_{i+m+1}} \\ P_{i,0}(x) &= f(x_i) \end{aligned}$$

used in devising Neville's method comes in handy. In as much as $P_{i,m}$ and $P_{i+1,m}$ both have degree at most m, their potential leading coefficients are the coefficients of their x^m terms. It follows that the coefficient of the x^{m+1} term of $(x - x_{i+m+1})P_{i,m}(x)$ equals the potential leading coefficient of $P_{i,m}(x)$, and, similarly, the coefficient of the x^{m+1} term of $(x - x_i)P_{i+1,m}$ equals the potential leading coefficient of $P_{i+1,m}$. Therefore, the coefficient of the x^{m+1} term of $(x - x_{i+m+1})P_{i,m}(x) - (x - x_i)P_{i+1,m}(x)$ is the difference of the potential leading coefficients of $P_{i,m}$ and $P_{i+1,m}$. To simplify the discussion, we use the notation $f_{i,j}$ for the potential leading coefficient of $P_{i,j}$. Now the coefficient of the x^{m+1} term of $(x - x_{i+m+1})P_{i,m}(x) - (x - x_i)P_{i+1,m}(x)$ is just $f_{i,m} - f_{i+1,m}$. Hence, the potential leading coefficient $f_{i,m+1}$ of $P_{i,m+1}$ (the coefficient of the x^{m+1} term of $P_{i,m+1}$) is given by

$$\begin{aligned} f_{i,m+1} &= \frac{f_{i,m} - f_{i+1,m}}{x_i - x_{i+m+1}} \\ f_{i,0} &= f(x_i). \end{aligned} \tag{3.3.3}$$

Crumpet 21: DividedDifferences

While we choose to use the notation $f_{i,j}$ for the potential leading coefficient of $P_{i,j}$, it is much more customary to use the expanded notation $f[x_i, x_{i+1}, \ldots, x_{i+j}]$ for this quantity, and to call it a j^{th} divided difference.

Finally, we have a formula for the potential leading coefficient that recycles previous calculations. Since N_{n+1} and $P_{0,n+1}$ interpolate the same set of points and both have degree at most $n+1$, they are equal by theorem 7. Therefore, their potential leading coefficients, a_{n+1} and $f_{0,n+1}$ are equal. By recursion 3.3.3, we then have $a_{n+1} = f_{0,n+1} = \frac{f_{0,n} - f_{1,n}}{x_0 - x_{n+1}}$.

It can not be stressed enough that we have not discovered a new polynomial. We have only discovered a new way to calculate the same old interpolating polynomials. N_n, L_n, and $P_{0,n}$ all interpolate the same data and all have degree at most n. They are, therefore, equal by theorem 7. Just the forms in which they are written possibly differ. The polynomial form in equation 3.3.2 is called the Newton form.

Crumpet 22: Newton Polynomials

Typically, the Newton form and divided differences are presented completely independent of Neville's recursive formula, an approach that takes considerably more work to develop. There are reasons to do so, however. Refraining from the use of Neville's formula follows more closely the historical development of the subject since Newton (1643–1727) preceded Neville (1889-1961) by over 200 years! Moreover, following the historical development more naturally leads to further study of divided differences.

As an example, take the polynomial interpolating $f(x) = e^x$ at $x = 0, 1, 2$, as we did in the discussion of Neville's method on page 94. $f_{0,0} = f(0) = 1$, $f_{1,0} = f(1) \approx 2.718281828459045$, and $f_{2,0} = f(2) \approx 7.38905609893065$. So

$$\begin{aligned} f_{0,1} &= \frac{f_{0,0} - f_{1,0}}{x_0 - x_1} = \frac{1 - 2.718281828459045}{0 - 1} \\ &\approx 1.718281828459045 \\ f_{1,1} &= \frac{f_{1,0} - f_{2,0}}{x_1 - x_2} = \frac{2.718281828459045 - 7.38905609893065}{1 - 2} \\ &\approx 4.670774270471606 \\ f_{0,2} &= \frac{f_{0,1} - f_{1,1}}{x_0 - x_2} = \frac{1.718281828459045 - 4.670774270471606}{0 - 2} \\ &\approx 1.47624622100628. \end{aligned}$$

Table 3.3: Newton form example, calculating $N_2(x)$.

x_i	$f_{i,0} = f(x_i)$	$f_{i,1}$	$f_{i,2}$
0	1	1.718281828459045	1.47624622100628
1	2.718281828459045	4.670774270471606	
2	7.38905609893065		

Therefore, $N_2(x) = 1 + 1.718281828459045(x) + 1.47624622100628(x)(x-1)$. $f_{0,i}$ are the coefficients of N_n. Though this computation is manageable without a table, it is most convenient to tabulate the values of $f_{i,j}$ as they are computed (just as is the case for Neville's method). This is true for both humans and computers! A tabulation of the computation makes it easier to internalize the recursion and imagine how this process might be automated. Table 3.3, which is called a table of divided differences (or divided difference table), shows such a tabulation. Adding a data point to the interpolation is as easy as computing another diagonal of coefficients (just like Neville's method).

Sidi's Method

We now return attention to Sidi's k^{th} degree root-finding method,

$$x_{n+1} = x_n - \frac{g(x_n)}{p'_{n,k}(x_n)},$$

where $p_{n,k}$ is the interpolating polynomial passing through the points

$$(x_n, g(x_n)), (x_{n-1}, g(x_{n-1})), \ldots, (x_{n-k}, g(x_{n-k})).$$

In its Newton form,

$$p_{n,k}(x) = g_{n,0} + g_{n-1,1}(x - x_n) + g_{n-2,2}(x - x_n)(x - x_{n-1}) + \cdots + g_{n-k,k}(x - x_n) \cdots (x - x_{n-k}),$$

so

$$p'_{n,k}(x_n) = g_{n-1,1} + g_{n-2,2}(x_n - x_{n-1}) + \cdots + g_{n-k,k}(x_n - x_{n-1}) \cdots (x_n - x_{n-k}). \tag{3.3.4}$$

In particular,

$$p'_{n,2}(x_n) = g_{n-1,1} + (x_n - x_{n-1})g_{n-2,2}$$

and

$$p'_{n,3}(x_n) = g_{n-1,1} + (x_n - x_{n-1})g_{n-2,2} + (x_n - x_{n-1})(x_n - x_{n-2})g_{n-3,3}$$

and so on. As a nested product,

$$p'_{n,k}(x_n) = g_{n-1,1} + (x_n - x_{n-1})\left[g_{n-2,2} + (x_n - x_{n-2})\left[\cdots + (x_n - x_{n-k})\left[g_{n-k,k}\right]\cdots\right]\right].$$

The nested form is particularly efficient for implementation.

```
Assumptions: $g$ is $k$ times differentiable.
Input: Initial values $x_0, x_1, \ldots, x_k$; diagonal entries $g_{k,0}, g_{k-1,1}, \ldots, g_{0,k}$ of the divided difference table for
    $g$.
Step 1: Set $s = g_{0,k}$;
Step 2: For $i = 1, 2, \ldots, k-1$ do Step 3:
    Step 3: Set $s = (x_k - x_i)s + g_{i,k-i}$;
Step 4: Set $x_{k+1} = x_k - \frac{g_{k,0}}{s}$;
Output: Approximation $x_{k+1}$.
```

While this pseudo-code is good as far as it goes, it is far from complete. The most obvious deficiency is that it only executes one step of Sidi's method. A less obvious deficiency is that its input and output do not match in type or quantity, so at the end of the routine, the computer is still not ready to compute another iteration. What we get from this routine is x_{k+1}. What we need to run it again are the two arrays $x_0, x_1, \ldots, x_k$ and $g_{k,0}, g_{k-1,1}, \ldots, g_{0,k}$. In order to prepare these arrays for the next iteration, we must re-index the values of x_i and then compute new values for the $g_{i,k-i}$.

```
Assumptions: $g$ is $k$ times differentiable.
Input: Initial values $x_0, x_1, \dots, x_k$; diagonal entries $g_{k,0}, g_{k-1,1}, \dots, g_{0,k}$ of the divided difference table for $g$.
Step 1: Set $x_{k+1}$ according to Sidi's method applied to $x_0, x_1, \dots, x_k$ and $g_{k,0}, g_{k-1,1}, \dots, g_{0,k}$;
Step 2: Set $g_{k+1,0} = g(x_{k+1})$;
Step 3: For $i = k, k-1, \dots, 1$ do Step 4:
    Step 4: Set $g_{i,k+1-i} = \frac{g_{i+1,k-i} - g_{i,k-i}}{x_{k+1} - x_i}$;
Output: Approximations $x_1, \dots, x_{k+1}$ and corresponding diagonal entries $g_{k+1,0}, g_{k,1}, \dots, g_{1,k}$ of the divided difference table for $g$.
```

This new pseudo-code, which utilizes the previous pseudo-code in its first step is an improvement. Now the input and output match in type and quantity, meaning the output of this routine may be used as input for the next iteration. However, this routine still only calculates one step of Sidi's method. Moreover, we have been ignoring another issue. Each of the routines spelled out in pseudo-code so far assume we have the diagonal entries of the corresponding divided difference table. It is not good practice to make the user of the code worry about this detail. The routine we write should supply these values. After all, the end-user, the person trying to find a root of a function, will only have immediate access to the function and some number of initial values. The routine must supply the rest. Finally, we present pseudo-code in the spirit of other root-finding methods.

```
Assumptions: $g$ has a root at $\hat{x}$; $g$ is $k$ times differentiable; $x_0, x_1, \dots, x_k$ are sufficiently close to $\hat{x}$.
Input: Initial values $x_0, x_1, \dots, x_k$; function $g$; desired accuracy $tol$; maximum number of iterations $N$.
Step 1: For $i = 0, 1, \dots, k$ do Step 2:
    Step 2: Set $g_{i,0} = g(x_i)$;
Step 3: For $j = 1, 2, \dots, k$ do Steps 4-5:
    Step 4: For $i = 0, 1, \dots, k-j$ do Step 5:
        Step 5: Set $g_{i,j} = \frac{g_{i+1,j-1} - g_{i,j-1}}{x_{i+j} - x_i}$
Step 6: For $i = 1 \dots N$ do Steps 7-11:
    Step 7: Compute $x = x_{k+1}$ according to Sidi's method applied to
        $x_0, x_1, \dots, x_k$ and $g_{k,0}, g_{k-1,1}, \dots, g_{0,k}$;
    Step 8: If $|x - x_k| \leq tol$ then return $x$;
    Step 9: Compute $g_{k+1,0}, g_{k,1}, \dots, g_{1,k}$;
    Step 10: Set $x_0 = x_1$; $x_1 = x_2$; $\cdots$ $x_{k-1} = x_k$; $x_k = x$;
    Step 11: Set $g_{k,0} = g_{k+1,0}$; $g_{k-1,1} = g_{k,1}$; $\cdots$ $g_{0,k} = g_{1,k}$;
Step 12: Print "Method failed. Maximum iterations exceeded."
Output: Approximation $x$ near exact fixed point, or message of failure.
```

As complete as this latest pseudo-code is, it leaves one item unaddressed. It requires k initial values to run Sidi's k^{th} degree method. When we encountered the secant method, we noted that needing two initial values as opposed to one was a disadvantage. The disadvantage is only magnified in Sidi's method where $k+1$ initial values are required. However, just as with the secant method, we can automatically generate initial values if needed. If Sidi's method is given one initial value, x_0, and we are trying to find a root of the function g, then we can set $x_1 = x_0 + g(x_0)$ just as we did for the secant method. You may recall, this was not particularly successful, however. The secant method often failed to converge with this selection of initial condition.

Much less is known about Sidi's method and how the selection of intial values affects convergence. It might make an interesting project to analyze good and bad practices for selecting initial values. In any case, if you have initial values $x_0, x_1, \dots, x_j$ with $1 < j < k$, the remaining $k+1-j$ intial values can be found using Sidi's method of degree j (on $x_0, x_1, \dots, x_j$) to get x_{j+1} followed by using Sidi's method of degree $j+1$ (on $x_0, x_1, \dots, x_{j+1}$) to get x_{j+2} followed by using Sidi's method of degree $j+2$ (on $x_0, x_1, \dots, x_{j+2}$) to get x_{j+3}, and so on until x_k is computed.

More divided differences

Divided difference tables are generally computed for the sake of finding coefficients for one interpolating polynomial, and one interpolating polynomial only. However, each table of divided differences is rife with representations of interpolating polynomials. One of the strengths of a divided difference table is that its entries may be reused should more data be added. This same property can be thought of in reverse. Suppose you have a divided difference table computed over 4 data values but you are only interested in an at-most-degree-2 interpolating polynomial. The divided difference table

$$\begin{array}{c|cccc}
x_0 & f_{0,0} & f_{0,1} & f_{0,2} & f_{0,3} \\
x_1 & f_{1,0} & f_{1,1} & f_{1,2} & \\
x_2 & f_{2,0} & f_{2,1} & & \\
x_3 & f_{3,0} & & &
\end{array}$$

actually gives us two different at-most-quadratic interpolating polynomials with four representations for each! First, the table was devised to compute the interpolating polynomial

$$P_3(x) = f_{0,0} + f_{0,1}(x - x_0) + f_{0,2}(x - x_0)(x - x_1) + f_{0,3}(x - x_0)(x - x_1)(x - x_2).$$

Notice that if we simply truncate the $f_{0,3}(x - x_0)(x - x_1)(x - x_2)$ term, we still have an interpolating polynomial with nodes x_0, x_1, x_2. We can support this claim in at least two ways. First, the term $f_{0,3}(x - x_0)(x - x_1)(x - x_2)$ is 0 at x_0, x_1, x_2 so it does not contribute to the interpolation at the nodes x_0, x_1, x_2. Second, we can "reverse engineer" the table, simply erasing the bottom-most diagonal. The remaining table is still a legitimate divided difference table since none of the remaining entries depends on any of the erased entries:

$$\begin{array}{c|ccc}
x_0 & f_{0,0} & f_{0,1} & f_{0,2} \\
x_1 & f_{1,0} & f_{1,1} & \\
x_2 & f_{2,0} & &
\end{array}$$

So

$$P_2(x) = f_{0,0} + f_{0,1}(x - x_0) + f_{0,2}(x - x_0)(x - x_1)$$

is one of the degree at most 2 interpolating polynomials. Erasing the top row of the table also leaves a legitimate divided difference table:

$$\begin{array}{c|ccc}
x_1 & f_{1,0} & f_{1,1} & f_{1,2} \\
x_2 & f_{2,0} & f_{2,1} & \\
x_3 & f_{3,0} & &
\end{array}$$

so

$$Q_2(x) = f_{1,0} + f_{1,1}(x - x_1) + f_{1,2}(x - x_1)(x - x_2)$$

is another degree at most 2 interpolating polynomial. Notice that P_2 and Q_2 are not just different representations of the same polynomial. They are two different polynomials! P_2 interpolates over the nodes x_0, x_1, x_2 while Q_2 interpolates over the nodes x_1, x_2, x_3.

The bottom diagonals of each truncated table give degree at most 2 interpolating polynomials as well. Remember, $f_{i,j}$ represents the potential leading coefficient of the interpolating polynomial over the nodes $x_i, x_{i+1}, \ldots, x_{i+j}$. Hence,

$$\tilde{Q}_2(x) = f_{3,0} + f_{2,1}(x - x_3) + f_{1,2}(x - x_3)(x - x_2)$$

interpolates over the nodes x_3, x_2, x_1 and

$$\tilde{P}_2(x) = f_{2,0} + f_{1,1}(x - x_2) + f_{0,2}(x - x_2)(x - x_1)$$

interpolates over the nodes x_2, x_1, x_0. These are not new polynomials. These are new representations for P_2 and Q_2. Actually, $\tilde{P}_2 = P_2$ and $\tilde{Q}_2 = Q_2$.

The critical feature of each of these interpolating polynomial representations is that each successive coefficient depends on all the same nodes as its predecessor, plus one new one. For example, $f_{2,0}$ depends on x_2, $f_{1,1}$ depends on x_2 and x_1, and $f_{0,2}$ depends on x_2, x_1, and x_0. Hence, these three coefficients can be used to produce the interpolating polynomial over the nodes x_0, x_1, x_2 in the form of polynomial $\tilde{P}_2$ (which, as we have already noted, equals P_2). Another representation for the same polynomial can be written by utilizing $f_{1,0}$ (which depends on x_1), $f_{0,1}$ (which depends on x_1 and x_0), and $f_{0,2}$ (which depends on x_1, x_0, x_2):

$$\hat{P}_2(x) = f_{1,0} + f_{0,1}(x - x_1) + f_{0,2}(x - x_1)(x - x_0)$$

to give a representation of the polynomial interpolating over x_0, x_1, x_2 (which, therefore, must equal P_2). There is one more representation of P_2 that can be extracted from the original divided difference table. It comes from the coefficients $f_{1,0}, f_{1,1}, f_{0,2}$. Can you write it down? Answer on page 107. There are two more representations of Q_2 that can be extracted from the original divided difference table. Can you write them down? Answers on page 108.

Key Concepts

Newton form of an interpolating polynomial: The Newton form, N_n, of the polynomial of degree at most n interpolating the points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ is

$$N_n(x) = a_0 + a_1(x - x_{i_0}) + a_2(x - x_{i_0})(x - x_{i_1}) + \cdots + a_n(x - x_{i_0}) \cdots (x - x_{i_{n-1}})$$

for n distinct indices $i_0, i_1, \ldots, i_{n-1}$ from the set $\{0, 1, 2, \ldots, n\}$. The Newton form for a particular set of data is not unique.

Potential leading coefficient: For an interpolating polynomial on $k + 1$ points, the coefficient of its x^k term is called its potential leading coefficient.

Divided differences: The coefficients of the Newton form of an interpolating polynomial are called divided differences.

Exercises

1. Modify the Neville's method pseudo-code on page 96 to produce pseudo-code for computing the coefficients of N_n.

2. C Modify the Neville's method code on page ?? to produce code for computing the coefficients of N_n. Test it by computing N_2 interpolating $f(x) = e^x$ at $x = 0, 1, 2$ and comparing your result to that on page 101.

3. Let $f(0.1) = 0.12$, $f(0.2) = 0.14$, $f(0.3) = 0.13$, and $f(0.4) = 0.15$.

   (a) Find the leading coefficient of the polynomial of least degree interpolating these data.

   (b) Suppose, additionally, that $f(0.5) = 0.11$. Use your previous work to find the leading coefficient of the polynomial of least degree interpolating all of the data.

4. Find a Newton form of the polynomial of degree at most 3 interpolating the points $(1, 2)$, $(2, 2)$, $(3, 0)$ and $(4, 0)$. [S]

5. Use the method of divided differences to find the at-most-second-degree polynomial interpolating the points $(0, 10)$, $(30, 58)$, $(1029, -32)$. [A]

6. Use divided differences to find an interpolating polynomial for the data $f(1) = 0.987$, $f(2.2) = -0.123$, and $f(3) = 0.432$. [S]

7. Create a divided differences table for the following data using only pencil and paper.

$$f(1.2) = 2.2 \quad f(1.4) = 2.1 \quad f(1.6) = 2.3$$

   (a) What is the interpolating polynomial of degree at most 2? Does it actually have degree 2?

   (b) Write down two distinct linear interpolating polynomials for this data based on your table.

8. Use divided differences to find the at-most-cubic polynomial of exercise 19 of section 3.2. Does it have the expected degree? [A]

9. Find the degree at most two interpolating polynomial of the form

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

   for the data in the table.

x	2	3	4
$f(x)$	3	5	4

10. C Use the the computer code from question 2 to compute the interpolating polynomial of at most degree four for the data:

x	$f(x)$
0.0	−6.00000
0.1	−5.89483
0.3	−5.65014
0.6	−5.17788
1.0	−4.28172

Then add $f(1.1) = -3.9958$ to the table, and compute the interpolating polynomial of degree at most 5 using a calculator. You may use the computer code to check your work. [S]

11. Use the computer code from question 2 to find interpolating polynomials of degrees (at most) one, two, and three for the following data. Approximate $f(8.4)$ using each polynomial.

$$f(8.1) = 16.94410,\ f(8.3) = 17.56492,$$
$$f(8.6) = 18.50515,\ f(8.7) = 18.82091$$

12. Find a bound on the error in using the interpolating polynomial of question 6 to approximate $f(2)$ assuming that all derivatives of f are bounded between -2 and 1 over the interval $[1, 3]$. [S]

13. Regarding the polynomial of question 9,

    (a) use the polynomial to approximate $f(2.5)$; and

    (b) assuming $f \in C^3$, find a theoretical bound on the error of approximating $f(x)$ on the interval $[2, 4]$.

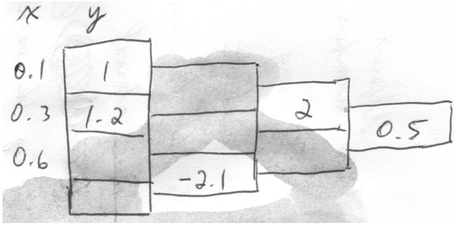
14. [A]

    (a) Find an error bound, in terms of $f^{(4)}(\xi_{8.4})$, for the approximation $P_3(8.4)$ in question 11.

    (b) Find an error bound, in terms of $f^{(4)}(x)$, for the approximation $P_3(x)$ in question 11 good for any $x \in [8.1, 8.7]$.

    (c) Suppose $f^{(4)}(x) = x \cos x - e^x$ for the function $f(x)$ of question 11. Use this information to find an error bound for the approximation $P_3(x)$ good for any $x \in [8.1, 8.7]$.

15. Buck spilled coffee on his divided differences table, obscuring several numbers. Nevertheless, there is enough legible information to find the at-most-degree-3 polynomial interpolating the data. Find it. [A]



x	y			
0.1	1			
			2	
0.3	1.2			0.5
0.6				
		-2.1		

16. Show that the polynomial interpolating the following data has degree 3.

x	-2	-1	0	1	2	3
$f(x)$	1	4	11	16	13	-4

17. For a function f, Newton's divided difference formula gives the interpolating polynomial

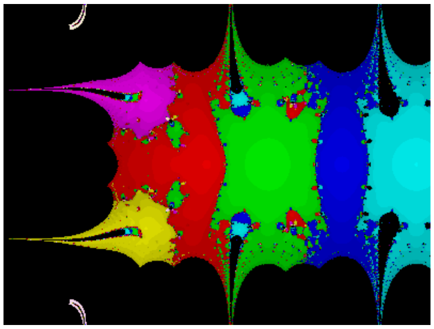
$$N_3(x) = 1 + 4x + 4x(x - 0.25) + \frac{16}{3}x(x - 0.25)(x - 0.5)$$

on the nodes $x_0 = 0$, $x_1 = 0.25$, $x_2 = 0.5$, $x_3 = 0.75$. Find $f(0.75)$. [S]

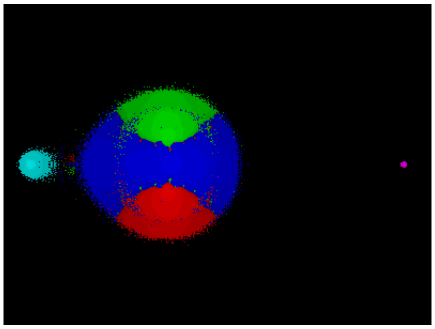
18. Match the function with its Seeded Sidi method convergence diagram. In each case, Sidi's 6^{th} degree method was used. The real axis passes through the center of each diagram, and the imaginary axis is represented, but is not necessarily centered. [S]

$$\begin{aligned} f(x) &= \sin x \\ g(x) &= \sin x - e^{-x} \\ h(x) &= e^x + 2^{-x} + 2\cos x - 6 \\ l(x) &= 56 - 152x + 140x^2 - 17x^3 - 48x^4 + 9x^5 \end{aligned}$$

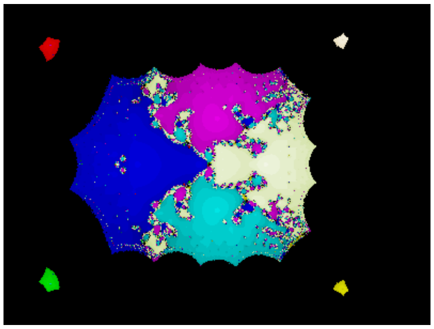
(a)



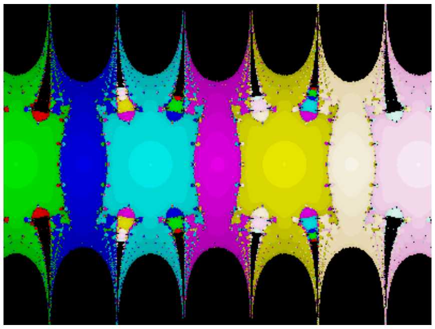
(b)



(c)



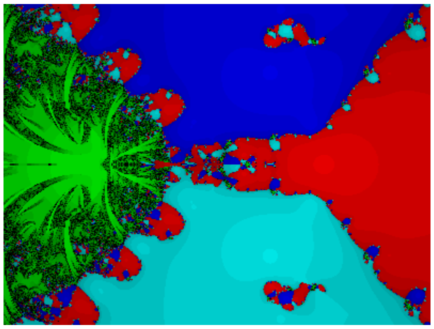
(d)



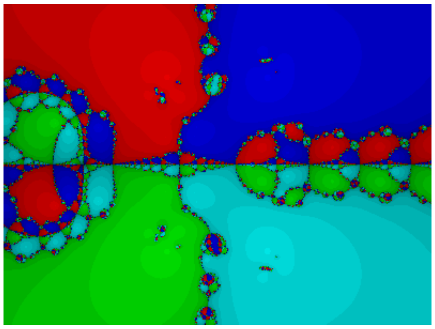
19. Match the function with its Seeded Sidi method convergence diagram. The real axis passes through the center of each diagram, and the imaginary axis is represented, but is not necessarily centered. [A]

$$\begin{aligned} f(x) &= x^4 + 2x^2 + 4 \\ g(x) &= (x^2)(\ln x) + (x-3)e^x \\ h(x) &= 1 + 2x + 3x^2 + 4x^3 + 5x^4 + 6x^5 \\ l(x) &= (\ln x)(x^3 + 1) \end{aligned}$$

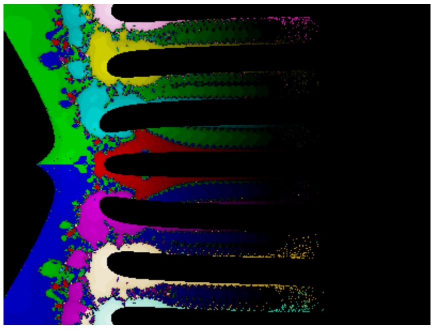
(a)



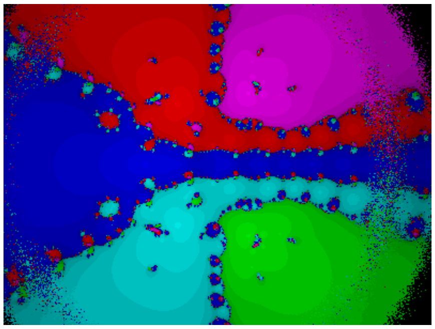
(b)



(c)



(d)



Answers

P_2 from $f_{1,0}, f_{1,1}, f_{0,2}$: $\overline{P}_2(x) = f_{1,0} + f_{1,1}(x - x_1) + f_{0,2}(x - x_1)(x - x_2)$

Q_2 **two new ways:** $\hat{Q}_2(x) = f_{2,0} + f_{1,1}(x - x_2) + f_{1,2}(x - x_2)(x - x_1)$ and $\overline{Q}_2(x) = f_{2,0} + f_{2,1}(x - x_2) + f_{1,2}(x - x_2)(x - x_3)$

Chapter 4

Numerical Calculus

4.1 Rudiments of Numerical Calculus

The basic idea

$g(x) = x - \frac{2\pi}{3}\sin(x)$ has a root between 0 and π. You are trying various methods and become interested in how the choice of initial value affects the results. Using Newton's method, you do some research into how the choice of x_0 affects x_2. You run some tests and come up with the following data.

x_0	x_2
93/70	2.084603181618954
95/70	2.055494116570853
97/70	2.030278824314539
99/70	2.009751835391139
101/70	1.993574976724822
103/70	1.981091507449763
105/70	1.971614474758557

Using fixed point iteration on $f(x) = \frac{2\pi}{3}\sin(x)$, you decide to examine how the choice of x_0 affects x_{10}, not x_2 since fixed point iteration generally converges slowly. You run some tests on this method and come up with the following data.

x_0	x_{10}
1/7	1.949880891899200
2/7	1.951091775564697
3/7	1.923339403354019
4/7	1.941460911122824
5/7	1.960870620285721
6/7	1.965674866641883
1	1.961228252911260

In the Newton's method experiment, x_2 is a function of x_0, and in the fixed point iteration experiment, x_{10} is a function of x_0. So you start to think of them completely independently from the original root-finding question. As they sit in their tabular form, they are just two functions for which you know a handful of values and not much more. What do these functions look like? Do we have enough information to perhaps find their derivatives, and, hence, local extrema? Can we find their antiderivatives? This is the stuff of numerical calculus. We can certainly approximate these things.

In chapter 3 we learned how to approximate functions by interpolation, so we know we can use the tabular data to approximate the functions themselves. But what about their derivatives and integrals? Well, polynomials are easy to differentiate and integrate. Perhaps we can use the derivatives and integrals of interpolating polynomials to approximate the derivatives and integrals of $x_2(x_0)$ and $x_{10}(x_0)$. Indeed we can!

In order to avoid the confusion of using x_0 for multiple purposes, we will rename our functions $\nu(x)$ for $x_2(x_0)$ and $\varphi(x)$ for $x_{10}(x_0)$. Hence, we have $\nu(93/70) = 2.0846\ldots$, $\nu(95/70) = 2.0554\ldots$, and so on. Similarly, we have now $\varphi(1/7) = 1.9498\ldots$, $\varphi(2/7) = 1.9510\ldots$, and so on. We will also take up the practice of calling the x-coordinates of the prescribed interpolation points nodes. Hence, the nodes we have for ν are 93/70, 95/70, and so on. The nodes we have for φ are 1/7, 2/7, and so on.

Crumpet 23: ν and φ

ν is the (lower case) thirteenth letter of the Greek alphabet and is pronounced `noo`. φ is the (lower case) twenty-first letter of the Greek alphabet and is pronounced `fee`. The letter `fee` is also written ϕ, but in mathematics it is much more common to see the variant φ, perhaps to avoid confusion between `fee` and the empty set, $\emptyset$. The capital versions of ν and φ are N and Φ, respectively.

We begin by considering interpolating polynomials on three nodes. For ν, we use the nodes 93/70, 99/70, and 1.5, and get

$$P_{2,\nu}(x) = .07673215587088045x^2 - .07445530457646088x + 1.95895140161684.$$

For φ, we use the nodes 1/7, 4/7, and 1, and get

$$P_{2,\varphi}(x) = 2.498590686342254x^2 - 7.726543017101505x + 7.939599956140455.$$

We have added a second subscript to P_2 in order to distinguish the interpolating polynomial for ν from that for φ. Now we can approximate derivatives and integrals for both ν and φ using $P_{2,\nu}$ and $P_{2,\varphi}$, respectively:

$$\begin{aligned}
\nu'(x) &\approx P'_{2,\nu}(x) = 4.997181372684508x - 7.726543017101505\\
\varphi'(x) &\approx P'_{2,\varphi}(x) = .1534643117417609x - .07445530457646088\\
\int \nu\,dx &\approx \int P_{2,\nu}dx = .8328635621140847x^3 - 3.863271508550753x^2 + 7.939599956140455x + C\\
\int \varphi\,dx &\approx \int P_{2,\varphi}dx = .02557738529029348x^3 - .03722765228823044x^2 + 1.95895140161684x + D.
\end{aligned}$$

So, for example,

$$\begin{aligned}
\nu'(1.4) &\approx P'_{2,\nu}(1.4)\\
&= 4.997181372684508(1.4) - 7.726543017101505\\
&= -.7304890953431942\\
\varphi'(0.5) &\approx P'_{2,\varphi}(0.5)\\
&= .1534643117417609(0.5) - .07445530457646088\\
&= .002276851294419568
\end{aligned}$$

and

$$\begin{aligned}
\int_{1.4}^{1.5} \nu(x)dx &\approx \int_{1.4}^{1.5} P_{2,\nu}(x)dx\\
&= \big[.8328635621140847x^3 - 3.863271508550753x^2 + 7.939599956140455x\big]_{1.4}^{1.5}\\
&= .1991481658283149\\
\int_0^1 \varphi(x)dx &\approx \int_0^1 P_{2,\varphi}(x)dx\\
&= \big[.02557738529029348x^3 - .03722765228823044x^2 + 1.95895140161684x\big]_0^1\\
&= 1.947301134618903.
\end{aligned}$$

That's it! This exercise encapsulates the entire strategy. Given some values of an otherwise unknown function, we will approximate the unknown function with a polynomial. We will then approximate derivatives and integrals of

Table 4.1: Estimating the derivatives and integrals of ν and φ.

quantity	using P_2	using P_6
$\nu'(1.4)$	$-.7304890953431942$	$-.7178145479410887$
$\varphi'(0.5)$	$.002276851294419568$	$.1447147284558277$
$\int_{1.4}^{1.5} \nu(x)dx$	$.1991481658283149$	$.1991932206801721$
$\int_0^1 \varphi(x)dx$	1.947301134618903	1.925578216262883

the unknown function by differentiating and integrating the polynomial. There is very little more to be said about the idea. There is, however, a lot more to be said about automation, accuracy, and efficiency, the focus of the rest of the chapter. But before we tackle those issues, we will have another look and ν and φ.

Using all the nodes of ν, and the help of a computer algebra system, we compute the sixth degree interpolating polynomial

$$\begin{aligned} P_{6,\nu}(x) &= -1342.393417879939x^6 + 11632.43754466623x^5 - 41996.4789301455x^4 \\ &\quad +80851.91317212582x^3 - 87536.60487741232x^2 + 50528.3026241064x \\ &\quad -12144.27629915625. \end{aligned}$$

Using all the nodes of φ (and a computer algebra system) we compute the sixth degree interpolating polynomial

$$\begin{aligned} P_{6,\varphi}(x) &= -25.41848741926543x^6 + 97.00017832506126x^5 - 147.1805326076494x^4 \\ &\quad +111.7996194440324x^3 - 43.71110414341027x^2 + 8.049781257197147x \\ &\quad +1.421773396945804. \end{aligned}$$

Again we have added a second subscript in order to distinguish the interpolating polynomial for ν from that for φ. Now we can get second estimates for $\nu'(1.4)$, $\varphi'(0.6)$, $\int_{1.4}^{1.5} \nu\, dx$, and $\int_0^1 \varphi\, dx$:

$$\begin{aligned} \nu'(1.4) &\approx P'_{6,\nu}(1.4) \approx -.7178145479410887 \\ \varphi'(0.5) &\approx P'_{6,\varphi}(0.5) \approx .1729311759579151 \\ \int_{1.4}^{1.5} \nu(x)dx &\approx \int_{1.4}^{1.5} P_{6,\nu}(x)dx \approx .1991932206801721 \\ \int_0^1 \varphi(x)dx &\approx \int_0^1 P_{6,\varphi}(x)dx \approx 1.925578216262883. \end{aligned}$$

Table 4.1 summarizes the eight estimates we have made so far. The first four digits of the estimates of $\int_{1.4}^{1.5} \nu(x)dx$ agree, and the first two of $\int_0^1 \varphi(x)dx$ agree. So there is some agreement for the estimates of the integrals. The estimates for the derivatives don't agree quite as well, however. The estimates for $\nu'(1.4)$ only agree in their first significant digit. They both suggest $\nu'(1.4) \approx -.7$. But there is essentially no agreement between the estimates of $\varphi'(0.5)$. One approximation is more than 60 times the other! Based on this simple analysis, we should have a hard time believing either estimate of $\varphi'(0.5)$. And we should only trust the first few digits of the others. We will see later that we can use this type of comparison to have the computer decide whether an approximation is good or not.

Issues

There are three issues with the method of estimating derivatives and integrals just outlined.

1. **Efficiency.** For illustrative purposes and understanding the basic concept of numerical calculus, it is a good idea to calculate some interpolating polynomials as done in the previous subsection. However, it is cumbersome and time-consuming to do so. We will dedicate significant energy into finding shortcuts to this direct method, thus making it more efficient and practical.

2. **Automation.** Numerical methods are meant to be run by a computer, not a human with a calculator. We need to find ways that a computer can handle interpolating polynomials. This issue has intimate ties with efficiency. After all, what will make an algorithm efficient is if it can be executed quickly by a computer!

3. **Accuracy.** So far we have done very little to determine how accurate our approximations are. We need to get a better handle on the error terms in order to understand how to use the method accurately.

Presently, we make strides toward addressing all three of these issues, but we leave the bulk of it for the upcoming sections.

In chapter 3, we labeled the nodes of an interpolating function $x_0, x_1, \dots, x_n$. It will be beneficial to begin calling them $x_0 + \theta_0 h, x_0 + \theta_1 h, \dots, x_0 + \theta_n h$ instead. And for most of our analysis, we will use $x_0 + \theta h$ instead of x for the point at which we desire an estimate. One might call this substitution a change of variables or a recalibration of the x-axis.

To see how this helps with the analysis, consider the degree at most 2 interpolating polynomial of f with nodes

$$x_0 + \theta_0 h, \ x_0 + \theta_1 h, \ \text{and } x_0 + \theta_n h.$$

In the notation of chapter 3, we have

$$P_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2),$$

but with the new notation, we replace x_0 by $x_0 + \theta_0 h$, x_1 by $x_0 + \theta_1 h$, x_2 by $x_0 + \theta_2 h$, and x by $x_0 + \theta h$, giving us

$$\begin{aligned} P_2(x_0 + \theta h) &= \frac{(\theta - \theta_1)(\theta - \theta_2)}{(\theta_0 - \theta_1)(\theta_0 - \theta_2)} f(x_0 + \theta_0 h) \\ &\quad + \frac{(\theta - \theta_0)(\theta - \theta_2)}{(\theta_1 - \theta_0)(\theta_1 - \theta_2)} f(x_0 + \theta_1 h) \\ &\quad + \frac{(\theta - \theta_0)(\theta - \theta_1)}{(\theta_2 - \theta_0)(\theta_2 - \theta_1)} f(x_0 + \theta_2 h). \end{aligned} \tag{4.1.1}$$

For the most part, we have just swapped x for θ and x_i for θ_i. This benign-looking change is actually a huge step forward! This formula makes it apparent that the actual values of the x_i are not important. It is only their location relative to some base point, x_0, measured by some characteristic length, h, that matters. θ and the θ_i are those measures. Essentially this makes x_0 the origin and h the unit of measure on the x-axis. We measure all values by how many lengths of h they are from x_0.

To illustrate the benefit, let us assume that we have three nodes, equally spaced, so the least and greatest nodes are the same distance from the third, middle node. Setting the central node as the base point, x_0, and the characteristic length, h, to the distance from this central node to the others, we can then label them

$$x_0 - h, \ x_0, \ \text{and } x_0 + h.$$

And we have already arrived at the essential point. It doesn't matter if the set of nodes is $\{1, 2, 3\}$ or $\{80, 90, 100\}$ or $\{-4.3, -4.2, -4.1\}$. In each of these sets, we have three nodes, one of which is the midpoint of the other two. Each set of nodes is equal to the set $\{x_0 - h, x_0, x_0 + h\}$ for some values of x_0 and h. Hence, if we can do any analysis with the set $\{x_0 - h, x_0, x_0 + h\}$, then we get information about working with any of the sets of nodes $\{1, 2, 3\}$ or $\{80, 90, 100\}$ or $\{-4.3, -4.2, -4.1\}$ and so on.

Back to the set of nodes $\{x_0 - h, x_0, x_0 + h\}$. For this set of nodes, we have $\theta_0 = -1$, $\theta_1 = 0$, and $\theta_2 = 1$. Substituting into 4.1.1,

$$\begin{aligned} P_2(x_0 + \theta h) &= \frac{(\theta)(\theta - 1)}{(-1)(-2)} f(x_0 - h) + \frac{(\theta + 1)(\theta - 1)}{(1)(-1)} f(x_0) + \frac{(\theta + 1)(\theta)}{(2)(1)} f(x_0 + h) \\ &= \frac{\theta^2 - \theta}{2} f(x_0 - h) + (1 - \theta^2) f(x_0) + \frac{\theta^2 + \theta}{2} f(x_0 + h). \end{aligned}$$

Now this formula can be used to get the interpolating parabola over any set of three equally spaced nodes.

In an attempt to apply this formula to ν, consider the nodes 93/70, 99/70, and 105/70. Since $\frac{99}{70} - \frac{93}{70} = \frac{105}{70} - \frac{99}{70}$, we have a set of nodes of the form $\{x_0 - h, x_0, x_0 + h\}$ with $x_0 = \frac{99}{70}$ and $h = \frac{6}{70} = \frac{3}{35}$. It just so happens that

$1.4 = \frac{99}{70} - \frac{1}{6} \cdot \frac{3}{35}$, so we use $\theta = -\frac{1}{6}$ to calculate $P_{2,\nu}(1.4)$:

$$\begin{aligned}
P_{2,\nu}(1.4) &= P_{2,\nu}\left(x_0 - \frac{1}{6}h\right) \\
&= \frac{\left(-\frac{1}{6}\right)^2 + \frac{1}{6}}{2}\nu\left(\frac{93}{70}\right) + \left(1 - \left(-\frac{1}{6}\right)^2\right)\nu\left(\frac{99}{70}\right) + \frac{\left(-\frac{1}{6}\right)^2 - \frac{1}{6}}{2}\nu\left(\frac{105}{70}\right) \\
&= \frac{7\nu\left(\frac{93}{70}\right) + 70\nu\left(\frac{99}{70}\right) - 5\nu\left(\frac{105}{70}\right)}{72} \\
&= \frac{7(2.084603181618954) + 70(2.009751835391139) - 5(1.971614474758557)}{72} \\
&= 2.019677477429439.
\end{aligned}$$

This seems a pretty good estimate since it is between $\nu(93/70) \approx 2.085$ and $\nu(99/70) \approx 2.009$ but significantly closer to 2.009. After all, 1.4 is between $93/70 \approx 1.328$ and $99/70 \approx 1.414$ but significantly closer to 1.414. Equation 3.2.3 gives us some idea how good we might expect this estimate to be.

But let's back this calculation up just a couple steps. The constants of the $\frac{7\nu\left(\frac{93}{70}\right)+70\nu\left(\frac{99}{70}\right)-5\nu\left(\frac{105}{70}\right)}{72}$ step were determined purely from the values of θ and the θ_i. And the $\frac{93}{70}$, $\frac{99}{70}$, and $\frac{105}{70}$ are just the three nodes, x_0-h, x_0, x_0+h, so what we really have here is a prescription, or formula, for the value $P_2(x_0 - \frac{1}{6}h)$ for *any* degree at most 2 interpolating polynomial over the nodes $x_0 - h$, x_0, and $x_0 + h$:

$$\nu\left(x_0 - \frac{1}{6}h\right) \approx P_{2,\nu}\left(x_0 - \frac{1}{6}h\right) = \frac{7\nu(x_0 - h) + 70\nu(x_0) - 5\nu(x_0 + h)}{72}.$$

And there is nothing special about the particular ν in this formula either. None of the constants $-\frac{1}{6}$, 7, 70, -5, nor 72 is dependent on ν, but rather only dependent on the spacing of the nodes. Therefore, given any function f, we can extract from this calculation the succinct approximation formula

$$f\left(x_0 - \frac{1}{6}h\right) \approx \frac{7f(x_0 - h) + 70f(x_0) - 5f(x_0 + h)}{72}. \tag{4.1.2}$$

This formula illustrates the real purpose in reframing the values of the x_i in terms of x_0, h, and the θ_i. This way, we get formulas applicable to a whole class of nodes, not just one particular set of nodes.

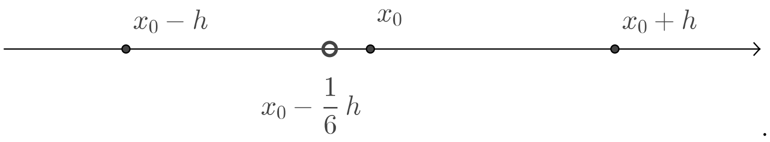
As for φ, the nodes $\frac{1}{7}$, $\frac{4}{7}$, and 1 are equally spaced, so the set $\{\frac{1}{7}, \frac{4}{7}, 1\}$ has the form $\{x_0 - h, x_0, x_0 + h\}$ where $x_0 = \frac{4}{7}$ and $h = \frac{3}{7}$. Not by accident, it happens that $\frac{4}{7} - \frac{1}{6} \cdot \frac{3}{7} = 0.5$, so $\varphi(0.5) = \varphi(x_0 - \frac{1}{6}h)$ where $x_0 = \frac{4}{7}$ and $h = \frac{3}{7}$. Now we can use formula 4.1.2 to approximate $\varphi(0.5)$!

$$\begin{aligned}
\varphi(0.5) &\approx P_{2,\varphi}(0.5) = \frac{7\varphi(x_0 - h) + 70\varphi(x_0) - 5\varphi(x_0 + h)}{72} \\
&= \frac{7(1.9498808918992) + 70(1.941460911122824) - 5(1.96122825291126)}{72} \\
&= 1.94090678829633.
\end{aligned}$$

This time, we have completely circumvented any direct calculation and evaluation of $P_{2,\varphi}$. Formula 4.1.2 allows us to calculate $P_{2,\varphi}(0.5)$ directly from the values of φ at the three nodes. No need to calculate, refer back to, evaluate, or simplify $P_{2,\varphi}$! All of that has been done in deriving the formula. Very quick. Very efficient.

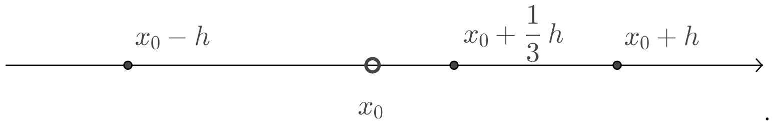
Stencils

A formula such as 4.1.2 is only applicable to a set of nodes and point of evaluation with the same geometry (relative positioning) as those used to derive the formula. Therefore, it will be important to keep track of the geometry used to derive such formulas. To that end, we often refer to a particular set of nodes with its corresponding point of evaluation as a stencil. For example, the nodes $x_0 - h$, x_0, $x_0 + h$ with point of evaluation $x_0 - \frac{1}{6}h$ form a stencil—a relative positioning of points that can be scaled (by changing the value of h) and translated (by changing the value of x_0). On a number line, this particular stencil looks like



x_0 can be located anywhere and h can be any size, even negative. It is this flexibility that makes formulas like 4.1.2 useful.

Now let's suppose we do not have evenly spaced data, but we are interested in a point midway between two others. An appropriate three-point stencil would use the nodes $x_0 - h$, the leftmost node, $x_0 + h$, the rightmost node, $x_0 + \theta_1 h$ for some θ_1 between -1 and 1, the middle node, and point of evaluation x_0, the point midway between the leftmost and rightmost nodes. For $\theta_1 = \frac{1}{3}$, this stencil looks like



.

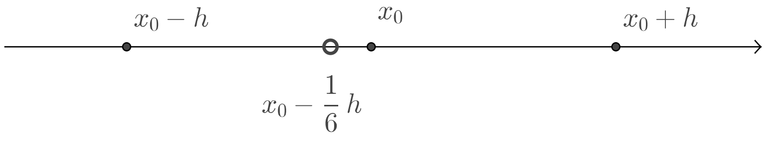
And we can derive a formula for $P_2(x_0)$ based on the values of f at the three nodes. Plugging $\theta = 0$, $\theta_0 = -1$, $\theta_1 = \frac{1}{3}$, and $\theta_2 = 1$ into equation 4.1.1, we get

$$\begin{aligned} P_2(x_0) &= \frac{(-\frac{1}{3})(-1)}{(-\frac{4}{3})(-2)} f(x_0 - h) + \frac{(1)(-1)}{(\frac{4}{3})(-\frac{2}{3})} f(x_0 + \frac{1}{3}h) + \frac{(1)(-\frac{1}{3})}{(2)(\frac{2}{3})} f(x_0 + h) \\ &= \frac{f(x_0 - h) + 9f(x_0 + \frac{1}{3}h) - 2f(x_0 + h)}{8}, \end{aligned}$$

again a succinct formula applicable to any function f. No need to calculate the interpolating polynomial or evaluate it directly for any data that fit this stencil. That part has already been done and simplified.

Derivatives

Derivative formulas can be derived likewise. Once derived for a given stencil, they can be used very easily and efficiently for other data fitting the same stencil. We now find the formula for the first derivative, $P_2'(x_0 - \frac{1}{6}h)$, over the stencil



used earlier. We begin by recognizing that in 4.1.1 x is a function of θ. In particular, $x(\theta) = x_0 + h\theta$, so $\frac{d}{d\theta}x(\theta) = h$. By the chain rule, $\frac{d}{d\theta}P_2(\theta) = \frac{d}{dx}P_2(x) \cdot \frac{d}{d\theta}x(\theta) = h\frac{d}{dx}P_2(x)$. From equation 4.1.1, we then have

$$\begin{aligned} \frac{d}{dx}P_2(x) &= \frac{\frac{d}{d\theta}P_2(\theta)}{h} \\ &= \frac{(\theta - \theta_1) + (\theta - \theta_2)}{h(\theta_0 - \theta_1)(\theta_0 - \theta_2)} f(x_0 + \theta_0 h) \\ &\quad + \frac{(\theta - \theta_0) + (\theta - \theta_2)}{h(\theta_1 - \theta_0)(\theta_1 - \theta_2)} f(x_0 + \theta_1 h) \\ &\quad + \frac{(\theta - \theta_0) + (\theta - \theta_1)}{h(\theta_2 - \theta_0)(\theta_2 - \theta_1)} f(x_0 + \theta_2 h). \end{aligned} \tag{4.1.3}$$

In particular, when $\theta_0 = -1$, $\theta_1 = 0$, $\theta_2 = 1$, and $\theta = -\frac{1}{6}$, we have

$$\begin{aligned} P_2'\left(x_0 - \frac{1}{6}h\right) &= \frac{-\frac{1}{6} - \frac{7}{6}}{h(-1)(-2)} f(x_0 - h) + \frac{\frac{5}{6} - \frac{7}{6}}{h(1)(-1)} f(x_0) + \frac{\frac{5}{6} - \frac{1}{6}}{h(2)(1)} f(x_0 + h) \qquad (4.1.4) \\ &= \frac{-2f(x_0 - h) + f(x_0) + f(x_0 + h)}{3h}. \end{aligned}$$

We now have a formula for $P_2'(x_0 - \frac{1}{6}h) \approx f'(x_0 - \frac{1}{6}h)$ for the stencil with nodes $x_0 - h$, x_0, $x_0 + h$ and $x = x_0 - \frac{1}{6}h$. We can now apply this formula to approximate $\nu'(1.4)$ and $\varphi'(0.5)$.

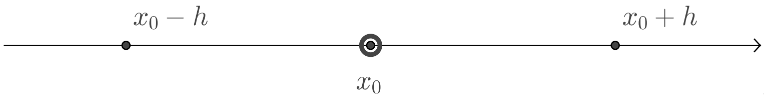
$$\begin{aligned} \nu'(1.4) &\approx \frac{-2\nu(\frac{93}{70}) + \nu(\frac{99}{70}) + \nu(\frac{105}{70})}{3(\frac{3}{35})} \\ &= \frac{-2(2.084603181618954) + 2.009751835391139 + 1.971614474758557)}{9/35} \\ &= -.7304890953430477. \end{aligned}$$

Notice this is not exactly what we got in table 4.1 for $\nu'(1.4)$ using P_2. The two estimates differ in the last few digits. This is due to floating-point error affecting the calculations in different ways. Generally there is more error in calculating directly from the interpolating polynomial because the data are processed much more heavily. Best not to trust the last several digits in either calculation, however. Now

$$\begin{aligned}\varphi'(0.5) &\approx \frac{-2\varphi(\frac{1}{7})+\varphi(\frac{4}{7})+\varphi(1)}{3(\frac{3}{7})}\\ &= \frac{-2(1.9498808918992)+1.941460911122824+1.96122825291126)}{9/7}\\ &= .002276851294420679.\end{aligned}$$

Again, this is close to the approximation in table 4.1, but not exactly the same due to different floating-point errors for the two calculations. But the point is made. Using a formula based on a stencil is preferable to working directly from the interpolating polynomial. It is easier, more efficient, and can be automated.

Before moving on to integration, we make one more observation. When trying to approximate f using an interpolating polynomial, it does not make much sense to consider a stencil like



,

where the point of evaluation is one of the nodes. We know, by definition of P_n, that $P_n(x_i) = f(x_i)$ for each node x_i. Hence, the "formula" would be $f(x_i) = P_2(x_i)$, and it would be exact, not an approximation. And not particularly informative since this is one of the facts from which we calculated P_2! On the other hand, it does make sense to consider such a stencil when trying to approximate derivatives of f. There is no guarantee the derivative of P_n will agree with the derivative of f anywhere, even at the nodes. Substituting $\theta_0 = -1$, $\theta_1 = 0$, $\theta_2 = 1$, and $\theta = 0$ into 4.1.3, we find

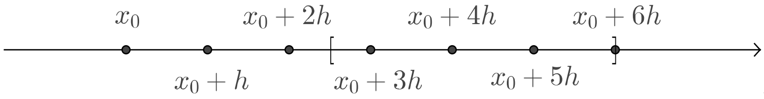
$$\begin{aligned}P_2'(x_0) &= \frac{1}{h(-1)(-2)}f(x_0-h)+\frac{1+(-1)}{h(1)(-1)}f(x_0)+\frac{1}{h(2)(1)}f(x_0+h)\\ &= \frac{f(x_0+h)-f(x_0-h)}{2h}, \qquad (4.1.5)\end{aligned}$$

for example.

Integrals

For integration formulas, we use a modified stencil. We need the nodes plus the endpoints of integration, which will be identified by square brackets, [for the left endpoint and] for the right endpoint. But the process is analogous. We find a formula for the interpolating polynomial and, in place of integrating the unknown function, we integrate the interpolating polynomial.

Following this procedure, we can derive a formula for the integral of f over the stencil



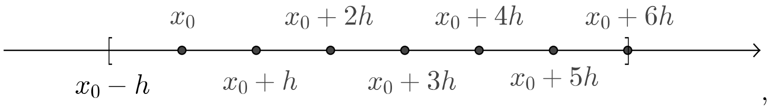
,

for example. The algebra is straightforward but tedious, so we do not show it here. It is best to use a computer algebra system to derive such a formula. The result, an approximation of the integral over $[x_0+2.5h, x_0+6h]$ using nodes x_0, x_0+h, x_0+2h, x_0+3h, x_0+4h, x_0+5h, and x_0+6h, is

$$\begin{aligned}\int_{x_0+2.5h}^{x_0+6h} f(x)dx &\approx \frac{h}{138240}\left[42056f(x_0+6h)+201831f(x_0+5h)+63357f(x_0+4h)\right.\\ &\quad \left.+195902f(x_0+3h)-28518f(x_0+2h)+10731f(x_0+h)-1519f(x_0)\right].\end{aligned}$$

This formula can now be used to approximate $\int_{1.4}^{1.5}\nu(x)dx$ instead of integrating the interpolating polynomial directly as done on page 111. You are invited to plug in the appropriate values of ν and compare your answer to the one in table on page 111. Answer on page 118.

The stencil for the approximation of $\int_0^1 \varphi(x)dx$ using $P_{6,\varphi}$ looks like



,

different from the one we used to approximate $\int_{1.4}^{1.5} \nu(x)dx$. Consequently, the approximation formula is different too. We need a formula for the integral over $[x_0-h, x_0+6h]$ with nodes x_0, x_0+h, x_0+2h, x_0+3h, x_0+4h, x_0+5h, and x_0+6h. The nodes are the same as before, but the interval of integration is different. The result is

$$\begin{aligned}\int_{x_0-h}^{x_0+6h} f(x)dx \quad &\approx \quad \frac{h}{8640}\left[5257f(x_0+6h) - 5880f(x_0+5h) + 59829f(x_0+4h)\right. \\ &\left. -81536f(x_0+3h) + 102459f(x_0+2h) - 50568f(x_0+h) + 30919f(x_0)\right]. \qquad (4.1.6)\end{aligned}$$

Again, a computer algebra system should be used to derive such a formula. You are now invited to plug in the appropriate values of φ to approximate $\int_0^1 \varphi(x)dx$ and compare your result to the one in table on page 111. Answer on page 118.

Key Concepts

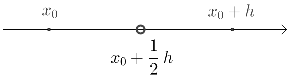
node: the abscissa (first coordinate) of a data point used in interpolation.

polynomial approximation: approximating the value of a function, its derivative or integral based on the corresponding value of an interpolating polynomial.

stencil: relative positioning of the abscissas used in a polynomial approximation.

Exercises

1. Derive an approximation formula for the first derivative over the stencil



following these steps. [S]

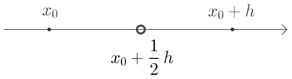
(a) Write down $L_1(x)$, the Lagrange form of the interpolating polynomial passing through the points

$$(x_0, f(x_0)) \quad \text{and} \quad (x_1, f(x_1)).$$

(b) Calculate the derivative $L_1'(x)$.

(c) Substitute $x_0+\frac{1}{2}h$ for x and x_0+h for x_1 in your formula from (b) and simplify.

2. Derive an approximation formula for the first derivative over the stencil



following these steps.

(a) Write down $L_1(x(\theta)) = L_1(x_0+\theta h)$, the Lagrange form of the interpolating polynomial passing through the points

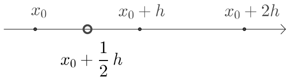
$$(x_0, f(x_0)) \quad \text{and} \quad (x_0+h, f(x_0+h))$$

in terms of θ, h, and x_0.

(b) Calculate the derivative $\frac{d}{dx}L_1(x(\theta))$. Remember, $x(\theta) = x_0+\theta h$, and use the chain rule.

(c) Substitute $\theta = \frac{1}{2}$ into your formula from (b) and simplify. [A]

3. Derive an approximation formula for the first derivative over the stencil



following these steps.

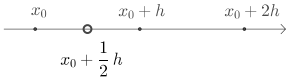
(a) Calculate $N_2(x)$, the Newton form of the interpolating polynomial passing through the points

$$(x_0, f(x_0)),\ (x_1, f(x_1)),\ \text{and } (x_2, f(x_2)).$$

(b) Calculate the derivative $N_2'(x)$.

(c) Substitute $x_0+\frac{1}{2}h$ for x, x_0+h for x_1, and x_0+2h for x_2 in your formula from (b) and simplify. [A]

4. Derive an approximation formula for the second derivative over the stencil



following these steps. [S]

(a) Calculate $N_2(x(\theta)) = N_2(x_0+\theta h)$, the Newton form of the interpolating polynomial passing through the points

$$(x_0, f(x_0)),\ (x_0+h, f(x_0+h)),$$
$$\text{and } (x_0+2h, f(x_0+2h))$$

in terms of θ, h, and x_0.

(b) Calculate the derivative $\frac{d^2}{dx^2}N_2(x(\theta))$. Remember, $x(\theta) = x_0+\theta h$, and use the chain rule.

(c) Substitute $\theta = \frac{1}{2}$ into your formula from (b) and simplify.

5. Formula 4.1.5 and the formula you got from question 1 should be different. However, they were derived over essentially the same stencil—two nodes with the point of evaluation centered between them. Only the labels on the stencils were different. In other words, they were derived from the same geometry, so, in some sense, must be the same. In question 1, x_0 plays the same role as $x_0 - h$ does in 4.1.5. Moreover, in question 1, the distance from the point of evaluation to either node is $\frac{h}{2}$ while in 4.1.5, that distance is h. Make the substitution x_0 for $x_0 - h$ in 4.1.5. Then make the substitution $\frac{h}{2}$ for the h in the denominator of 4.1.5. With these substitutions, formula 4.1.5 should match exactly the formula you got in question 1. In other words, different labelings in a stencil produce different labelings in the associated formula. Nothing more.

6. Use formula 4.1.6 to approximate the integral.

   (a) $\displaystyle\int_{-4}^{3} e^x dx$ [A]

   (b) $\displaystyle\int_{-1}^{6} \sin x\, dx$

   (c) $\displaystyle\int_{10}^{17} \frac{1}{x-5} dx$ [S]

   (d) $\displaystyle\int_{-3}^{4} \left(x^5 - 4\right) dx$

   (e) $\displaystyle\int_{0}^{1} e^{-x} dx$ [A]

   (f) $\displaystyle\int_{-\pi/2}^{\pi/2} \cos x\, dx$

   (g) $\displaystyle\int_{1}^{2} \frac{1}{x} dx$ [A]

   (h) $\displaystyle\int_{4}^{6.1} \left(9 - x^4\right) dx$

7. For each integral in question 6, (i) calculate the integral exactly, and (ii) calculate the absolute error in the approximation. [S] [A]

8. Let $f(x) = (x-1)^2 \sin x$. Use formula 4.1.4 to approximate $f'(0)$ using

   (a) $h = 1$

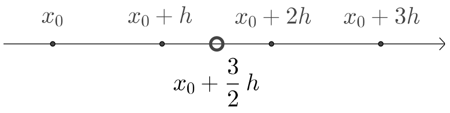
   (b) $h = \frac{1}{2}$ [A]

   (c) $h = \frac{1}{4}$

   (d) $h = \frac{1}{8}$

9. Calculate the absolute error in each approximation of question 8. Does the error get smaller as h gets smaller? [A]

10. Derive an approximation formula over the stencil



   (a) for the value of the function.

   (b) for the first derivative.

   (c) for the second derivative.

   (d) for the third derivative. What can you say about this formula?

11. The polynomial $p(x) = 3x^4 - 2x^2 + x - 7$ is an interpolating polynomial for f. Use p to approximate

   (a) $f(1)$

   (b) $f(2)$ [A]

   (c) $f'(1)$

   (d) $f'(2)$ [S]

   (e) $\displaystyle\int_0^1 f(x)dx$

   (f) $\displaystyle\int_0^2 f(x)dx$ [A]

12. The polynomial $q(x) = -7x^4 + 3x^2 - x + 4$ is an interpolating polynomial for g. Use q to approximate

   (a) $g(1)$ [A]

   (b) $g(2)$

   (c) $g'(1)$ [A]

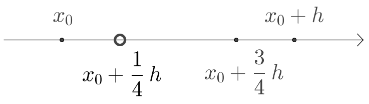
   (d) $g'(2)$

   (e) $\displaystyle\int_0^1 g(x)dx$ [S]

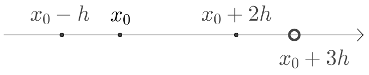
   (f) $\displaystyle\int_0^2 g(x)dx$

13. Use 4.1.3 to find the formula for the first derivative over the stencil

   (a)

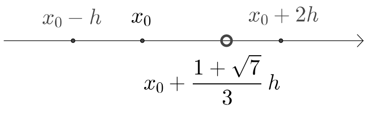


   (b)

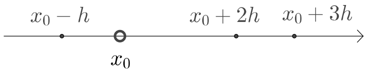


[A]

   (c)

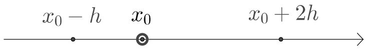


   (d)

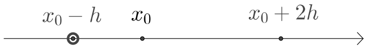


[S]

   (e)

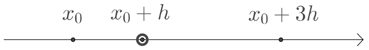


   (f)

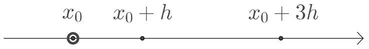


[A]

   (g)



   (h)



[A]

14. Find a general approximation formula for the integral using two nodes by doing the following.

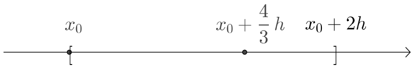
(a) Write down the (linear) interpolating polynomial with nodes $x_0 + \theta_2 h$ and $x_0 + \theta_3 h$.

(b) Integrate the polynomial over the interval $[x_0 + \theta_0 h, x_0 + \theta_1 h]$.

(c) Simplify. [A]

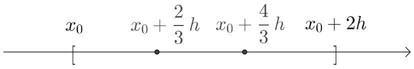
15. Use the general approximation formula you derived in question 14 to find an approximation formula over the stencil.

(a)

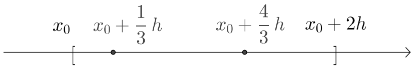


[A]

(b)

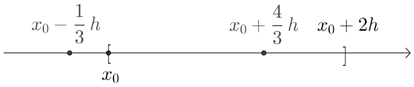


(c)

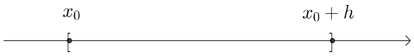


[S]

(d)



(e)



[A]

16. A general three point formula for the first derivative using $f(x_0)$, $f(x_0 + \alpha h)$, and $f(x_0 + 2h)$, $\alpha \neq 0$ and $\alpha \neq 2$, is given by

$$
\begin{aligned}
f'(x_0) \quad = \quad & \frac{1}{2h}\Big[-\frac{2+\alpha}{\alpha} f(x_0) \\
& + \frac{4}{\alpha(2-\alpha)} f(x_0 + \alpha h) \\
& - \frac{\alpha}{2-\alpha} f(x_0 + 2h)\Big] + O(h^2)
\end{aligned}
$$

Use Taylor expansions of $f(x_0 + \alpha h)$ and $f(x_0 + 2h)$ to derive the given formula.

Answers

$\int_{x_0+2.5h}^{x_0+6h} f(x)dx$:

$$
\begin{gathered}
\frac{1/35}{138240}\,[42056(1.971614474758557) + 201831(1.981091507449763) \\
+63357(1.993574976724822) + 195902(2.009751835391139) \\
-28518(2.030278824314539) + 10731(2.055494116570853) \\
-1519(2.084603181618954)]
\end{gathered}
$$

$\int_{x_0-h}^{x_0+6h} f(x)dx$**:**

$$
\begin{gathered}
\frac{1/7}{8640}\,[5257(1.96122825291126) - 5880(1.965674866641883) \\
+59829(1.960870620285721) - 81536(1.941460911122824) \\
+102459(1.923339403354019) - 50568(1.951091775564697) \\
+30919(1.9498808918992)]
\end{gathered}
$$

4.2 Undetermined Coefficients

The basic idea

According to equation 3.2.3, the difference between f and an interpolating polynomial is a multiple of $f^{(n+1)}(\xi_x)$. In other words, the error in approximating f by the interpolating polynomial P_n depends directly on $f^{(n+1)}$. But $f^{(n+1)}(x)$ is identically zero whenever f is a polynomial of degree less than $n+1$. Consequently, $(f - P_n)(x)$ is identically zero in this case. At the risk of sounding redundant, this last thought is worthy of repeating. If f is any polynomial of degree less than $n+1$, then P_n, computed for any set of $n+1$ nodes, equals f exactly, for all x. As a result, derivatives of P_n and integrals of P_n are not just approximations of the corresponding derivatives and integrals of f. They are exact because $P_n = f$ for all x. This observation can be used to derive formulas for derivatives and integrals without ever computing P_n or its derivatives or integrals!

All the formulas we have been deriving for approximating derivatives and integrals of the arbitrary function f have taken the form

$$\sum_{i=0}^{n} a_i f(x_i)$$

where $x_0, x_1, \ldots, x_n$ are the nodes of the interpolating polynomial, places where the value of f is known, and the a_i are constants resulting from the derivation. The Method of Undetermined Coefficients takes a direct approach to calculating the constants a_i. Knowing that the "approximation" formula must be exact for all polynomials of degree $0, 1, \ldots, n$, we can create $n+1$ equations in the $n+1$ unknowns, $a_0, a_1, \ldots, a_n$. The solution of the resulting system of equations gives the values of the coefficients.

Derivatives

We seek an approximation of the k^{th} derivative of f based on knowledge of the values $f(x_0 + \theta_0 h), f(x_0 + \theta_1 h), \ldots, f(x_0 + \theta_n h)$. To be precise, we desire an approximation of the form

$$f^{(k)}(x_0 + \theta h) \approx \sum_{i=0}^{n} a_i f(x_0 + \theta_i h). \tag{4.2.1}$$

Due to equation 3.2.3, the approximation must be exact for all polynomials of degree n or less. In particular, it must be exact for the polynomials $p_j(x) = (x - x_0)^j$, $j = 0, 1, \ldots, n$. Symbolically, it must be that

$$p_j^{(k)}(x_0 + \theta h) = \sum_{i=0}^{n} a_i p_j(x_0 + \theta_i h)$$

for $j = 0, 1, \ldots, n$. Notice the approximation has become an (*exact*) equality. Noting that $p_j(x_0 + \theta_i h) = ((x_0 + \theta_i h) - x_0)^j = (\theta_i h)^j$, the system of equations becomes

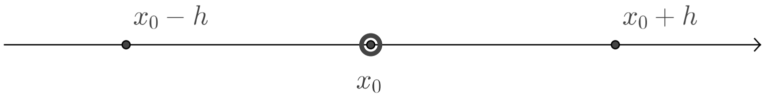
$$p_j^{(k)}(x_0 + \theta h) = a_0 + \sum_{i=1}^{n} (\theta_i h)^j a_i \tag{4.2.2}$$

for $j = 0, 1, \ldots, n$. It is the solution of this system that will yield the a_i.

Crumpet 24: Vandermonde Matrices

In general, a system of linear equations may have zero, one, or many solutions. However, system 4.2.2 has a special form. In each equation, the constants $(\theta_i h)^j$ form a geometric progression. Such a matrix of coefficients is called a Vandermonde matrix, and it is known that as long as the θ_i are distinct, this system will have one solution.

To illustrate, suppose we have the stencil



and are interested in formulas for both the first and second derivatives of f (at x_0). For this stencil, $\theta = 0$, $\theta_0 = -1$, $\theta_1 = 0$, and $\theta_2 = 1$, so we are looking for formulas of the forms

$$\begin{aligned} f'(x_0) &\approx a_0 f(x_0 - h) + a_1 f(x_0) + a_2 f(x_0 + h) \\ &\text{and} \\ f''(x_0) &\approx b_0 f(x_0 - h) + b_1 f(x_0) + b_2 f(x_0 + h). \end{aligned}$$

Each of these formulas must be exact when $f = p_0$, when $f = p_1$, and when $f = p_2$. These three requirements give three equations in the three unknowns.

Beginning with the first derivative formula, we detail system 4.2.2 with $k = 1$ and $n = 2$:

$$\begin{aligned} p_0'(x_0) &= a_0 p_0(x_0 - h) + a_1 p_0(x_0) + a_2 p_0(x_0 + h) \\ p_1'(x_0) &= a_0 p_1(x_0 - h) + a_1 p_1(x_0) + a_2 p_1(x_0 + h) \\ p_2'(x_0) &= a_0 p_2(x_0 - h) + a_1 p_2(x_0) + a_2 p_2(x_0 + h) \end{aligned}$$

By definition, $p_0(x) = (x - x_0)^0 = 1$ so $p_0'(x_0) = 0$; $p_1(x) = (x - x_0)^1 = x - x_0$ so $p_1'(x_0) = 1$; and $p_2(x) = (x - x_0)^2$ so $p_2'(x) = 2(x - x_0)$ giving $p_2'(x_0) = 0$. Substituting this information into the equations above,

$$\begin{aligned} 0 &= a_0 + a_1 + a_2 \\ 1 &= -ha_0 + ha_2 \\ 0 &= h^2 a_0 + h^2 a_2. \end{aligned}$$

The system can be solved by substitution, elimination, or computer algebra system. The solution is $a_0 = \frac{-1}{2h}$, $a_1 = 0$, and $a_2 = \frac{1}{2h}$, giving the approximation formula

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

just as we got on page 115 in formula 4.1.5.

The second derivative formula is derived in the same manner. Since the second derivative formula must be exact when $f = p_0$, when $f = p_1$, and when $f = p_2$, the a_i must satisfy

$$\begin{aligned} p_0''(x_0) &= b_0 p_0(x_0 - h) + b_1 p_0(x_0) + b_2 p_0(x_0 + h) \\ p_1''(x_0) &= b_0 p_1(x_0 - h) + b_1 p_1(x_0) + b_2 p_1(x_0 + h) \\ p_2''(x_0) &= b_0 p_2(x_0 - h) + b_1 p_2(x_0) + b_2 p_2(x_0 + h), \end{aligned}$$

system 4.2.2 with $k = 2$ and $n = 2$. Notice the right-hand sides are exactly the same as they are for the first derivative formula, save the name change from a_i to b_i. Only the left-hand side changes substantively. $p_0''(x) = 0$ so $p_0''(x_0) = 0$; $p_1''(x) = 0$ so $p_1(x_0) = 0$; and $p_2''(x) = 2$ so $p_2''(x_0) = 2$. Making these substitutions into the equations above,

$$\begin{aligned} 0 &= b_0 + b_1 + b_2 \\ 0 &= -hb_0 + hb_2 \\ 2 &= h^2 b_0 + h^2 b_2. \end{aligned}$$

Again, the system can be solved by substitution, elimination, or computer algebra system. The solution is $b_0 = b_2 = \frac{1}{h^2}$ and $b_1 = \frac{2}{h^2}$, giving the approximation formula

$$f''(x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}.$$

Integrals

The idea for estimating integrals is identical to that of estimating derivatives. The mechanics only change nominally. Where there were derivatives before, we will have integrals now. We seek an approximation of $\int_a^b f(x)dx$ based on knowledge of the values $f(x_0+\theta_0 h), f(x_0+\theta_1 h), \ldots, f(x_0+\theta_n h)$:

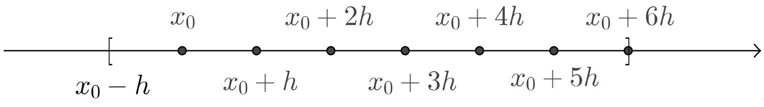
$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_0+\theta_i h). \tag{4.2.3}$$

The approximation will be exact for all polynomials of degree n or less. In particular, it will be exact for $p_j(x) = (x-x_0)^j$, $j = 0, 1, \ldots, n$. Therefore, the system of equations

$$\int_a^b p_j(x)dx = a_0 + \sum_{i=1}^{n} (\theta_i h)^j a_i \qquad j = 0, 1, \ldots, n \tag{4.2.4}$$

must be satisfied by the a_i.

To illustrate, suppose we have the stencil



.

For this stencil, $a = x_0 - h$, $b = x_0 + 6h$, and $\theta_i = ih$, $i = 0, 1, \ldots, 6$. Therefore, we will have a system of seven equations in the seven unknowns. First, the left-hand sides:

$$\begin{aligned}
\int_a^b p_0(x)dx = \int_{x_0-h}^{x_0+6h} p_0(x)dx &= \int_{x_0-h}^{x_0+6h} 1dx = (x-x_0)\Big|_{x_0-h}^{x_0+6h} = 7h \\
\int_a^b p_1(x)dx = \int_{x_0-h}^{x_0+6h} p_1(x)dx &= \int_{x_0-h}^{x_0+6h} (x-x_0)dx = \frac{1}{2}(x-x_0)^2\Big|_{x_0-h}^{x_0+6h} = \frac{35}{2}h^2 \\
\int_a^b p_2(x)dx = \int_{x_0-h}^{x_0+6h} p_2(x)dx &= \int_{x_0-h}^{x_0+6h} (x-x_0)^2dx = \frac{1}{3}(x-x_0)^3\Big|_{x_0-h}^{x_0+6h} = \frac{217}{3}h^3 \\
\int_a^b p_3(x)dx = \int_{x_0-h}^{x_0+6h} p_3(x)dx &= \int_{x_0-h}^{x_0+6h} (x-x_0)^3dx = \frac{1}{4}(x-x_0)^4\Big|_{x_0-h}^{x_0+6h} = \frac{1295}{4}h^4 \\
\int_a^b p_4(x)dx = \int_{x_0-h}^{x_0+6h} p_4(x)dx &= \int_{x_0-h}^{x_0+6h} (x-x_0)^4dx = \frac{1}{5}(x-x_0)^5\Big|_{x_0-h}^{x_0+6h} = \frac{7777}{5}h^5 \\
\int_a^b p_5(x)dx = \int_{x_0-h}^{x_0+6h} p_5(x)dx &= \int_{x_0-h}^{x_0+6h} (x-x_0)^5dx = \frac{1}{6}(x-x_0)^6\Big|_{x_0-h}^{x_0+6h} = \frac{46655}{6}h^6 \\
\int_a^b p_6(x)dx = \int_{x_0-h}^{x_0+6h} p_6(x)dx &= \int_{x_0-h}^{x_0+6h} (x-x_0)^6dx = \frac{1}{7}(x-x_0)^7\Big|_{x_0-h}^{x_0+6h} = 39991h^7.
\end{aligned}$$

Now putting them together with the right-hand sides (and swapping sides):

$$\begin{aligned}
\sum_{i=0}^{6}(\theta_i h)^0 a_i &= a_0+a_1+a_2+a_3+a_4+a_5+a_6 = 7h\\
\sum_{i=0}^{6}(\theta_i h)^1 a_i &= ha_1+2ha_2+3ha_3+4ha_4+5ha_5+6ha_6 = \frac{35}{2}h^2\\
\sum_{i=0}^{6}(\theta_i h)^2 a_i &= h^2a_1+4h^2a_2+9h^2a_3+16h^2a_4+25h^2a_5+36h^2a_6 = \frac{217}{3}h^3\\
\sum_{i=0}^{6}(\theta_i h)^3 a_i &= h^3a_1+8h^3a_2+27h^3a_3+64h^3a_4+125h^3a_5+216h^3a_6 = \frac{1295}{4}h^4\\
\sum_{i=0}^{6}(\theta_i h)^4 a_i &= h^4a_1+16h^4a_2+81h^4a_3+256h^4a_4+625h^4a_5+1296h^4a_6 = \frac{7777}{5}h^5\\
\sum_{i=0}^{6}(\theta_i h)^5 a_i &= h^5a_1+32h^5a_2+243h^5a_3+1024h^5a_4+3125h^5a_5+7776h^5a_6 = \frac{46655}{6}h^6\\
\sum_{i=0}^{6}(\theta_i h)^6 a_i &= h^6a_1+64h^6a_2+729h^6a_3+4096h^6a_4+15625h^6a_5+46656h^6a_6 = 39991h^7
\end{aligned}$$

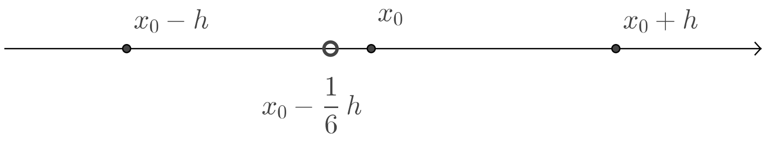
The system again may be solved by substitution, elimination, or computer algebra, at least in principle. Not many humans have sufficient patience and precision to solve such a system with paper and pencil, though. Trusting a computer algebra system, the solution is $a_0 = \frac{30919}{8640}h$, $a_1 = -\frac{2107}{360}h$, $a_2 = \frac{34153}{2880}h$, $a_3 = -\frac{1274}{135}h$, $a_4 = \frac{19943}{2880}h$, $a_5 = -\frac{49}{72}h$, and $a_6 = \frac{5257}{8640}h$ giving the approximation formula

$$\begin{aligned}
\int_{x_0-h}^{x_0+6h} f(x)dx \quad &\approx \quad \frac{h}{8640}\left[5257f(x_0+6h)-5880f(x_0+5h)+59829f(x_0+4h)-81536f(x_0+3h)\right.\\
&\qquad \left.+102459f(x_0+2h)-50568f(x_0+h)+30919f(x_0)\right] \qquad (4.2.5)
\end{aligned}$$

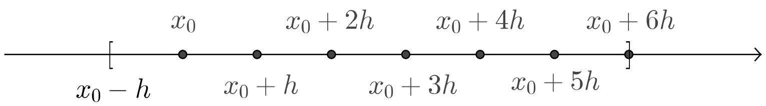
just as we got on page 116 in formula 4.1.6.

Practical considerations

We have used stencils like



and



not because the results are particularly helpful, but rather to (a) illustrate the methods and (b) emphasize that these methods work in general for any stencil you may dream up. Most of the differentiation and integration formulas presented in numerical analysis sources stick to a small host of regularly spaced stencils where, for derivatives the point of evaluation is a node, and for integrals, all the nodes lie between the endpoints or there are nodes at both endpoints. It is possible the regularly-spaced stencils are all you will ever need, but it is good to know that you can derive appropriate formulas for more unusual stencils should the need arise.

As for their derivation, the main advantage of the method of undetermined coefficients over working directly with interpolating polynomials is the ease of automation and lessening of the necessary and often laborious algebra needed. In the method of undetermined coefficients, the only polynomials that need to be differentiated or integrated are the polynomials $p_j = (x-x_0)^j$, a much simpler task than integrating or differentiating interpolating polynomials. Formulas with up to three or four nodes can be handled this way with pencil and paper. The trade-off is the necessity of solving a system of equations, again a simpler task than differentiating and simplifying interpolating polynomials of degree 3 or 4. As a final benefit to the method of undetermined coefficients, it is a general solution technique used not only in numerical analysis for deriving calculus approximations, but in other studies as well, particularly differential equations. The method is applicable whenever the *form* of a solution or formula is known, but the constants (coefficients) remain a mystery.

Crumpet 25: Undetermined Coefficients in Differential Equations

In differential equations, we know that a particular solution of the equation

$$y - 2y' + 3y'' = 5\sin x \tag{4.2.6}$$

has the form $y = A\sin x + B\cos x$, but we do not immediately know the values of A and B. They are undetermined coefficients (at this point). They are determined by substituting the known form into the equation being solved.

$$\begin{aligned} y' &= A\cos x - B\sin x \\ y'' &= -A\sin x - B\cos x \end{aligned}$$

So the equation being solved becomes

$$(A\sin x + B\cos x) - 2(A\cos x - B\sin x) + 3(-A\sin x - B\cos x) = 5\sin x.$$

Collecting the coefficients of $\sin x$ and $\cos x$ on the left side,

$$(-2A + 2B)\sin x + (-2A - 2B)\cos x = 5\sin x.$$

We now match coefficients on left and right sides to get the system of equations

$$\begin{aligned} -2A + 2B &= 5 \\ -2A - 2B &= 0 \end{aligned}$$

whose solution is $A = -\frac{5}{4}$ and $B = \frac{5}{4}$. Therefore, $y = -\frac{5}{4}\sin x + \frac{5}{4}\cos x$ solves equation 4.2.6.

Conceptually, this process is no different from the method of undetermined coefficients used in deriving numerical calculus formulas. The solution to some problem is known, save for some (undetermined) coefficients. The parameters of the problem require the coefficients to satisfy some system of linear equations. The system is solved, and the solution to the original problem is consequently known completely, coefficients determined.

When we get involved with stencils with more than 3 or 4 nodes, solving the resulting (relatively large) system of linear equations by hand is not a task to which most of us would look forward. However, it is a standard calculation any computer algebra system can do easily and efficiently. Yes, it is advisable to use a computer algebra system to derive formulas as complicated as 4.1.6. We have used Maxima[1] to handle or double check a number of the more tedious calculations presented in this text.

Reference

It is unusual to use stencils with more than five nodes anyway. It is not because the formulas for more nodes are significantly more complicated or difficult to use, however. As evidenced by formula 3.2.3, the error term for an interpolating polynomial involves higher and higher derivatives of f as more nodes are added. This is generally fine as long as f has sufficiently many derivatives and the values of the high derivatives are not prohibitively large. However, numerical methods are often employed when the smoothness of f is known to be limited, the high derivatives are known to be large, or the properties of its derivatives are unknown completely. For these functions, stencils with fewer nodes, which give rise to formulas with lower order error terms, are often *more accurate*, not less. And in the case of unknown smoothness, the lower order methods have a better chance of being accurate.

[1] See http://maxima.sourceforge.net/

As a final note, some care must be taken not to ask too much of a derivative formula. With $n+1$ nodes, the error term for the interpolating polynomial involves $f^{(n+1)}$, so there is no hope of using these nodes to estimate $f^{(n+1)}$ or any higher derivatives at any point. If you, however, forget this fact, it shows up in a direct way in the method of undetermined coefficients. If $k > n$, then the system of equations with undetermined coefficients becomes

$$\sum_{i=0}^{n} (\theta_i h)^j a_i = 0, \qquad j = 0, 1, \ldots, n$$

because the k^{th} derivative of p_j is identically 0 for all $j \le n < k$. The only solution to this system is $a_0 = a_1 = \cdots = a_n = 0$ giving the "approximation" formula

$$f^{(k)}(x_0 + \theta h) = 0.$$

Indeed, this is exact for all polynomials of degree n or less. However, the error in using this formula is exactly $f^{(k)}(x_0 + \theta h)$, a relative error of exactly 1, making it completely useless.

Stability

In Experiment 2 on page 3, section 1.1, we took a brief look at approximating the first derivative of $f(x) = \sin x$ using the fact that

$$f'(1) = \lim_{h \to 0} \frac{\sin(1+h) - \sin(1-h)}{2h}.$$

The conclusion we drew was that this computation was highly susceptible to floating-point error. If calculations are done exactly, then we expect $\frac{\sin(1+h)-\sin(1-h)}{2h}$ to approximate $f'(1)$ better and better as h becomes smaller and smaller. Not so for floating-point calculations, as the experiment revealed. There was a point at which making h smaller made the approximation worse! And this example is not unique. This problem always arises when approximating f' using the centered difference formula

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{4.2.7}$$

But how can we predict at what value of h that might happen without comparing our results to the exact value of the derivative? After all, numerical differentiation is employed most often when the exact formula for the derivative is unknown or prohibitively difficult to compute.

Suppose f can be computed to near machine precision. In typical floating point calculations that means a relative floating-point error of approximately 10^{-15} or absolute floating-point error $\varepsilon_f \approx 10^{-15}|f(x)|$. Since we assume h is small, we can approximate both $|\tilde{f}(x+h) - f(x+h)|$ and $|\tilde{f}(x-h) - f(x-h)|$ by ε_f giving an absolute error of approximately $2\varepsilon_f$ in calculating the numerator $f(x+h) - f(x-h)$. Assuming h is calculated exactly, we have the absolute error

$$\varepsilon_r = |\tilde{f}'(x) - f'(x)| \approx \frac{2\varepsilon_f}{2h} = \frac{\varepsilon_f}{h} = \frac{|f(x)|}{10^{15}} \cdot \frac{1}{h}. \tag{4.2.8}$$

As we will see shortly, the algorithmic error, ε_a, is caused by truncation and equals $\left|\frac{f'''(\xi)}{6} h^2\right|$ for some value of ξ near x. Since ξ is near x, we approximate $f'''(\xi)$ by $f'''(x)$ and conclude that

$$\varepsilon_a \approx \frac{|f'''(x)|}{6} h^2. \tag{4.2.9}$$

We now minimize the value of $\varepsilon_r + \varepsilon_a$ by setting its derivative (with respect to h) equal to zero and solving the resulting equation:

$$\begin{aligned}
0 = \frac{d}{dh}(\varepsilon_r + \varepsilon_a) &\approx \frac{d}{dh}\left(\frac{|f(x)|}{10^{15}} \cdot \frac{1}{h} + \frac{|f'''(x)|}{6} \cdot h^2\right) \\
&= -\frac{|f(x)|}{10^{15}} \cdot \frac{1}{h^2} + \frac{|f'''(x)|}{3} \cdot h \\
&\Rightarrow \\
\frac{|f'''(x)|}{3} \cdot h &\approx \frac{|f(x)|}{10^{15}} \cdot \frac{1}{h^2} \\
h^3 &\approx \frac{|f(x)|}{|f'''(x)|} \cdot \frac{3}{10^{15}} \\
h &\approx \sqrt[3]{\frac{3|f(x)|}{|f'''(x)|}} \cdot 10^{-5}.
\end{aligned}$$

For Experiment 2 on page 3, this means we should expect the optimal value of h to be around $\sqrt[3]{\frac{3\sin(1)}{\sin(1)}} \cdot 10^{-5} \approx 1.44(10)^{-5}$. We reproduce the table from Experiment 2 here with the addition of a third column, the actual absolute error:

h	$\tilde{p}^*(h)$	$\lvert\tilde{p}^*(h) - f'(1)\rvert$
10^{-2}	0.5402933008747335	$9.00(10)^{-6}$
10^{-3}	0.5403022158176896	$9.00(10)^{-8}$
10^{-4}	0.5403023049677103	$9.00(10)^{-10}$
10^{-5}	0.5403023058569989	$1.11(10)^{-11}$
10^{-6}	0.5403023058958567	$2.77(10)^{-11}$
10^{-7}	0.5403023056738121	$1.94(10)^{-10}$

Indeed, when $h = 10^{-5}$, we get our best results! However, the prediction of the optimal value of h was based on knowledge of f''', something we generally will not be able to do. Unless we happen to know that $\frac{|f(x)|}{|f'''(x)|}$ is far from 1, we assume it is reasonably close to 1, in which case the optimal value of h is around 10^{-5}. Similar estimates can be made for other derivative formulas.

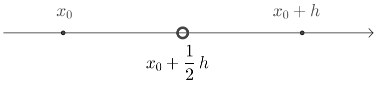
Because numerical differentiation is so sensitive to floating-point error, we say that it is unstable. The root finding methods and numerical integration we have discussed are all stable methods. Their sensitivity to floating-point error is commensurate with that of calculating f.

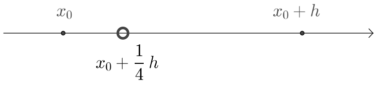
Key Concepts

undetermined coefficients: A method for solving problems in which the solution is known save for a set of (undetermined) coefficients.

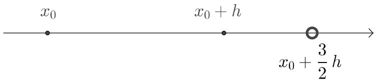
Exercises

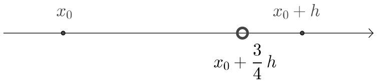
1. Using the method of undetermined coefficients, derive an approximation formula for the first derivative over the stencil.

(a)


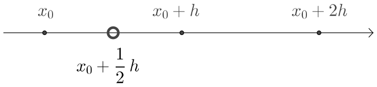
(b)


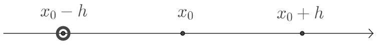
[A]

(c)


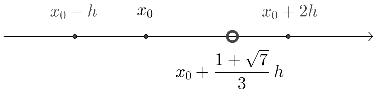
(d)


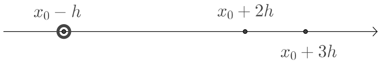
[S]

(e)


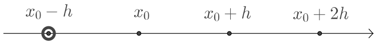
(f)


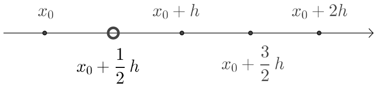
[A]

(g)


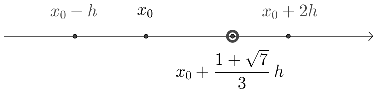
(h)


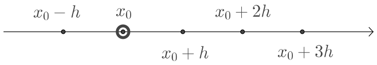
[A]

(i)


(j)


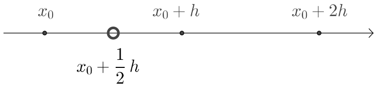
[S]

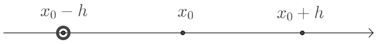
(k)


(l)


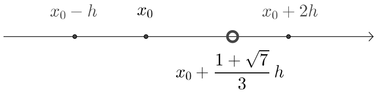
[A]

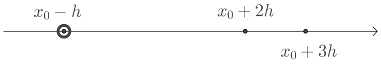
2. Using the method of undetermined coefficients, derive an approximation formula for the second derivative over the stencil.

(a)


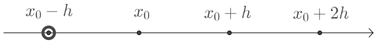
(b)


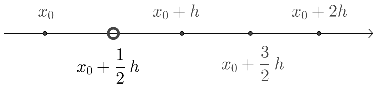
[A]

(c)


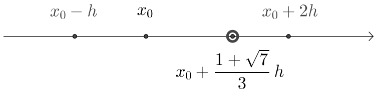
(d)


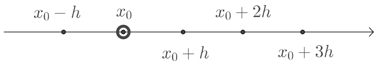
[A]

(e)


(f)


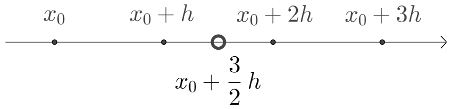
[S]

(g)


(h) 

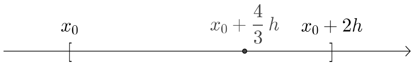
[A]

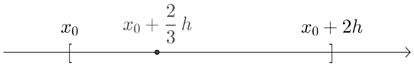
3. Use the method of undetermined coefficients to derive an approximation formula over the stencil



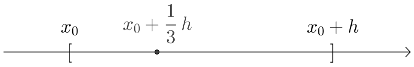
   (a) for the value of the function.
   (b) for the first derivative.
   (c) for the second derivative.
   (d) for the third derivative. What can you say about this formula?
   (e) compare the method of undetermined coefficients to the direct method employed in question 10 of section 4.1.

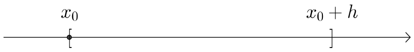
4. Use the method of undetermined coefficients to derive an approximation formula for the integral over the stencil.

   (a) 

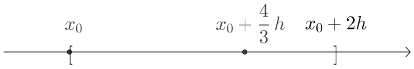
   (b) 

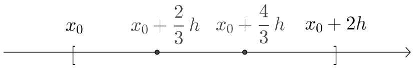
   [S]

   (c) 

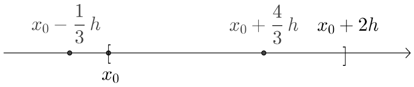
   (d) 

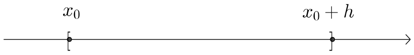
   [A]

   (e) 

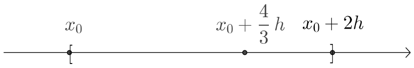
   (f) 

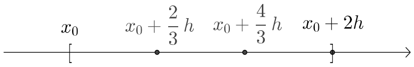
   [A]

   (g) 

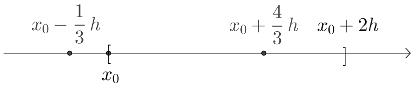
   (h) 

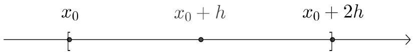
   [A]

   (i) 

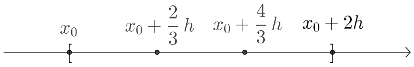
   (j) 

   [A]

   (k) 

   (l) 

   [S]

   (m) 

5. Using the method of undetermined coefficients, find a general approximation formula for $\int_{x_0+\theta_0 h}^{x_0+\theta_1 h} f(x)dx$ using the two nodes $x_0 + \theta_2 h$ and $x_0 + \theta_3 h$.

4.3 Error Analysis

Errors for first derivative formulas

In section 3.2, we found that if f has sufficient derivatives, then f and P_n, an interpolating polynomial of degree at most n, differ according to equation 3.2.3 on page 90, copied here for convenience:

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n).$$

We can use this formula to derive a concise formula for the error in approximating $f'(x)$ by $P_n'(x)$.

As done in section 3.2, suppose $n \geq 1$ and $x_0, x_1, \ldots, x_n$ are n distinct real numbers. Set $w(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$, $a = \min(x_0, \ldots, x_n, x)$, and $b = \max(x_0, \ldots, x_n, x)$. We know from equation 3.2.3 that, assuming f has $n+1$ derivatives on (a, b) and $f', f'', \ldots, f^{(n)}$ are all continuous on $[a, b]$, for each $x \in [a, b]$,

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}w(x)$$

for some $\xi_x \in (a, b)$. Hence,

$$f'(x) - P_n'(x) = \frac{d}{dx}\left[\frac{f^{(n+1)}(\xi_x)}{(n+1)!}\right]w(x) + \frac{f^{(n+1)}(\xi_x)}{(n+1)!}w'(x).$$

Since w vanishes at each node, this formula simplifies nicely when x is a node. Without loss of generality, we evaluate for $x = x_0$ and get

$$f'(x_0) - P_n'(x_0) = \frac{f^{(n+1)}(\xi_{x_0})}{(n+1)!}w'(x_0).$$

From here on, the error formula is only valid at a node! This last expression can be simplified further by noting that

$$w'(x) = \sum_{i=0}^{n}\prod_{\substack{j=0\\ i\neq j}}^{n}(x - x_j) = \sum_{i=0}^{n} p_i(x),$$

where p_i is as defined for equation 3.2.2 on page 89. But $p_i(x_0) = 0$ for all i except $i = 0$, so

$$w'(x_0) = p_0(x_0) = (x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n).$$

Substituting this expression for w', we have the first derivative error formula

$$f'(x_0) - P_n'(x_0) = \frac{f^{(n+1)}(\xi_{x_0})}{(n+1)!}(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n).$$

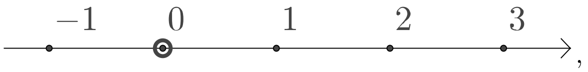
Making the substitutions $x_0 + \theta_i h$ for x_i, $i = 1, 2, \ldots, n$, to get a formula in terms of h and the θ_i:

$$f'(x_0) - P_n'(x_0) = \frac{f^{(n+1)}(\xi_{x_0})}{(n+1)!}(-\theta_1 h)(-\theta_2 h)\cdots(-\theta_n h).$$

This error formula simplifies just a bit:

$$f'(x_0) - P_n'(x_0) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\theta_1\theta_2\cdots\theta_n(-h)^n. \tag{4.3.1}$$

For the stencil



$n = 4$, $\theta_1 = -1$, $\theta_2 = 1$, $\theta_3 = 2$, and $\theta_4 = 3$, so the error in calculating f' over this stencil is

$$\frac{f^{(5)}(\xi)}{120}(-1)(1)(2)(3)(-h)^4 = -\frac{f^{(5)}(\xi)}{20}h^4.$$

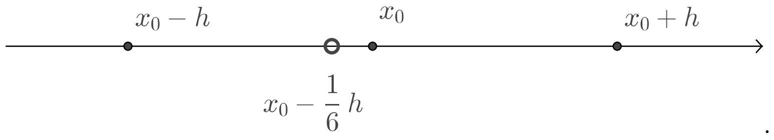
Error terms for the first derivative over other stencils are computed similarly as long as the derivative is evaluated at a node. Table 4.2 summarizes some common first derivative formulas, including error terms.

Notice that the error term contains $(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n)$, the product of the differences between the point of evaluation and all other nodes, as a factor. When the differences between the point of evaluation and the other nodes is small, the product is small. Consequently, first derivative approximation formulas are generally more accurate when the point of evaluation is centrally located among the nodes. Hence, we might expect a first derivative formula involving nodes $x_0 < x_1 < x_2$ to be more accurate when the point of evaluation is x_1 rather than when the point of evaluation is x_0 or x_2. The same can be said about higher derivative formulas. The more centrally located the point of evaluation, the more accurate the approximation.

Errors for other formulas

It is tempting to think we can simply repeat the procedure we used with first derivatives, taking the second derivative of $f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}w(x)$ to find the error for second derivative estimates, and the third derivative of $f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}w(x)$ to find the error for third derivative estimates, and so on. Alas, the matter is not so simple. Higher derivatives of $f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}w(x)$ involve derivatives of the factor $\frac{f^{(n+1)}(\xi_x)}{(n+1)!}$ which do not vanish even when x is a node. Since ξ_x is entirely unknown, so are its derivatives, making this approach unworkable. Other methods for producing precise bounds for certain higher derivative formulas or certain integral formulas are limited in scope.

There is, however, a general method for determining *good enough* error terms for any derivative or integral formula. We replace each evaluation of f in the approximation by a Taylor series expanded about x_0 and simplify. This gives an expression for the approximation in terms of $f(x_0)$, $f'(x_0)$, $f''(x_0)$, and so on. We compare it to the Taylor series representation of the quantity being estimated. The difference between the two is the error. In summary, that's it. Making a rigorous argument of this method takes some care and is worthy of an example. We demonstrate the method for the approximation of the first derivative over the stencil



.

Again, we choose this stencil not because the stencil is generally useful, but rather to emphasize that the *method* is generally useful.

In subsection 4.1 on page 114, we derived the approximation

$$f'\left(x_0 - \frac{1}{6}h\right) \approx \frac{-2f(x_0 - h) + f(x_0) + f(x_0 + h)}{3h}. \tag{4.3.2}$$

The left hand side, the quantity being approximated, as a Taylor series looks like

$$f'\left(x_0 - \frac{1}{6}h\right) = f'(x_0) - \frac{1}{6}hf''(x_0) + \frac{1}{72}h^2 f'''(x_0) - \frac{1}{1296}h^3 f^{(4)}(x_0) + \cdots.$$

The terms of the right hand side, the approximation, as Taylor series look like

$$\begin{aligned}
f(x_0 - h) &= f(x_0) - hf'(x_0) + \frac{1}{2}h^2 f''(x_0) - \frac{1}{6}h^3 f'''(x_0) + \frac{1}{24}h^4 f^{(4)}(x_0) - \cdots \\
f(x_0) &= f(x_0) \\
f(x_0 + h) &= f(x_0) + hf'(x_0) + \frac{1}{2}h^2 f''(x_0) + \frac{1}{6}h^3 f'''(x_0) + \frac{1}{24}h^4 f^{(4)}(x_0) + \cdots.
\end{aligned}$$

We now substitute these Taylor series into the right hand side of 4.3.2 and simplify. To facilitate the algebra, we begin by summing $-2f(x_0 - h) + f(x_0) + f(x_0 + h)$:

$$\begin{array}{rcl}
-2f(x_0 - h) & = & -2f(x_0) + 2hf'(x_0) - h^2 f''(x_0) + \frac{1}{3}h^3 f'''(x_0) - \frac{1}{12}h^4 f^{(4)}(x_0) - \cdots \\
f(x_0) & = & f(x_0) \\
f(x_0 + h) & = & f(x_0) + hf'(x_0) + \frac{1}{2}h^2 f''(x_0) + \frac{1}{6}h^3 f'''(x_0) + \frac{1}{24}h^4 f^{(4)}(x_0) + \cdots \\
\hline
-2f(x_0 - h) + f(x_0) + f(x_0 + h) & = & 3hf'(x_0) - \frac{1}{2}h^2 f''(x_0) + \frac{1}{2}h^3 f'''(x_0) - \frac{1}{24}h^4 f^{(4)}(x_0) + \cdots.
\end{array}$$

Hence, we have

$$\begin{aligned}\frac{-2f(x_0-h)+f(x_0)+f(x_0+h)}{3h} &= \frac{3hf'(x_0)-\frac{1}{2}h^2f''(x_0)+\frac{1}{2}h^3f'''(x_0)-\frac{1}{24}h^4f^{(4)}(x_0)+\cdots}{3h}\\ &= f'(x_0)-\frac{1}{6}hf''(x_0)+\frac{1}{6}h^2f'''(x_0)-\frac{1}{72}h^3f^{(4)}(x_0)+\cdots.\end{aligned}$$

For the error, $e(h)=f'\left(x_0-\frac{1}{6}h\right)-\frac{-2f(x_0-h)+f(x_0)+f(x_0+h)}{3h}$, we then get

$$\begin{aligned}&\left(f'(x_0)-\frac{1}{6}hf''(x_0)+\frac{1}{72}h^2f'''(x_0)-\frac{1}{1296}h^3f^{(4)}(x_0)+\cdots\right)\\ &\quad-\left(f'(x_0)-\frac{1}{6}hf''(x_0)+\frac{1}{6}h^2f'''(x_0)-\frac{1}{72}h^3f^{(4)}(x_0)+\cdots\right)\\ &=-\frac{11}{72}h^2f'''(x_0)+\frac{17}{1296}h^3f^{(4)}(x_0)+\cdots.\end{aligned}$$

We now know that we have an error of the form $O(h^2f'''(\xi_h))$, the form of the remaining term with least degree, but we do not have rigorous proof of that fact. Think of what has been done so far as discovery. Now that we know the f''' terms do not cancel, we go back and truncate all the Taylor series after the f'' terms, replacing higher order derivatives with an error term, and "redo" the algebra. We thus have

$$\begin{aligned}f'\left(x_0-\frac{1}{6}h\right) &= f'(x_0)-\frac{1}{6}hf''(x_0)+\frac{1}{72}h^2f'''(\xi_1)\\ f(x_0-h) &= f(x_0)-hf'(x_0)+\frac{1}{2}h^2f''(x_0)-\frac{1}{6}h^3f'''(\xi_2)\\ f(x_0) &= f(x_0)\\ f(x_0+h) &= f(x_0)+hf'(x_0)+\frac{1}{2}h^2f''(x_0)+\frac{1}{6}h^3f'''(\xi_3)\end{aligned}$$

where $\xi_1\in(x_0-\frac{1}{6}h,x_0)$, $\xi_2\in(x_0-h,x_0)$, and $\xi_3\in(x_0,x_0+h)$. And now when we compute $e(h)=f'\left(x_0-\frac{1}{6}h\right)-\frac{-2f(x_0-h)+f(x_0)+f(x_0+h)}{3h}$, we know all the terms involving f, f', and f'' vanish. The only terms left are those involving f''':

$$\begin{aligned}e(h) &= \frac{1}{72}h^2f'''(\xi_1)-\frac{-2(-\frac{1}{6}h^3f'''(\xi_2))+\frac{1}{6}h^3f'''(\xi_3)}{3h}\\ &= \frac{1}{72}h^2f'''(\xi_1)-\frac{1}{9}h^2f'''(\xi_2)-\frac{1}{18}h^2f'''(\xi_3)\\ &= \frac{h^2}{9}\left[\frac{1}{8}f'''(\xi_1)-f'''(\xi_2)-\frac{1}{2}f'''(\xi_3)\right].\end{aligned}$$

The final formality is that of converting this expression into big-oh notation:

$$\begin{aligned}|e(h)| &= \left|\frac{h^2}{9}\left[\frac{1}{8}f'''(\xi_1)-f'''(\xi_2)-\frac{1}{2}f'''(\xi_3)\right]\right|\\ &\le \frac{h^2}{9}\left[\left|\frac{1}{8}f'''(\xi_1)\right|+|f'''(\xi_2)|+\left|\frac{1}{2}f'''(\xi_3)\right|\right]\\ &\le \frac{h^2}{9}\cdot\frac{13}{8}\max\left\{|f'''(\xi_1)|,|f'''(\xi_2)|,|f'''(\xi_3)|\right\}\\ &= h^2\cdot M\,|f'''(\xi_h)|\end{aligned}$$

for some $\xi_h\in(x_0-h,x_0+h)$ and $M=\frac{13}{72}$ (the value of ξ_h is ξ_1, ξ_2, or ξ_3). We conclude

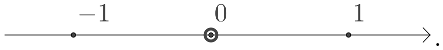
$$e(h)=O(h^2f'''(\xi_h)).$$

In general, ξ_h is guaranteed to be between the least node and the greatest node. In the case of an integral approximation, the endpoints of integration are treated as nodes for the purpose of locating ξ_h.

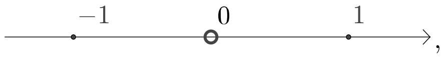
Gaussian quadrature

Ultimately, the accuracy of a numerical calculus formula is measured by its error term, a quantity having the form $O(h^n f^{(k)}(\xi_h))$. If we are interested in the rate of convergence, we consider n, the power of h appearing in the error term. The greater the power, the speedier the convergence. However, if we are interested in the largest class of polynomials for which the formula is exact, we need to consider the value k, the order of the derivative appearing in the error term. The greater k is, the larger the class of polynomials for which the formula is exact. In fact, if the error term contains a factor of $f^{(k)}(\xi_h)$, then the formula is exact for all polynomials up to (and including) degree $k-1$. The further implication is that there are degree k polynomials for which the formula is not exact, for if this were not the case, then the error term would involve a higher derivative. We call the value $k-1$ the degree of precision. Formally, the degree of precision of a numerical calculus formula is the integer m such that the formula is exact for all polynomials of degree up to and including m but is not exact for all polynomials of degree $m+1$. Gaussian quadrature formulas aim to maximize the degree of precision for integral formulas.

The numerical derivatives and integrals over a stencil with $n+1$ points that we have derived so far are exact for all polynomials up to degree n as they must be. They have degree of precision at least n. As it turns out, a select few have degree of precision greater than n. Consider the second derivative approximation over the stencil



The stencil has three points, so we expect it to be exact for all polynomials up to degree 2 (and it is). However, its error term is $O(h^2 f^{(4)}(\xi_h))$, indicating that the formula is exact for all polynomials up to degree 3. The degree of precision is actually 3, not 2. The first derivative formula over the same stencil is similar. Though it has an error term of $\frac{h^2}{6} f'''(\xi_h)$, indicating that the formula has degree of precision 2 as expected, the formula itself only involves two of the three points available! The coefficient of $f(x_0)$ turns out to be zero. It follows that we can derive the same formula using the stencil



having only two points yet having degree of precision 2. Several other centered differences have this attribute. The Newton-Cotes formulas with an odd number of nodes also have this property. Their error terms exceed degree of precision expectations by one degree. We noted earlier that a centrally located point of evaluation tends to increase accuracy, and now we see that the increase can be dramatic.

What we might gather from these observations is that it is not only the number of nodes that determines the error term of a numerical calculus formula. The location of the nodes is also important. Up to now, we have only seen how node location affects derivative approximation. We know that centrally locating the point of evaluation generally increases accuracy. We now take up the question of how to locate nodes in order to increase the accuracy of integral formulas. The idea of a centralized point of evaluation has no meaning in this context, however. Integrals do not have a single point of evaluation. They are taken over an interval. It is the locations of the nodes relative to the endpoints of evaluation that are important. We now find out where to put the nodes to attain the greatest degree of precision for any given number of nodes.

Let G_n be the n^{th} Legendre polynomial, defined recursively by

$$\begin{aligned} G_{n+1}(x) &= \frac{(2n+1)xG_n(x) - nG_{n-1}(x)}{n+1} \\ G_0(x) &= 1 \\ G_1(x) &= x. \end{aligned}$$

We set the θ_i equal to the roots of G_n to derive the n-point quadrature formula over the interval $[x_0-h, x_0+h]$ with greatest degree of precision possible. With placement of the nodes chosen, we force the formula to be exact for polynomials up to degree $n-1$ as we did earlier. The difference this time is, due to the particular values of θ_i, the resulting formula will be exact for all polynomials up to degree $2n-1$. When the nodes are placed at the roots of the n^{th} Legendre polynomial, we get a quadrature formula for $\int_{x_0-h}^{x_0+h} f(x)dx$ that exceeds the expected degree of precision by n, the number of nodes!

We demonstrate for $n=1$ and $n=3$.

$$G_1(x) = x$$

has for its only root, 0. Hence, we seek a formula of the form

$$\int_{x_0-h}^{x_0+h} f(x)dx \approx a_0 f(x_0)$$

which is exact for polynomials up to degree 0. The one equation for the one unknown, a_0, is

$$\int_{x_0-h}^{x_0+h} (1)dx = a_0(1)$$

or $2h = a_0$. Hence, we have

$$\int_{x_0-h}^{x_0+h} f(x)dx \approx 2hf(x_0),$$

which we claim has degree of precision 1, not 0. Indeed, for $f(x) = x - x_0$,

$$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{1}{2}(x-x_0)^2\bigg|_{x_0-h}^{x_0+h} = 0$$

and

$$2hf(x_0) = 2h(x_0 - x_0) = 0,$$

so it is exact for degree one polynomials. However, for $f(x) = (x - x_0)^2$,

$$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{1}{3}(x-x_0)^3\bigg|_{x_0-h}^{x_0+h} = \frac{2}{3}h^3$$

and

$$2hf(x_0) = 2h(x_0 - x_0)^2 = 0,$$

so it is not exact for all degree two polynomials. Therefore, its degree of precision is 1. Note the formula $\displaystyle\int_{x_0-h}^{x_0+h} f(x)dx \approx 2hf(x_0)$ is equivalent to the Midpoint Rule as found in Table 4.5.

Now

$$\begin{aligned} G_2(x) &= \frac{3xG_1(x) - G_0(x)}{2} \\ &= \frac{1}{2}(3x^2 - 1) \end{aligned}$$

so

$$\begin{aligned} G_3(x) &= \frac{5xG_2(x) - 2G_1(x)}{3} \\ &= \frac{\frac{5}{2}(3x^3 - x) - 2x}{3} \\ &= \frac{5(3x^3 - x) - 4x}{6} \\ &= \frac{15x^3 - 9x}{6} \\ &= \frac{1}{2}(5x^3 - 3x), \end{aligned}$$

which has roots $-\sqrt{\frac{3}{5}}, 0, \sqrt{\frac{3}{5}}$. Hence, we seek a formula of the form

$$\int_{x_0-h}^{x_0+h} f(x)dx \approx a_0 f\left(x_0 - \sqrt{\frac{3}{5}}h\right) + a_1 f(x_0) + a_2 f\left(x_0 + \sqrt{\frac{3}{5}}h\right)$$

which is exact for polynomials up to degree 2. The three equations for the three unknowns are

$$\begin{aligned}\int_{x_0-h}^{x_0+h}(1)dx = 2h &= a_0 + a_1 + a_2 \\ \int_{x_0-h}^{x_0+h}(x-x_0)dx = 0 &= -\sqrt{\frac{3}{5}}ha_0 + \sqrt{\frac{3}{5}}ha_2 \\ \int_{x_0-h}^{x_0+h}(x-x_0)^2dx = \frac{2}{3}h^3 &= \frac{3}{5}h^2a_0 + \frac{3}{5}h^2a_2.\end{aligned}$$

The solution is

$$a_0 = a_2 = \frac{5}{9}h \quad \text{and} \quad a_1 = \frac{8}{9}h,$$

so the quadrature formula is

$$\int_{x_0-h}^{x_0+h} f(x)dx \approx \frac{h}{9}\left[5f\left(x_0 - \sqrt{\frac{3}{5}}h\right) + 8f(x_0) + 5f\left(x_0 + \sqrt{\frac{3}{5}}h\right)\right].$$

The formula was derived to be exact for polynomials up to degree 2, so its degree of precision is at least 2. We claim the degree of precision is actually 5. For $f(x) = (x-x_0)^3$,

$$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{1}{4}(x-x_0)^4\Big|_{x_0-h}^{x_0+h} = 0$$

and

$$\frac{h}{9}\left[5f\left(x_0 - \sqrt{\frac{3}{5}}h\right) + 8f(x_0) + 5f\left(x_0 + \sqrt{\frac{3}{5}}h\right)\right] = \frac{h}{9}\left[5\left(-\sqrt{\frac{3}{5}}h\right)^3 + 0 + 5\left(\sqrt{\frac{3}{5}}h\right)^3\right] = 0,$$

so it is exact for degree three polynomials. For $f(x) = (x-x_0)^4$,

$$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{1}{5}(x-x_0)^5\Big|_{x_0-h}^{x_0+h} = \frac{2}{5}h^5$$

and

$$\begin{aligned}\frac{h}{9}\left[5f\left(x_0 - \sqrt{\frac{3}{5}}h\right) + 8f(x_0) + 5f\left(x_0 + \sqrt{\frac{3}{5}}h\right)\right] &= \frac{h}{9}\left[5\left(-\sqrt{\frac{3}{5}}h\right)^4 + 0 + 5\left(\sqrt{\frac{3}{5}}h\right)^4\right] \\ &= \frac{5}{9}h\left[\frac{9}{25}h^4 + \frac{9}{25}h^4\right] \\ &= \frac{2}{5}h^5,\end{aligned}$$

so it is exact for degree four polynomials. For $f(x) = (x-x_0)^5$,

$$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{1}{6}(x-x_0)^6\Big|_{x_0-h}^{x_0+h} = 0$$

and

$$\frac{h}{9}\left[5f\left(x_0 - \sqrt{\frac{3}{5}}h\right) + 8f(x_0) + 5f\left(x_0 + \sqrt{\frac{3}{5}}h\right)\right] = \frac{h}{9}\left[5\left(-\sqrt{\frac{3}{5}}h\right)^5 + 0 + 5\left(\sqrt{\frac{3}{5}}h\right)^5\right] = 0,$$

so it is exact for degree five polynomials. However, for $f(x) = (x-x_0)^6$,

$$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{1}{7}(x-x_0)^7\Big|_{x_0-h}^{x_0+h} = \frac{2}{7}h^7$$

and

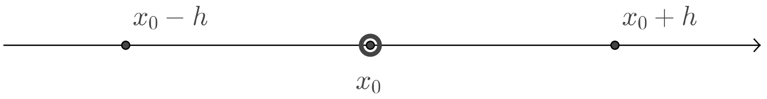
$$\begin{aligned}
\frac{h}{9}\left[5f\left(x_0-\sqrt{\frac{3}{5}}h\right)+8f(x_0)+5f\left(x_0+\sqrt{\frac{3}{5}}h\right)\right] &= \frac{h}{9}\left[5\left(-\sqrt{\frac{3}{5}}h\right)^6+0+5\left(\sqrt{\frac{3}{5}}h\right)^6\right] \\
&= \frac{5}{9}h\left[\frac{27}{125}h^6+\frac{27}{125}h^6\right] \\
&= \frac{3}{25}h^7,
\end{aligned}$$

so it is not exact for all degree six polynomials. Its degree of precision is 5. The formula is listed as the second Gaussian quadrature formula in table 4.5.

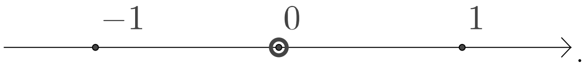
We can also find the degree of precision of any numerical calculus formula by observing the form of its error term. If the error term has the form $O(h^n f^{(k)}(\xi_h))$, then its degree of precision is $k-1$.

Some standard formulas

Tables 4.2 , 4.3 , 4.4 , and 4.5 summarize some standard formulas for derivatives and integrals. Notice there are no one-point formulas for any derivatives, no two-point formulas for second derivatives or higher, and no three-point formulas for third derivatives or higher. The stencils have been streamlined to show only the values of θ_i. Hence, the stencil



appears in the table as



Key Concepts

Degree of precision: The integer m such that a numerical calculus formula is exact for all polynomials of degree up to and including m but is not exact for all polynomials of degree $m+1$.

Error terms: Error terms for numerical calculus approximations can be found by replacing all occurrences of f in an approximation formula by Taylor series expansions about x_0 and reducing.

Gaussian quadrature: A quadrature method which maximizes the degree of precision relative to the number of nodes used.

Quadrature: Another name for a numerical integration formula.

Weighted Mean Value Theorem: Assume that f and g are continuous on $[a,b]$. If g never changes sign and is non-negative in $[a,b]$, then we have that,

$$\int_a^b f(x)g(x)dx = f(c)\int_a^b g(x)dx$$

for some c in (a,b).

Table 4.2: Some standard first derivative formulas.

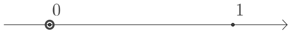
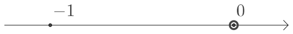
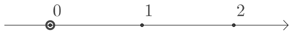
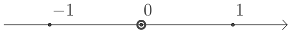
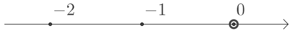
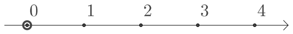
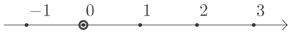
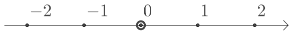
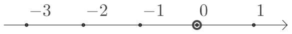
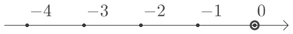
Stencil	Formula	Name
	2-point formulas	
	$f'(x_0) = \dfrac{-f(x_0) + f(x_0+h)}{h} - \dfrac{h}{2} f''(\xi_h)$	Forward Difference
	$f'(x_0) = \dfrac{-f(x_0-h) + f(x_0)}{h} + \dfrac{h}{2} f''(\xi_h)$	Backward Difference
	3-point formulas	
	$f'(x_0) = \dfrac{-3f(x_0) + 4f(x_0+h) - f(x_0+2h)}{2h} + \dfrac{h^2}{3} f'''(\xi_h)$	Forward Difference
	$f'(x_0) = \dfrac{-f(x_0-h) + f(x_0+h)}{2h} + \dfrac{h^2}{6} f'''(\xi_h)$	Centered Difference
	$f'(x_0) = \dfrac{f(x_0-2h) - 4f(x_0-h) + 3f(x_0)}{2h} + \dfrac{h^2}{3} f'''(\xi_h)$	Backward Difference
	5-point formulas	
	$f'(x_0) = \dfrac{-25f(x_0) + 48f(x_0+h) - 36f(x_0+2h) + 16f(x_0+3h) - 3f(x_0+4h)}{12h} + \dfrac{h^4}{5} f^{(5)}(\xi_h)$	Forward Difference
	$f'(x_0) = \dfrac{-3f(x_0-h) - 10f(x_0) + 18f(x_0+h) - 6f(x_0+2h) + f(x_0+3h)}{12h} + \dfrac{h^4}{20} f^{(5)}(\xi_h)$	
	$f'(x_0) = \dfrac{f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)}{12h} + \dfrac{h^4}{30} f^{(5)}(\xi_h)$	Centered Difference
	$f'(x_0) = \dfrac{-f(x_0-3h) + 6f(x_0-2h) - 18f(x_0-h) + 10f(x_0) + 3f(x_0+h)}{12h} + \dfrac{h^4}{20} f^{(5)}(\xi_h)$	
	$f'(x_0) = \dfrac{3f(x_0-4h) - 16f(x_0-3h) + 36f(x_0-2h) - 48f(x_0-h) + 25f(x_0)}{12h} + \dfrac{h^4}{5} f^{(5)}(\xi_h)$	Backward Difference

Table 4.3: Some second derivative formulas.

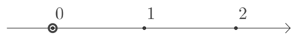
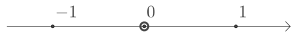
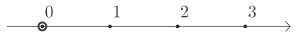
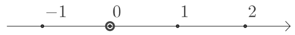
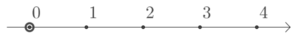
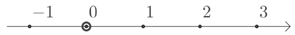
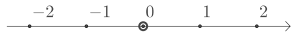
Stencil	Formula	Name
	3-point formulas	
	$f''(x_0) = \dfrac{f(x_0) - 2f(x_0+h) + f(x_0+2h)}{h^2} + O(hf^{(3)}(\xi_h))$	Forward Difference
	$f''(x_0) = \dfrac{f(x_0-h) - 2f(x_0) + f(x_0+h)}{h^2} + O(h^2 f^{(4)}(\xi_h))$	Centered Difference
	4-point formulas	
	$f''(x_0) = \dfrac{2f(x_0) - 5f(x_0+h) + 4f(x_0+2h) - f(x_0+3h)}{h^2} + O(h^2 f^{(4)}(\xi_h))$	Forward Difference
	$f''(x_0) = \dfrac{f(x_0-h) - 2f(x_0) + f(x_0+h)}{h^2} + O(h^2 f^{(4)}(\xi_h))$	
	5-point formulas	
	$f''(x_0) = \dfrac{35f(x_0) - 104f(x_0+h) + 114f(x_0+2h) - 56f(x_0+3h) + 11f(x_0+4h)}{12h^2} + O(h^3 f^{(5)}(\xi_h))$	Forward Difference
	$f''(x_0) = \dfrac{11f(x_0-h) - 20f(x_0) + 6f(x_0+h) + 4f(x_0+2h) - f(x_0+3h)}{12h^2} + O(h^3 f^{(5)}(\xi_h))$	
	$f''(x_0) = \dfrac{-f(x_0-2h) + 16f(x_0-h) - 30f(x_0) + 16f(x_0+h) - f(x_0+2h)}{12h^2} + O(h^4 f^{(6)}(\xi_h))$	Centered Difference

Table 4.4: Some third derivative formulas.

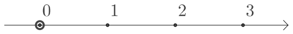
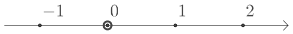
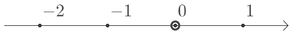
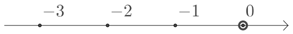
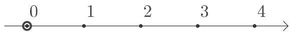
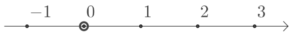
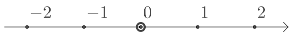
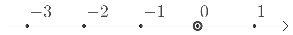
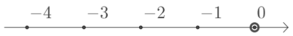
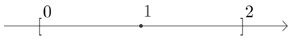
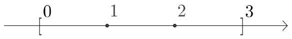
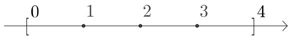
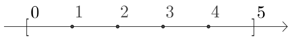
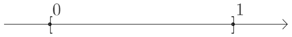
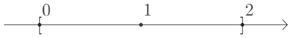
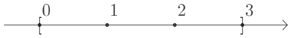
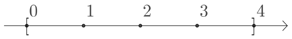
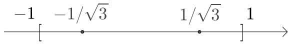
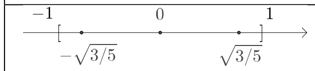
Stencil	Formula	Name
	4-point formulas	
	$f'''(x_0) = \frac{-f(x_0)+3f(x_0+h)-3f(x_0+2h)+f(x_0+3h)}{h^3} + O(h)f^{(4)}(\xi_h)$	Forward Difference
	$f'''(x_0) = \frac{-f(x_0-h)+3f(x_0)-3f(x_0+h)+f(x_0+2h)}{h^3} + O(hf^{(4)}(\xi_h))$	
	$f'''(x_0) = \frac{-f(x_0-2h)+3f(x_0-h)-3f(x_0)+f(x_0+h)}{h^3} + O(hf^{(4)}(\xi_h))$	
	$f'''(x_0) = \frac{-f(x_0)+3f(x_0+h)-3f(x_0+2h)+f(x_0+3h)}{h^3} + O(hf^{(4)}(\xi_h))$	Backward Difference
	5-point formulas	
	$f'''(x_0) = \frac{-5f(x_0)+18f(x_0+h)-24f(x_0+2h)+14f(x_0+3h)-3f(x_0+4h)}{2h^3} + O(h^2f^{(5)}(\xi_h))$	Forward Difference
	$f'''(x_0) = \frac{-3f(x_0-h)+10f(x_0)-12f(x_0+h)+6f(x_0+2h)-f(x_0+3h)}{2h^3} + O(h^2f^{(5)}(\xi_h))$	
	$f'''(x_0) = \frac{-f(x_0-2h)+2f(x_0-h)-2f(x_0+h)+f(x_0+2h)}{2h^3} + O(h^2f^{(5)}(\xi_h))$	Centered Difference
	$f'''(x_0) = \frac{f(x_0-3h)-6f(x_0-2h)+12f(x_0-h)-10f(x_0)+3f(x_0+h)}{2h^3} + O(h^2f^{(5)}(\xi_h))$	
	$f'''(x_0) = \frac{3f(x_0-4h)-14f(x_0-3h)+24f(x_0-2h)-18f(x_0-h)+5f(x_0)}{2h^3} + O(h^2f^{(5)}(\xi_h))$	Backward Difference

Table 4.5: Some integration formulas.

Stencil	Formula	Name
	open Newton-Cotes formulas	
	$\int_{x_0}^{x_0+2h} f(x)dx = 2hf(x_0+h) + O(h^3 f''(\xi_h))$	Midpoint Rule
	$\int_{x_0}^{x_0+3h} f(x)dx = \frac{3h}{2}\left[f(x_0+h) + f(x_0+2h)\right] + O(h^3 f''(\xi_h))$	
	$\int_{x_0}^{x_0+4h} f(x)dx = \frac{4h}{3}\left[2f(x_0+h) - f(x_0+2h) + 2f(x_0+3h)\right] + O(h^5 f^{(4)}(\xi_h))$	
	$\int_{x_0}^{x_0+5h} f(x)dx = \frac{5h}{24}\left[11f(x_0+h) + f(x_0+2h) + f(x_0+3h) + 11f(x_0+4h)\right] + O(h^5 f^{(4)}(\xi_h))$	
	closed Newton-Cotes formulas	
	$\int_{x_0}^{x_0+h} f(x)dx = \frac{h}{2}\left[f(x_0) + f(x_0+h)\right] + O(h^3 f''(\xi_h))$	Trapezoidal Rule
	$\int_{x_0}^{x_0+2h} f(x)dx = \frac{h}{3}\left[f(x_0) + 4f(x_0+h) + f(x_0+2h)\right] + O(h^5 f^{(4)}(\xi_h))$	Simpson's Rule
	$\int_{x_0}^{x_0+3h} f(x)dx = \frac{3h}{8}\left[f(x_0) + 3f(x_0+h) + 3f(x_0+2h) + f(x_0+3h)\right] + O(h^5 f^{(4)}(\xi_h))$	Simpson's $\frac{3}{8}$ Rule
	$\int_{x_0}^{x_0+4h} f(x)dx = \frac{2h}{45}\left[7f(x_0) + 32f(x_0+h) + 12f(x_0+2h) + 32f(x_0+3h) + 7f(x_0+4h)\right] + O(h^7 f^{(6)}(\xi_h))$	Bode's Rule
	Gaussian quadrature formulas	
	$\int_{x_0-h}^{x_0+h} f(x)dx = h\left[f\left(x_0 - \frac{1}{\sqrt{3}}h\right) + f\left(x_0 + \frac{1}{\sqrt{3}}h\right)\right] + O(h^5 f^{(4)}(\xi_h))$	
	$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{h}{9}\left[5f\left(x_0 - \sqrt{\frac{3}{5}}h\right) + 8f(x_0) + 5f\left(x_0 + \sqrt{\frac{3}{5}}h\right)\right] + O(h^7 f^{(6)}(\xi_h))$	

Exercises

1. Let $f(x) = e^x - \sin x$. Complete the following table using the approximation formula

$$f'(x_0) \approx \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h}.$$

h	approximate $f'(2)$	abs. error
.01		
.005		
−.005		
−.01		

Is it OK to use negative values for h?

2. For each value of x in the table, use the most accurate three-point formula to approximate $f'(x)$. [𝔸]

x	$f(x)$	$f'(x)$
−2.7	0.054797	
−2.5	0.11342	
−2.3	0.65536	
−2.1	0.98472	

3. Approximate the integral using Simpson's rule.

   (a) $\displaystyle\int_{-0.5}^{0} x \ln(x+1) dx$ [𝕊]

   (b) $\int_1^3 \ln(x+1)\, dx$

   (c) $\displaystyle\int_{-0.25}^{0.25} (\cos x)^2 dx$ [𝔸]

   (d) $\int_1^3 e^{\sin x}\, dx$

   (e) $\int_1^2 x^4\, dx$ [𝔸]

4. Do question 3 using the Trapezoidal rule. [𝕊] [𝔸]
5. Do question 3 using the Midpoint rule. [𝕊] [𝔸]
6. Find the error of the approximation in question 3. [𝕊] [𝔸]
7. Find the error of the approximation in question 4. [𝕊] [𝔸]
8. Find the error of the approximation in question 5. [𝕊] [𝔸]
9. Find the error in approximating $\int_{-7}^{11}(32x^2 + \sqrt{7}x - 2)dx$ using Simpson's $\frac{3}{8}$ Rule.
10. Find the error in approximating $\int_{-17}^{36}(32x^5 + 7x^3 - 2)dx$ using Bode's Rule. [𝔸]
11. For the following values of f, x_0, and h, use the formula

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{h^2}{6} f'''(\xi)$$

to approximate $f'(x_0)$.

   (a) $f(x) = e^x$; $x_0 = 2$; $h = 0.1$. [𝕊]

   (b) $f(x) = (\cosh 2x)^2 - \sin x$; $x_0 = \pi$; $h = 0.05$. [𝔸]

   (c) $f(x) = \ln(2x - 3) + 5x$; $x = 10$; $h = 1$.

12. Compute both a lower bound and an upper bound on the error for the approximation in question 11. Verify that the actual error is between these bounds. [𝕊] [𝔸]
13. For each part of question 11, find the value of ξ guaranteed by the formula. [𝕊] [𝔸]
14. State the degree of precision of the closed Newton-Cotes formula on 5 nodes, Bode's Rule.
15. State the degree of precision of the five point formula. [𝕊]

$$\begin{aligned} f'(x_0) = \frac{1}{12h}&[-25f(x_0) + 48f(x_0 + h) - 36f(x_0 + 2h) \\ &+16f(x_0 + 3h) - 3f(x_0 + 4h)] + \frac{h^4}{5} f^{(5)}(\xi) \end{aligned}$$

16. Find the degree of precision of the quadrature formula

$$\int_3^5 f(x)\,dx \approx \frac{1}{2}\left[3f\left(\frac{11}{3}\right) + f(5)\right].$$

17. Find the error term for the quadrature method, and state its degree of precision.

    (a) $\displaystyle\int_{x_0}^{x_0+h} f(x)\,dx \approx hf(x_0)$ [𝔸]

    (b) $\displaystyle\int_{x_0}^{x_0+h} f(x)\,dx \approx hf\left(x_0 + \frac{h}{4}\right)$

    (c) $\displaystyle\int_{x_0}^{x_0+h} f(x)dx \approx \frac{h}{4}\left[3f\left(x_0 + \frac{2}{3}h\right) + f(x_0)\right]$ [𝕊]

    (d) $\displaystyle\int_{x_0}^{x_0+2h} f(x)dx \approx \frac{h}{2}\left[3f\left(x_0 + \frac{4}{3}h\right) + f(x_0)\right]$

    (e) $\displaystyle\int_{x_0}^{x_0+3h} f(x)dx \approx \frac{3h}{4}\left[f(x_0) + 3f(x_0+2h)\right]$ [𝔸]

    (f) $\displaystyle\int_{x_0}^{x_0+2h} f(x)dx \approx \frac{h}{2}\left[f\left(x_0 - \frac{h}{2}\right) + 3f\left(x_0 + \frac{3}{2}h\right)\right]$

    (g) $\displaystyle\int_{x_0}^{x_0+2h} f(x)dx \approx \frac{h}{3}\left[f(x_0-h) - 2f(x_0) + 7f(x_0+h)\right]$ [𝔸]

    (h) $\displaystyle\int_{x_0}^{x_0+3h} f(x)dx \approx 3h\left[3f\left(x_0 + \frac{3}{2}h\right) - 6f(x_0+h) + 4f\left(x_0 + \frac{3}{4}h\right)\right]$

    (i) $\displaystyle\int_{x_0}^{x_0+3h} f(x)dx \approx -\frac{h}{12}\left[208f\left(x_0 + \frac{3}{2}h\right) - 891f(x_0+h) + 1344f\left(x_0 + \frac{3}{4}h\right) - 625f\left(x_0 + \frac{3}{5}h\right)\right]$ [𝔸]

18. Find the error term for the derivative approximation:

    (a) $f'(x_0) \approx \dfrac{f(x_0+2h) - f(x_0)}{2h}$ [𝔸]

    (b) $f'(x_0) \approx \dfrac{f(x_0+2h) - f(x_0-h)}{3h}$

    (c) $f'(x_0) \approx \dfrac{-3f(x_0) + 4f(x_0 + \frac{h}{2}) - f(x_0+h)}{h}$ [𝕊]

    (d) $f'(x_0) \approx \dfrac{-13f(x_0-10h) - 12f(x_0+5h) + 25f(x_0+8h)}{270h}$

    (e) $f'(x_0) \approx \dfrac{-7f(x_0+h) + 416f(x_0+\frac{1}{2}h) - 2916f(x_0+\frac{1}{3}h) + 5632f(x_0+\frac{1}{4}h) - 3125f(x_0+\frac{1}{5}h)}{12h}$ [𝔸]

    (f) $f''(x_0) \approx \dfrac{2f(x_0-h) - 3f(x_0) + f(x_0+2h)}{3h^2}$

    (g) $f''(x_0) \approx \dfrac{7f(x_0-5h) - 12f(x_0) + 5f(x_0+7h)}{210h^2}$ [𝔸]

    (h) $f''(x_0) \approx \dfrac{5f(x_0-5h) - 12f(x_0+2h) + 7f(x_0+7h)}{210h^2}$

    (i) $f''(x_0) \approx \dfrac{5f(x_0-2h) + 32f(x_0-h) - 60f(x_0) + 25f(x_0+2h) - 2f(x_0+4h)}{60h^2}$ [𝔸]

19. Diffy Rence writes down the following approximation:

$$f''(3.0) \approx 25[\sin(2.8) - 2\sin(3.0) + \sin(3.2)].$$

    What is $f(x)$? [𝕊]

20. Let $f(x) = \sin x$.

    (a) Find a bound on the error of the approximation

$$f'(6) \approx \frac{-3\sin 6 + 4\sin 6.1 - \sin 6.2}{0.2}$$

    according to the appropriate error term.

(b) Compare this bound to the actual error.

21. What can you say about the error in approximating the first derivative of

$$f(x) = -13x^4 + 17x^3 - 15x^2 + 12x - 99$$

using a 5-point formula?

22. Let $f(x) = 3x^3 - 2x^2 + x$.

(a) Compute the error (not a bound on the error) in estimating $f'(2)$ using the forward difference

$$\frac{f(x_0 + h) - f(x_0)}{h}$$

with $h = 0.1$.

(b) Find $\xi_{0.1}$ as guaranteed by the error term.

23. Let $f(x) = \sin x$. Find a bound on the error of the approximation.

(a) $f''(3.0) \approx 25[\sin(2.8) - 2\sin(3.0) + \sin(3.2)]$ [𝔸]

(b) $f''(3.0) \approx 1600\,[2\sin(3.0) - 5\sin(3.025) + 4\sin(3.05) - \sin(3.075)]$

(c) $f'''(3.0) \approx 500000\,[-5\sin(3.0) + 18\sin(3.01) - 24\sin(3.02) + 14\sin(3.03) - 3\sin(3.04)]$ [𝕊]

(d) $f'''(3.0) \approx 1000\,[-\sin(2.8) + 3\sin(2.9) - 3\sin(3.0) + \sin(3.1)]$

(e) $\displaystyle\int_3^4 f(x)dx \approx \frac{1}{6}\,[\sin(3) + 4\sin(3.5) + \sin(4)]$

(f) $\displaystyle\int_3^4 f(x)dx \approx \frac{1}{2}\left[\sin\left(\frac{7}{2} - \frac{1}{2\sqrt{3}}\right) + \sin\left(\frac{7}{2} + \frac{1}{2\sqrt{3}}\right)\right]$ [𝕊]

24. Suppose you have the following data on a function f. [𝕊]

x	0	1	2	3	4
$f(x)$	-0.2381	-0.3125	-0.4545	-0.8333	-5

(a) Approximate $f'(4)$ and $f'(2)$ using 5-point formulas.

(b) Which approximation would you expect to be more accurate, and why?

(c) Did it turn out that way? The data came from $f(x) = \frac{1}{x-4.2}$.

25. Refer to the quadrature method

$$\int_{x_0}^{x_0+h} f(x)\,dx = \frac{h}{2}\left[f\left(x_0 + \frac{h}{3}\right) + f\left(x_0 + \frac{2h}{3}\right)\right] + \frac{h^3}{36}f''(\xi)$$

in all of the following questions. [𝔸]

(a) What is the rate of convergence?

(b) What is the degree of precision?

(c) Use the method to approximate $\int_0^\pi \sin x\,dx$.

(d) Find a bound on the error of this approximation.

(e) Compare the bound to the actual error.

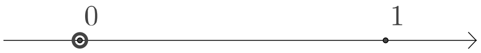
26. The Trapezoidal rule applied to $\displaystyle\int_0^2 f(x)dx$ gives the value 5, and the Midpoint rule gives the value 4. What value does Simpson's rule give?

27. The Trapezoidal Rule applied to $\int_0^2 f(x)\,dx$ gives the value 4, and Simpson's Rule gives the value 2. What is $f(1)$? [𝔸]

28. When approximating $f'''(x_0)$ using five nodes, the rate of convergence will be at least what? [𝔸]

29. Show that the average of the forward difference, $\frac{-f(x_0)+f(x_0+h)}{h}$, and backward difference, $\frac{-f(x_0-h)+f(x_0)}{h}$, approximations of $f'(x_0)$ gives the central difference approximation, $\frac{f(x_0+h)-f(x_0-h)}{2h}$, of $f'(x_0)$.

30. Chuck was "approximating" a definite integral using Simpson's Rule. As you can see from his work below, he was integrating a cubic polynomial. Calculate the error he incurred even though you can not read all the coefficients. [𝔸]



31. Repeat 30 supposing Chuck was using the Trapezoidal Rule. [A]

32. Sketch the graph of a function $f(x)$, and indicate on it values for x_0 and h so that the backward difference $\frac{f(x_0)-f(x_0-h)}{h}$ gives a ***better*** approximation of $f'(x_0)$ than does the central difference $\frac{f(x_0+h)-f(x_0-h)}{2h}$.

33. Sketch the graph of a function $f(x)$ for which the Trapezoidal Rule gives a better approximation of $\int_0^1 f(x)\,dx$ than does Simpson's Rule, and explain how you know. [S]

34. Suppose a 5 point formula is used to approximate $f''(x_0)$ for stepsizes $h = 0.1$ and $h = 0.02$. If $E_{0.1}$ represents the error in the approximation for $h = 0.1$ and $E_{0.02}$ represents the error in the approximation for $h = 0.02$, what would you expect $\frac{E_{0.1}}{E_{0.02}}$ to be, approximately? [S]

35. A general three point formula using nodes x_0, $x_0 + \alpha h$, and $x_0 + 2h$, $(\alpha \neq 0, 2)$ is given by

$$f'(x_0) \approx \frac{1}{2h}\left[-\frac{2+\alpha}{\alpha}f(x_0) + \frac{4}{\alpha(2-\alpha)}f(x_0+\alpha h) - \frac{\alpha}{2-\alpha}f(x_0+2h)\right].$$

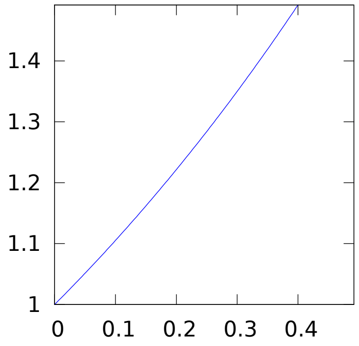
   (a) Show that this formula reduces to one of the standard formulas when $\alpha = 1$.

   (b) Find the error term for this formula.

36. Find three different approximations for $f'(0.2)$ using three-point formulas. [A]

x	$f(x)$
0	1
0.1	1.10517
0.2	1.22140
0.3	1.34986
0.4	1.49182

The graph of $f'''(x)$ is shown below. Use it to rank your three approximations in order from least expected error to greatest expected error, and explain why you ranked them the way you did.



37. Verify numerically that the error in using the formula $f'(x_0) = \frac{-2f(x_0-h)-3f(x_0)+6f(x_0+h)-f(x_0+2h)}{6h}$ to approximate $f'(3)$ using the function $f(x) = (\cos 3x)^2 + \ln x$ is really $O(h^3)$.

38. Numerically approximate the best estimate that can be obtained from the formula

$$f'(3) = \frac{-2f(3-h) - 3f(3) + 6f(3+h) - f(3+2h)}{6h}$$

with double precision computation and $f(x) = (\cos 3x)^2 + \ln x$. What value of h gives this optimal approximation? [A]

39. Find the degree of precision of the quadrature formula

$$\int_{-1}^{1} f(x)dx = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right).$$

40. The quadrature formula $\displaystyle\int_0^2 f(x)dx = c_0f(0) + c_1f(1) + c_2f(2)$ is exact for all polynomials of degree less than or equal to 2. Determine c_0, c_1, and c_2.

4.4 Composite Integration

In section 4.3 we supplied error terms that took the form $O(h^k f^{(l)}(\xi_h))$. As a prime example, the trapezoidal rule, $\displaystyle\int_{x_0}^{x_0+h} f(x)dx = \frac{h}{2}\left[f(x_0)+f(x_0+h)\right] + O(h^3 f''(\xi_h))$, has error term $O(h^3 f''(\xi_h))$. This conclusion follows directly from a Taylor series analysis, but what does it mean?

Error terms for derivative approximations are comparatively easy to understand. Consider the first derivative approximation $f'(x_0) = \dfrac{-f(x_0-h)+f(x_0+h)}{2h} + \dfrac{h^2}{6}f'''(\xi_h)$. The smaller h is, the smaller the error in approximating $f'(x_0)$ is (as long as the $f'''(\xi_h)$ term doesn't counteract the benefit of shrinking h). Error terms for integral approximations are not as straightforward because, in each case, the quantity being approximated depends on h. Changing h in the integration formula also changes the quantity being approximated. This is true of each formula in table 4.5. The trapezoidal rule is as good an example as any. The left hand side, the quantity being approximated, is $\int_{x_0}^{x_0+h} f(x)dx$, so smaller h means approximating the integral over a smaller interval. So how does having a smaller error in approximating a different number tell us anything about the potential benefit of computing with smaller values of h? Careful study of the trapezoidal rule will reveal the answer.

According to the trapezoidal rule, $\frac{h}{2}\left[f(x_0)+f(x_0+h)\right]$ approximates the integral of f over the interval $[x_0, x_0+h]$. If h is replaced by $h/2$, the resulting approximation, $\frac{h}{4}\left[f(x_0)+f(x_0+\frac{h}{2})\right]$, is an approximation of the integral of f over the interval $[x_0, x_0+\frac{h}{2}]$. It is no longer an approximation of the integral over $[x_0, x_0+h]$! To use the trapezoidal rule to approximate the original quantity, the integral of f over $[x_0, x_0+h]$, using $h/2$ instead of h requires two applications of the trapezoidal rule—one over the interval $[x_0, x_0+\frac{h}{2}]$ and one over the interval $[x_0+\frac{h}{2}, x_0+h]$. The sum of these two approximations is an approximation for the integral of f over $[x_0, x_0+h]$. Reducing h further requires more applications of the trapezoidal rule over more intervals. In general, reducing h to $\frac{h}{n}$ for any whole number n requires n applications of the trapezoidal rule:

$$\begin{aligned}
\int_{x_0}^{x_0+h} f(x)dx &= \int_{x_0}^{x_0+\frac{h}{n}} f(x)dx + \int_{x_0+\frac{h}{n}}^{x_0+2\frac{h}{n}} f(x)dx + \cdots + \int_{x_0+(n-1)\frac{h}{n}}^{x_0+h} f(x)dx \\
&\approx \frac{h}{2n}\left[f(x_0) + f\left(x_0+\frac{h}{n}\right)\right] + \frac{h}{2n}\left[f\left(x_0+\frac{h}{n}\right) + f\left(x_0+2\frac{h}{n}\right)\right] + \\
&\quad \cdots + \frac{h}{2n}\left[f\left(x_0+(n-1)\frac{h}{n}\right) + f(x_0+h)\right]. \qquad (4.4.1)
\end{aligned}$$

Decomposing $\int_{x_0}^{x_0+h} f(x)dx$ into the sum $\int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx$ and summing approximations of these integrals is called composite integration.

As for using the trapezoidal rule to do the approximating, the error in a single application of the trapezoidal rule is $O(h^3 f''(\xi_h))$. The error in the above sum is, therefore, bounded by $\sum_{i=1}^n M\left(\frac{h}{n}\right)^3 f''(\mu_i) = Mh\left(\frac{h}{n}\right)^2 \cdot \frac{1}{n}\sum_{i=1}^n f''(\mu_i)$ for some μ_i with $x_0+(i-1)\frac{h}{n} < \mu_i < x_0+i\frac{h}{n}$. Assuming f'' is continuous on $[x_0, x_0+h]$, the intermediate value theorem allows us to replace $\frac{1}{n}\sum_{i=1}^n f''(\mu_i)$ with $f''(\xi_n)$ for some $\xi_n \in (x_0, x_0+h)$ because $\frac{1}{n}\sum_{i=1}^n f''(\mu_i)$ is the average of the $f''(\mu_i)$, which is no more than the maximum of the $f''(\mu_i)$ and no less than the minimum of the $f''(\mu_i)$. Making this replacement gives us the error bound $Mh\left(\frac{h}{n}\right)^2 f''(\xi_n)$. In conclusion, the trapezoidal rule used multiple times when necessary to approximate $\int_{x_0}^{x_0+h} f(x)dx$ actually has error $O\left(\left(\frac{1}{n}\right)^2 f''(\xi_n)\right)$, where n is the number of subintervals used in the calculation and ξ_n depends on n. Now the nature of the error is clearer. It is measured by how many subintervals are used in the calculation. More subintervals (greater n) means less error (assuming the benefit of more subintervals is not counteracted by the f'' factor). Other composite integration formulas are similar. If a single-interval quadrature formula has error $O(h^k f^{(l)}(\xi_h))$, then the corresponding composite version has error $O\left(\left(\frac{1}{n}\right)^{k-1} f^{(l)}(\xi_n)\right)$. More intervals generally means smaller error.

Composite Trapezoidal Rule

Equation 4.4.1 encapsulates the composite trapezoidal rule but does not represent the most efficient way to use it. Simplifying the expression will help. Notice that all of the function evaluations except $f(x_0)$ and $f(x_0+h)$ occur

Table 4.6: Minimum number of intervals to achieve certain accuracies using the composite trapezoidal rule to approximate $\int_0^3 e^{-x^2}dx$.

accuracy	$2.2(10)^{-2}$	$5(10)^{-5}$	10^{-5}	10^{-7}	10^{-11}	10^{-15}
subintervals	2	3	8	75	7453	> 745300

twice, so we can condense the formula to

$$\begin{aligned}\int_{x_0}^{x_0+h} f(x)dx &\approx \frac{h}{2n}\left[f(x_0)+f(x_0+h)\right]+\frac{h}{n}\left[f\left(x_0+\frac{h}{n}\right)+\cdots+f\left(x_0+(n-1)\frac{h}{n}\right)\right]\\ &= \frac{h}{2n}\left[f(x_0)+f(x_0+h)+2\sum_{i=1}^{n-1} f\left(x_0+i\frac{h}{n}\right)\right].\end{aligned}$$

This leads to the following pseudo-code where we make the substitutions $a = x_0$ and $b = x_0 + h$.

```
Assumptions: $f$ has a continuous second derivative on $[a,b]$.
Input: Function $f$; interval over which to integrate $[a,b]$; number of subintervals $n$.
Step 1: Set $s = \frac{b-a}{n}$; $I = \frac{f(a)+f(b)}{2}$;
Step 2: For $i = 1, 2, \ldots, n-1$ do Step 3:
    Step 3: Set $I = I + f(a+is)$;
Step 4: Set $I = sI$;
Output: Approximate value of $\int_a^b f(x)dx$.
```

Other composite integration formulas should be simplified likewise to minimize the number of times f is evaluated.

Adaptive quadrature

$$\int_0^3 e^{-x^2}dx \approx 4.57837939409486$$

and it is simple enough to approximate this value with the composite trapezoidal rule. Table 4.6 shows the minimum number of subintervals needed to achieve various accuracies, assuming the calculations are done with enough significant digits that floating point error does not overwhelm the calculation. It should be apparent that achieving high accuracy results using the trapezoidal rule is not practical. It requires too many computations. We will take up this deficiency in the next section. For now, let's analyze the usefulness of the error bound $O\left(\left(\frac{1}{n}\right)^2 f''(\xi_n)\right)$. Assuming $f''(\xi_n)$ is roughly

Crumpet 26: error function

The error function is defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

and is critical in the study of statistics as it is used to calculate probabilities associated with the normal distribution. The factor $\frac{2}{\sqrt{\pi}}$ comes from the fact that $\int_{-\infty}^{\infty} e^{-t^2}dt = \frac{\sqrt{\pi}}{2}$, an interesting fact itself.

Computer algebra systems will have the error function built-in just as they do the sine or logarithm functions. Hence, the easiest way to evaluate $\int_0^3 e^{-x^2}dx$ is to have a computer algebra system (or perhaps your calculator) compute $\frac{\sqrt{\pi}}{2}\operatorname{erf}(3)$.

constant, we should expect to improve our estimate from an accuracy of $2.2(10)^{-2}$ to an accuracy of $5(10)^{-5}$, an increase in accuracy of $\frac{2.2(10)^{-2}}{5(10)^{-5}} \approx 440$ times, by increasing the number of subintervals by a factor of about $\sqrt{440} \approx 21$. In other words, we should expect it to take approximately 42 subintervals to achieve $5(10)^{-5}$ accuracy based on accuracy of $2.2(10)^{-2}$ with 2 intervals. Since it only takes 3, we conclude that the assumption that $f''(\xi_2) \approx f''(\xi_3)$ is bad! Luckily, the badness of this assumption actually works in our favor. It takes less, not more, than the expected number of intervals to achieve $5(10)^{-5}$ accuracy. On the other hand, increasing the accuracy from $5(10)^{-5}$ to 10^{-5}, an increase by a factor of 5, we should expect to need about $\sqrt{5} \approx 2.2$ times as many subintervals. $3 \times 2.2 = 6.6$, so the 8 needed is just about what we would expect. Similarly, to increase the accuracy from 10^{-5} to 10^{-7}, an increase in accuracy by a factor of 100, we should expect to need about 10 times as many subintervals. Indeed, 75 is about 10 times as many as 8. Likewise, to increase accuracy by a factor of 10,000 (as in going from 10^{-7} to 10^{-11} or from 10^{-11} to 10^{-15}), we should expect to need to increase the number of subintervals by a factor of 100. Indeed, the table bears this estimate out as well.

Just remember, if f'' does not exist or is wildly discontinuous, or just wildly varying, the assumption that $f''(\xi_n)$ is constant could be a bad one, no matter how many subintervals are used. The more common case is when f'' is continuous and reasonably tame, though. Even in this case, when the number of subintervals is small, the assumption is often not a good one, but when the number of subintervals is large, it is a pretty reliable assumption. The exact number of subintervals needed before this assumption is reasonable changes from one function to another, however.

Taking this lesson to heart, we approximate

$$\int_0^3 \left(x - e^x \cos\sqrt{e^{2x} - x^2}\right) dx$$

using the trapezoidal rule with 50 subintervals and find that it is accurate to within about 10^{-1} of the exact value. How many subintervals should we expect to need to achieve 10^{-3} accuracy? About 10 times as many, or about 500. With 500 subintervals, we actually attain accuracy of about $.997(10)^{-3}$, spot on! The assumption that $f''(\xi_n)$ is constant seems to be valid for this integral with $n \geq 50$ (and maybe for some $n < 50$ too). Alas, this is the type of analysis that can not be done in practice. In practice, we calculate integrals numerically because we don't know how to compute their values exactly! In "real life" situations, we have no way of knowing how accurate an integral estimate is with 3 or 50 or 500 or 3000 subintervals. We need the computer to estimate errors as it calculates, just as we had it do for root-finding algorithms.

Even though we know the assumption is not perfect, especially for small n, we assume $f''(\xi_n)$ is constant, so the error of the trapezoidal rule becomes $O\left(\left(\frac{1}{n}\right)^2\right)$. The f'' factor is subsumed by the implied constant of the big-oh notation. Accordingly, halving the number of intervals can be expected to increase the error by a factor of about 4. Introducing the notation $T_k(a,b)$ for the composite trapezoidal rule approximation of $\int_a^b f(x)dx$ with k subintervals and $e_k = \int_a^b f(x)dx - T_k(a,b)$ for its error,

$$e_n \approx M\left(\frac{1}{n}\right)^2 \quad \text{and} \quad e_{2n} \approx M\left(\frac{1}{2n}\right)^2$$

so

$$\frac{e_n}{e_{2n}} \approx \frac{M\left(\frac{1}{n}\right)^2}{M\left(\frac{1}{2n}\right)^2} = 4, \quad \text{which implies} \quad e_n \approx 4e_{2n}.$$

Because $\int_a^b f(x)dx = T_2(a,b) + e_2 = T_1(a,b) + e_1$,

$$\begin{aligned} T_2(a,b) - T_1(a,b) &= e_1 - e_2 \\ &\approx 4e_2 - e_2 \\ &= 3e_2 \end{aligned}$$

so $e_2 \approx \frac{1}{3}(T_2(a,b) - T_1(a,b))$. Explicitly,

$$\int_a^b f(x)dx - T_2(a,b) \approx \frac{1}{3}(T_2(a,b) - T_1(a,b)).$$

We now have a way of approximating the error numerically, a significant breakthrough! The error is approximately one third the difference between the trapezoidal rule approximations with one subinterval and with two.

To harness this knowledge, we need to incorporate this estimate into our calculation. Suppose we wish to estimate $\int_a^b f(x)dx$ to within an accuracy of tol. We begin by calculating $T_2(a,b)$ and $T_1(a,b)$. If $\frac{1}{3}\left|T_2(a,b)-T_1(a,b)\right| < tol$, we are done. $T_2(a,b)$ is our approximation. In the more likely case that $\frac{1}{3}\left|T_2(a,b)-T_1(a,b)\right| \geq tol$, we divide the interval $[a,b]$ into two subintervals, $[a, \frac{a+b}{2}]$ and $[\frac{a+b}{2}, b]$ and compare our error estimates on these subintervals to $\frac{tol}{2}$. If $\frac{1}{3}\left|T_2(a,\frac{a+b}{2})-T_1(a,\frac{a+b}{2})\right| < \frac{tol}{2}$, we are done with the subinterval $[a,\frac{a+b}{2}]$. $T_2(a,\frac{a+b}{2})$ is a satisfactory approximation of $\int_a^{\frac{a+b}{2}} f(x)dx$. If not, we bisect the interval again and compare error estimates to $\frac{tol}{4}$. On the other half of $[a,b]$, if $\frac{1}{3}\left|T_2(\frac{a+b}{2},b)-T_1(\frac{a+b}{2},b)\right| < \frac{tol}{2}$, we are done with the subinterval $[\frac{a+b}{2},b]$. $T_2(\frac{a+b}{2},b)$ is a satisfactory approximation of $\int_{\frac{a+b}{2}}^{b} f(x)dx$. If not, we bisect the interval again and compare error estimates to $\frac{tol}{4}$. Each time a subinterval fails to meet the error tolerance, we divide it in half and try again. The process will normally end successfully because, with each subinterval division, we will generally have the error decreasing by a factor of 4 while the error requirement is decreasing by a factor of only 2. In the end, the sum of the T_2 estimates where the error tolerance is met will be our approximation for $\int_a^b f(x)dx$.

The simplest way to code this algorithm is to use a recursive function. It is possible to do without, but the record keeping is burdensome. Depending on the programming language you are using, the trade-off may be simplicity for speed. Some languages do not handle recursive functions quickly.

```
Assumptions: $f$ has a continuous second derivative on $[a,b]$.
Input: Function $f$; interval over which to integrate $[a,b]$; tolerance $tol$.
Step 1: Set $m = \frac{b+a}{2}$; $I_1 = T_1(a,b)$; $I_2 = T_2(a,b)$;
Step 2: If $|I_2 - I_1| < 3tol$ then return $I_2$;
Step 3: Do Steps 1-5 with inputs $f$; $[a, \frac{a+b}{2}]$; and $\frac{tol}{2}$; and set $A$ equal to the result;
Step 4: Do Steps 1-5 with inputs $f$; $[\frac{a+b}{2}, b]$; and $\frac{tol}{2}$; and set $B$ equal to the result;
Step 5: Return $A + B$;
Output: Approximate value of $\int_a^b f(x)dx$.
```

A tabulated example of such a computation might help clarify any confusion over how this algorithm works. The following table approximates the integral $\int_0^3 \ln(3+x)dx$ with a tolerance of .006.

a	b	$T_1(a,b)$	$T_2(a,b)$	$\frac{1}{3}\left\|T_2(a,b)-T_1(a,b)\right\|$	tol	
0	3	4.33555	4.42389	.02944	.00600	✗
0	1.5	1.95201	1.96732	.00510	.00300	✗
0	0.75	0.90763	**0.90997**	.00077	.00150	✓
0.75	1.5	1.05968	**1.06124**	.00051	.00150	✓
1.5	3	2.47187	**2.47961**	.00257	.00300	✓

$$\int_0^3 \ln(3+x)dx \approx 0.90997 + 1.06124 + 2.47961 = 4.45082$$

The calculation in the table requires 7 evaluations of f and underestimates the integral by about .00390. In order of occurrence, the evaluations happen at $x = 0, 3, 1.5, .75, .375, 1.125, 2.25$. The composite trapezoidal rule with 7 evaluations (6 subintervals each of length .5) underestimates the integral by about .00346. The non-adaptive composite trapezoidal rule gives a slightly better estimate with essentially the same amount of computation. But remember, it is not necessarily efficiency we are after. It is automatic error estimates. The adaptive trapezoidal rule does something the conventional composite trapezoidal rule does not. It monitors itself for accuracy, so when the routine completes, you not only get an estimate, but you can have some confidence in its accuracy *even when you have no way to calculate the integral exactly* for comparison.

Key Concepts

Composite numerical integration: Dividing the interval of integration into a number of subintervals, applying a simple quadrature formula to each subinterval and summing the results.

Adaptive numerical integration: Leveraging the error term of a simple quadrature formula in order to obtain automatic calculation of the number and nature of subintervals needed to obtain a definite integral with some prescribed accuracy.

Exercises

1. Use the composite midpoint rule with 3 subintervals to approximate

   (a) $\displaystyle\int_1^3 \ln(\sin(x))dx$ [S]

   (b) $\displaystyle\int_5^7 \sqrt{x\cos x}\,dx$

   (c) $\displaystyle\int_1^4 \frac{e^x\ln(x)}{x}dx$ [A]

   (d) $\displaystyle\int_{10}^{13} \sqrt{1+\cos^2 x}\,dx$

   (e) $\displaystyle\int_{\ln 3}^{\ln 7} \frac{e^x}{1+x}dx$ [A]

   (f) $\displaystyle\int_0^1 \frac{x^2-1}{x^2+1}dx$

2. Redo question 1 using the composite trapezoidal rule. [S] [A]

3. Redo question 1 using the composite Simpson's rule. [S] [A]

4. Redo question 1 using the composite Simpson's $\frac{3}{8}$ rule. [S] [A]

5. Redo question 1 using the composite version of the quadrature rule [S] [A]

$$\int_{x_0}^{x_0+3h} f(x)dx = \frac{3h}{2}\left[f(x_0+h)+f(x_0+2h)\right].$$

6. Use a composite version of the quadrature rule

$$\int_{x_0}^{x_0+h} f(x)\,dx \approx \frac{h}{2}\left[f\left(x_0+\frac{h}{3}\right)+f\left(x_0+\frac{2h}{3}\right)\right]$$

   with three subintervals to approximate

$$\int_0^3 \frac{x^3}{x^3+1}dx.$$

7. Use the (simple) trapezoidal rule on $\int_0^\pi \sin^4 x\,dx$ to help estimate the number of intervals $[0,\pi]$ must be divided into in order to approximate $\int_0^\pi \sin^4 x\,dx$ to within 10^{-4} using the *composite* trapezoidal rule. NOTE: $\int_0^\pi \sin^4 x\,dx = \frac{3}{8}\pi$. [S]

8. Repeat question 7 using the midpoint rule. [A]

9. Repeat question 7 using Simpson's rule.

10. Suppose composite Simpson's rule with 100 subintervals was used to estimate $\int_5^{12} f(x)\,dx$, and the absolute error turned out to be less than 10^{-5}. What function might $f(x)$ have been?

11. Derive a summation formula for the composite version of

    (a) the midpoint rule.

    (b) Simpson's rule. [A]

    (c) Simpson's $\frac{3}{8}$ rule. [A]

    (d) the quadrature formula

$$\int_{x_0}^{x_0+h} f(x)\,dx \approx \frac{h}{2}\left[f\left(x_0+\frac{h}{3}\right)+f\left(x_0+\frac{2h}{3}\right)\right].$$

12. Based on our discussion of composite integration, the error term for composite Simpson's rule applied to $\int_a^b f(x)\,dx$ with n subintervals is $O\left(\left(\frac{1}{n}\right)^4 f^{(4)}(\xi_n)\right)$. With a bit more work, it can be shown that the error term is actually $-\frac{b-a}{90}h^4 f^{(4)}(\xi_n)$ where $h = \frac{b-a}{n}$. No big-oh needed. This error is exact for some $\xi_n \in [a,b]$. Use this error term to find a theoretical bound on the error in estimating

$$\int_2^4 \frac{1}{1-x}\,dx$$

    using (composite) Simpson's rule with $h = 0.1$.

13. Why does the composite trapezoidal rule ALWAYS (for any h) give an underestimate of

$$\int_0^\pi \sin x\,dx?$$

14. Demonstrate geometrically and with some words the approximation of $\int_7^8 \frac{x\sin x}{8}\,dx$ using the composite trapezoidal rule with 4 trapezoids (that is, 4 subintervals).

15. Approximate $\int_1^3 \ln(\sin(x))dx$ using adaptive Simpson's method with tolerance 0.002. [S]

16. Use adaptive Simpson's method to approximate $\displaystyle\int_0^1 \ln(x+1)dx$ accurate to within 10^{-4}. [A]

17. Derive a quadrature formula for

$$\int_a^b f(x)\,dx$$

    using unspecified nodes $a \le x_0 < x_1 \le b$. In other words, derive a "general trapezoidal rule" where x_0 and x_1 are allowed to be any two distinct values in $[a,b]$.

18. In your formula from question 17, make the substitutions $x_0 = a$, $x_1 = b$, and $x_1 - x_0 = h$, and show that it thus reduces to the trapezoidal rule.

19. Let $I = \int_0^2 x^2\ln(x^2+1)\,dx$. [A]

    (a) Approximate I using the Midpoint rule.

    (b) Use your answer to (a) to estimate the number of subintervals needed to approximate I to within 10^{-4}. NOTE: $I = \frac{24\ln(5)-6\tan^{-1}(2)-4}{9}$.

20. Let $I = \int_0^2 x^2\ln(x^2+1)\,dx$.

    (a) Approximate I using Simpson's rule.

    (b) Use your answer to (a) to estimate the number of subintervals needed to approximate I to within 10^{-4}. NOTE: $I = \frac{24\ln(5)-6\tan^{-1}(2)-4}{9}$.

21. Use the computer to calculate the estimate suggested in question 19b. Is the absolute error less than 10^{-4}? [A]

22. Use the computer to calculate the estimate suggested in question 20b. Is the absolute error less than 10^{-4}?

23. Use the composite trapezoidal rule to estimate $\displaystyle\int_0^1 \ln(x+1)dx$ accurate to within 10^{-6}. How many subintervals are needed? [𝕊]

24. Repeat question 23 using the composite midpoint rule.

25. Use composite Simpson's rule to estimate $\displaystyle\int_0^1 \ln(x+1)dx$ accurate to within 10^{-6}. How many subintervals are needed?

26. Repeat question 25 using composite Simpson's $\frac{3}{8}$ rule. [𝔸]

27. Write computer code that implements adaptive Simpson's rule as a recursive function. Some notes about the structure: [𝔸]

    (a) The inputs to the function should be $f(x)$, a, b, and a maximum overall error, *tol*.

    (b) The output of the function should be the estimate and, if you are feeling particularly stirred, the number of function evaluations.

28. Use your code from question 27 to approximate $\int_1^3 \ln(\sin(x))dx$ with tolerance 0.002. [𝔸]

29. Use your code from question 27 to approximate $\displaystyle\int_0^1 \ln(x+1)dx$ accurate to within 10^{-4}.

30. (i) Use your code from question 27 to approximate the integral using $tol = 10^{-5}$. (ii) Calculate the actual error of the approximation. (iii) Is the approximation accurate to within 10^{-5} as requested?

    (a) $\displaystyle\int_0^{2\pi} x\sin(x^2)dx$ [𝔸]

    (b) $\displaystyle\int_{0.1}^{2} \frac{1}{x}\,dx$

    (c) $\displaystyle\int_0^{2} x^2\ln(x^2+1)\,dx$

    NOTE: $\int_0^2 x^2\ln(x^2+1)\,dx = \frac{24\ln(5)-6\tan^{-1}(2)-4}{9}$.

31. Write computer code that implements the general trapezoidal rule of question 1 in such a way that x_0 and x_1 are chosen at random.

32. Write computer code that implements a composite version of the quadrature method in question 31.

33. Do some numerical experiments to compare the (standard) composite trapezoidal rule to the (random) composite trapezoidal rule of question 32. What do you find?

4.5 Extrapolation

In calculus, you undoubtedly encountered Euler's constant, e, which you were probably told is approximately 2.718, or maybe just 2.7. And unless you were involved in a digits-of-e memorization contest, you probably never saw more digits of e than your calculator could show. We're about to change that. The first 50 digits of e are

$$2.7182818284590452353602874713526624977572470936999.$$

How many of them do you remember? Not to worry if it is not very many. No quiz on the digits of e is imminent.

Crumpet 27: Digits of e

The first 1000 digits of e, 50 per line, are

2.7182818284590452353602874713526624977572470936999

59574966967627724076630353547594571382178525166427

42746639193200305992181741359662904357290033429526

05956307381323286279434907632338298807531952510190

11573834187930702154089149934884167509244761460668

08226480016847741185374234544243710753907774499206

95517027618386062613313845830007520449338265602976

06737113200709328709127443747047230696977209310141

69283681902551510865746377211125238978442505695369

67707854499699679468644549059879316368892300987931

27736178215424999229576351482208269895193668033182

52886939849646510582093923982948879332036250944311

73012381970684161403970198376793206832823764648042

95311802328782509819455815301756717361332069811250

99618188159304169035159888851934580727386673858942

28792284998920868058257492796104841984443634632449

68487560233624827041978623209002160990235304369941

84914631409343173814364054625315209618369088870701

67683964243781405927145635490613031072085103837505

10115747704171898610687396965521267154688957035035

However, do you recall from calculus that

$$\lim_{h \to 0} (1+h)^{1/h} = e?$$

Can you prove it? Proof on page 155. Based on this fact, we might use

$$\tilde{e}(h) = (1+h)^{1/h}$$

to approximate e. No time like the present!

$$\begin{aligned}
\tilde{e}(0.01) &\approx 2.704813829421529\\
\tilde{e}(0.005) &\approx 2.711517122929293\\
\tilde{e}(0.0025) &\approx 2.714891744381238\\
\tilde{e}(0.00125) &\approx 2.716584846682473\\
\tilde{e}(0.000625) &\approx 2.717432851769196.
\end{aligned}$$

Sadly, this sequence of approximations is not converging very quickly. We have two digits of accuracy in the first approximation and still only three digits of accuracy in the fifth. We could, of course, continue to make h smaller to get more accurate approximations, but based on the slow improvement observed so far, this does not seem like a very promising route. Instead, we can combine the estimates we already have to get an improved approximation. This idea should remind you, at least on the surface, of Aitken's delta-squared method. In that method, we combined three consecutive approximations to form another that was generally a better approximation than any of the original three. We will do something similar here, combining inadequate approximations to find better ones. We will name the various new approximations for continued reuse.

$$\begin{aligned}
2\tilde{e}(0.005) - \tilde{e}(0.01) &\equiv \tilde{e}_1(0.01) = 2.718220416437056\\
2\tilde{e}(0.0025) - \tilde{e}(0.005) &\equiv \tilde{e}_1(0.005) = 2.718266365833184\\
2\tilde{e}(0.00125) - \tilde{e}(0.0025) &\equiv \tilde{e}_1(0.0025) = 2.718277948983707\\
2\tilde{e}(0.000625) - \tilde{e}(0.00125) &\equiv \tilde{e}_1(0.00125) = 2.718280856855920. \qquad (4.5.1)
\end{aligned}$$

Each of these new approximations is accurate to 5 or 6 significant digits! Already a significant improvement. We can combine them further to find yet better approximations:

$$\begin{aligned}
\frac{4\tilde{e}_1(0.005) - \tilde{e}_1(0.01)}{3} &\equiv \tilde{e}_2(0.01) = 2.718281682298560\\
\frac{4\tilde{e}_1(0.0025) - \tilde{e}_1(0.005)}{3} &\equiv \tilde{e}_2(0.005) = 2.718281810033881\\
\frac{4\tilde{e}_1(0.00125) - \tilde{e}(0.0025)}{3} &\equiv \tilde{e}_2(0.0025) = 2.718281826146657. \qquad (4.5.2)
\end{aligned}$$

The first of these approximations is accurate to seven significant digits, the second to eight, and the third to nine! And we can combine them further:

$$\begin{aligned}
\frac{8\tilde{e}_2(0.005) - \tilde{e}_2(0.01)}{7} &\equiv \tilde{e}_3(0.01) = 2.718281828281785\\
\frac{8\tilde{e}_2(0.0025) - \tilde{e}_2(0.005)}{7} &\equiv \tilde{e}_3(0.005) = 2.718281828448482. \qquad (4.5.3)
\end{aligned}$$

Now we have approximations accurate to ten and eleven significant digits! Looking back, we took five approximations that had no better than 3 significant digits of accuracy and combined them to get two approximations that were accurate to at least 10 significant digits each. Magic! Okay, not magic, mathemagic! Here is how it works.

Suppose we are approximating p using the formula $\tilde{p}(h)$, and we know that

$$\tilde{p}(h) = p + c_1 \cdot h^{m_1} + c_2 \cdot h^{m_2} + c_3 \cdot h^{m_3} + \cdots .$$

Then

$$\tilde{p}\left(\alpha h\right) = p + c_1 \cdot \left(\alpha h\right)^{m_1} + c_2 \cdot \left(\alpha h\right)^{m_2} + c_3 \cdot \left(\alpha h\right)^{m_3} + \cdots .$$

Now, if we multiply the second equation by α^{-m_1} and subtract the first from it, the h^{m_1} terms vanish, and we get an approximation with error term beginning with $c_2 \cdot h^{m_2}$:

$$\begin{array}{rcl}
\alpha^{-m_1}\tilde{p}\left(\alpha h\right) & = & \alpha^{-m_1}p + c_1 \cdot h^{m_1} + c_2\alpha^{m_2-m_1} \cdot h^{m_2} + c_3\alpha^{m_3-m_1} \cdot h^{m_3} + \cdots\\
-\left[\tilde{p}(h)\right. & = & \left. p + c_1 \cdot h^{m_1} + c_2 \cdot h^{m_2} + c_3 \cdot h^{m_3} + \cdots\right]\\ \hline
\alpha^{-m_1}\tilde{p}\left(\alpha h\right) - \tilde{p}(h) & = & (\alpha^{-m} - 1)p + c_2(\alpha^{m_2-m_1} - 1) \cdot h^{m_2} + c_3(\alpha^{m_3-m_1} - 1) \cdot h^{m_3} + \cdots
\end{array}$$

With a little rearranging,

$$\frac{\alpha^{-m_1}\tilde{p}\left(\alpha h\right) - \tilde{p}(h)}{\alpha^{-m_1} - 1} = p + d_2 \cdot h^{m_2} + d_3 \cdot h^{m_3} + \cdots \qquad (4.5.4)$$

for some constants $d_2, d_3, \ldots$. If $m_2 > m_1$, then this method will tend to improve on the two approximations $\tilde{p}(h)$ and $\tilde{p}\left(\alpha h\right)$ by combining them into a single approximation with error commensurate with some constant multiple of h^{m_2}. This calculation is the basis for Richardson's extrapolation.

It just so happens $\tilde{e}(h)$ has exactly the form needed.

$$\tilde{e}(h) = e + c_1 h + c_2 h^2 + c_3 h^3 + c_4 h^4 + O(h^5) \qquad (4.5.5)$$

for some constants c_1, c_2, c_3, c_4. The actual values of the constants are not relevant for this computation. To understand the computation of $\tilde{e}_1$, we use equation 4.5.4 with $\alpha = \frac{1}{2}$ and $m_1 = 1$ to get

$$\begin{aligned}\tilde{e}_1(h) &= \frac{2\tilde{e}\left(\frac{h}{2}\right) - \tilde{e}(h)}{2-1} \\ &= 2e + c_1h + \frac{1}{2}c_2h^2 + \frac{1}{4}c_3h^3 + \frac{1}{8}c_4h^4 + O(h^5) \\ &\qquad - \left[e + c_1h + c_2h^2 + c_3h^3 + c_4h^4 + O(h^5)\right] \\ &= e + d_2h^2 + d_3h^3 + d_4h^4 + O(h^5)\end{aligned}$$

for some constants d_2, d_3, d_4. $\tilde{e}_1(h)$ is the formula that gave us the round of approximations accurate to 5 or 6 significant digits. It is not hard to find the constants d_i in terms of the constants c_i, but, again, the values of the constants are immaterial and can only serve to complicate further refinements. What is important is the form of the error. Now that we know $\tilde{e}_1(h) = e + d_2h^2 + d_3h^3 + d_4h^4 + O(h^5)$, we find $\tilde{e}_2(h)$ using formula 4.5.4 with $\alpha = \frac{1}{2}$ and $m_1 = 2$:

$$\begin{aligned}\tilde{e}_2(h) &= \frac{4\tilde{e}_1\left(\frac{h}{2}\right) - \tilde{e}_1(h)}{3} \\ &= e + k_3h^3 + k_4h^4 + O(h^5)\end{aligned}$$

for some constants k_3 and k_4. $\tilde{e}_2(h)$ is the formula that gave us the round of approximations accurate to 7 to 9 significant digits. We can again use formula 4.5.4, this time with $\alpha = \frac{1}{2}$ and $m_1 = 3$:

$$\begin{aligned}\tilde{e}_3(h) &= \frac{8\tilde{e}_2\left(\frac{h}{2}\right) - \tilde{e}_2(h)}{7} \\ &= e + l_4h^4 + O(h^5)\end{aligned}$$

for some constant l_4. $\tilde{e}_3(h)$ is the formula that gave us the approximations accurate to 10 and 11 significant digits. Now is a good time to see if you can use the expression for $\tilde{e}_3(h)$ and formula 4.5.4 to derive an $O(h^5)$ formula for $\tilde{e}_4(h)$. Then use your formula to compute $\tilde{e}_4(0.01)$ using the previously given values of $\tilde{e}_3(0.01)$ and $\tilde{e}_3(0.005)$. How accurate is $\tilde{e}_4(0.01)$? Answers on page 155.

As a special case, Richardson's extrapolation with $\alpha = \frac{1}{2}$ applied to any approximation of the form

$$\tilde{p}_0(h) = p + c_1h + c_2h^2 + c_3h^3 + \cdots$$

gives the recursively defined refinements

$$\tilde{p}_k(h) = \frac{2^k\tilde{p}_{k-1}\left(\frac{h}{2}\right) - \tilde{p}_{k-1}(h)}{2^k - 1}, \quad k = 1, 2, 3, \ldots$$

which are expected to increase in accuracy as k increases. For other α or other forms of error, the formula for $\tilde{p}_k(h)$ changes according to 4.5.4.

Crumpet 28: A Taylor polynomial for $\tilde{e}(h)$

$\tilde{e}$ is undefined at 0, so its derivatives at 0 are as well. However, if we extend the definition of $\tilde{e}$ to

$$\tilde{e}(h) = \begin{cases}(1+h)^{1/h} & \text{if } h \neq 0 \\ e & \text{if } h = 0\end{cases},$$

thus defining $\tilde{e}$ at 0, then $\tilde{e}(h)$ becomes infinitely differentiable at 0, and its fifth Taylor polynomial, for example, is:

$$\tilde{e}(h) = e - \frac{e}{2}\cdot h + \frac{11e}{24}\cdot h^2 - \frac{7e}{16}\cdot h^3 + \frac{2447e}{5760}\cdot h^4 + \frac{f^{(5)}(\xi)}{120}h^5$$

for some $\xi \in (0, h)$.

Differentiation

Using extrapolation, high order differentiation approximation formulas can be derived from low order formulas. We begin with the lowest order approximation, $f'(x_0) = \dfrac{-f(x_0)+f(x_0+h)}{h} - \dfrac{h}{2}f''(\xi_h)$. The standard error term, $-\frac{h}{2}f''(\xi_h)$ does not give the error in the form $c \cdot h^{m_1} + O(h^{m_2})$ as required by Richardson's extrapolation, so we return to Taylor series to determine the $O(h^{m_2})$ term:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{1}{2}h^2 f''(x_0) + \frac{1}{6}h^3 f'''(x_0) + \cdots$$

so

$$\frac{-f(x_0)+f(x_0+h)}{h} = f'(x_0) + \frac{1}{2}hf''(x_0) + \frac{1}{6}h^2 f'''(x_0) + \cdots.$$

Hence,

$$\begin{aligned} f'(x_0) - \frac{-f(x_0)+f(x_0+h)}{h} &= -\frac{1}{2}hf''(x_0) - \frac{1}{6}h^2 f'''(x_0) - \cdots \\ &= c_1 h + O(h^2) \end{aligned}$$

and extrapolation will yield an $O(h^2)$ formula. Letting $\tilde{p}(h) = \frac{-f(x_0)+f(x_0+h)}{h}$, $\alpha = 2$, and $m_1 = 1$, formula 4.5.4 tells us the approximation

$$\frac{\frac{1}{2}\tilde{p}(2h) - \tilde{p}(h)}{\frac{1}{2} - 1}$$

will be an $O(h^2)$ formula for $f'(x_0)$. Simplifying,

$$\begin{aligned} \frac{\frac{1}{2}\tilde{p}(2h) - \tilde{p}(h)}{\frac{1}{2} - 1} &= \frac{\frac{1}{2}\left[\frac{-f(x_0)+f(x_0+2h)}{2h}\right] - \frac{-f(x_0)+f(x_0+h)}{h}}{-\frac{1}{2}} \\ &= \frac{\frac{-f(x_0)+f(x_0+2h)}{4h} - \frac{-4f(x_0)+4f(x_0+h)}{4h}}{-\frac{1}{2}} \\ &= \frac{\frac{3f(x_0)-4f(x_0+h)+f(x_0+2h)}{4h}}{-\frac{1}{2}} \\ &= \frac{-3f(x_0)+4f(x_0+h)-f(x_0+2h)}{2h}. \end{aligned}$$

Hence, we have $f'(x_0) = \frac{-3f(x_0)+4f(x_0+h)-f(x_0+2h)}{2h} + O(h^2)$, but this is not news. This is the first 3-point formula in table 4.2! Other high order derivative formulas can be derived by extrapolation too, but, generally, nothing new is learned from the result. We simply have a new way of deriving high order differentiation formulas.

Integration

Applying extrapolation to definite integrals is more rewarding. We begin with any composite integration formula and apply Richardson's extrapolation. We now consider the composite trapezoidal rule and use the notation $T_k(a,b)$ to represent the approximation of $\int_a^b f(x)dx$ using the trapezoidal rule with k subintervals.

Before continuing we need to have a good idea what it means for the composite trapezoidal rule to have error term $O\left(\left(\frac{1}{n}\right)^2\right)$. In essence, it means we should expect the error to decrease by a factor of about 4 when the number of intervals is doubled. We should expect the error to decrease by a factor of about 9 when the number of intervals is tripled. And generally we should expect the error to decrease by a factor of about β^2 when the number of intervals is multiplied by β. To see this effect in action, consider the definite integral

$$\int_0^1 \sin x \, dx$$

whose exact value is $1 - \cos(1) \approx .4596976941318602$. The absolute errors of $T_5(0,1)$, $T_{10}(0,1)$, and $T_{15}(0,1)$ are

$$\begin{aligned}
\left|\int_0^1 \sin x\,dx - T_5(0,1)\right| &\approx 1.533(10)^{-3} \\
\left|\int_0^1 \sin x\,dx - T_{10}(0,1)\right| &\approx 3.831(10)^{-4} \\
\left|\int_0^1 \sin x\,dx - T_{15}(0,1)\right| &\approx 1.702(10)^{-4}
\end{aligned}$$

We should expect the error $\left|\int_0^1 \sin x\,dx - T_5(0,1)\right|$ to be about four times the error $\left|\int_0^1 \sin x\,dx - T_{10}(0,1)\right|$ and nine times the error $\left|\int_0^1 \sin x\,dx - T_{15}(0,1)\right|$. To check, we compute the ratios:

$$\begin{aligned}
\frac{\left|\int_0^1 \sin x\,dx - T_5(0,1)\right|}{\left|\int_0^1 \sin x\,dx - T_{10}(0,1)\right|} &= \frac{1.533(10)^{-3}}{3.831(10)^{-4}} \approx 4.001 \\
\frac{\left|\int_0^1 \sin x\,dx - T_5(0,1)\right|}{\left|\int_0^1 \sin x\,dx - T_{15}(0,1)\right|} &= \frac{1.533(10)^{-3}}{1.702(10)^{-4}} \approx 9.007.
\end{aligned}$$

What should you expect the ratio $\frac{\left|\int_0^1 \sin x\,dx - T_{10}(0,1)\right|}{\left|\int_0^1 \sin x\,dx - T_{15}(0,1)\right|}$ to be about? Answer on page 155.

Finally, we apply Richardson's extrapolation with $\alpha = \frac{1}{2}$ and $m_1 = 2$ to produce the higher order estimate,

$$T_{k,1}(a,b) \equiv \frac{4T_{2k}(a,b) - T_k(a,b)}{3}.$$

We defer to numerics to get a handle on the error term of the refinement $T_{k,1}$. We begin by collecting some data. Continuing with the analysis of $\int_0^1 \sin x\,dx$, note that

$$\begin{aligned}
T_5(0,1) &\approx .4581643459604436 \\
T_{10}(0,1) &\approx .4593145488579763 \\
T_{20}(0,1) &\approx .4596019197882473 \\
T_{40}(0,1) &\approx .4596737512942187.
\end{aligned}$$

Hence,

$$\begin{aligned}
T_{5,1}(0,1) &= \frac{4T_{10}(0,1) - T_5(0,1)}{3} \approx .4596979498238206 \\
T_{10,1}(0,1) &= \frac{4T_{20}(0,1) - T_{10}(0,1)}{3} \approx .4596977100983375 \\
T_{20,1}(0,1) &= \frac{4T_{40}(0,1) - T_{20}(0,1)}{3} \approx .4596976951295424
\end{aligned}$$

and

$$\begin{aligned}
\frac{\left|\int_0^1 \sin x\,dx - T_{5,1}(0,1)\right|}{\left|\int_0^1 \sin x\,dx - T_{10,1}(0,1)\right|} &\approx 16.01 \\
\frac{\left|\int_0^1 \sin x\,dx - T_{10,1}(0,1)\right|}{\left|\int_0^1 \sin x\,dx - T_{20,1}(0,1)\right|} &\approx 16.00.
\end{aligned}$$

When we double the number of subintervals, the error is decreased by a factor of 16. That's 2^4, not 2^3 as we might have expected! The first refinement takes us from a $O\left(\left(\frac{1}{n}\right)^2\right)$ approximation to a $O\left(\left(\frac{1}{n}\right)^4\right)$ approximation. In other words, the error of $T_{n,1}$ is $O\left(\left(\frac{1}{n}\right)^4\right)$.

Table 4.7: Romberg's method

T_1	$T_{1,1}$	$T_{1,2}$	$T_{1,3}$	$\cdots$
T_2	$T_{2,1}$	$T_{2,2}$	$\vdots$	
T_4	$T_{4,1}$	$\vdots$		
T_8	$\vdots$			
$\vdots$				

Now that we know the error of $T_{n,1}$ is $O\left(\left(\frac{1}{n}\right)^4\right)$ we can extrapolate again. Applying Richardson's extrapolation with $\alpha = \frac{1}{2}$ and $m_1 = 4$, we have

$$\begin{aligned} T_{5,2}(0,1) &= \frac{16T_{10,1}(0,1) - T_{5,1}(0,1)}{15} \approx .4596976941166387 \\ T_{10,2}(0,1) &= \frac{16T_{20,1}(0,1) - T_{10,1}(0,1)}{15} \approx .4596976941316228. \end{aligned}$$

We now have approximations $T_{5,2}$ and $T_{10,2}$ whose errors are only about $1.522(10)^{-11}$ and $2.374(10)^{-13}$, respectively. Use this information to calculate $T_{5,3}$ and its absolute error. Answers on page 155.

The method of combining Richardson's extrapolation with the trapezoidal rule is known as Romberg's method or Romberg integration. The calculation is often tabulated for organizational purposes as in Table 4.7. Rows are added until the differences $|T_{k,n} - T_{k,n+1}|$ and $|T_{2k,n} - T_{k,n+1}|$ are both less than some tolerance.

Though Richardson's extrapolation may be applied to any composite integration formula, the computations of the error terms above help explain why the trapezoidal rule is the right one to use. We might infer from our calculations (and it can be proven true) that the error term of the composite trapezoidal rule contains only even powers of $\frac{1}{n}$. To be explicit, we have

$$\int_a^b f(x)dx = T_n(a,b) + c_2\left(\frac{1}{n}\right)^2 + c_4\left(\frac{1}{n}\right)^4 + c_6\left(\frac{1}{n}\right)^6 + \cdots$$

so each refinement increases the least degree in the error term by 2, not 1. Skipping the odd degrees makes this particular choice very efficient. But this method comes with a price. Hidden within c_2 is the assumption that f has a continuous second derivative. Hidden within c_4 is the assumption that f has a continuous fourth derivative. And so on. The accuracy of each refinement depends on f having two more continuous derivatives. The more refinements we do, the smoother f must be for this method to work. For this reason, it is advisable to use Romberg's method only when the integrand is known to have sufficient derivatives.

Key Concepts

Richardson's extrapolation: If approximation $\tilde{p}$ is know to have the form

$$\tilde{p}(h) = p + c_1 h^{m_1} + O(h^{m_2})$$

then the approximation

$$\frac{\alpha^{-m_1}\tilde{p}(\alpha h) - \tilde{p}(h)}{\alpha^{-m_1} - 1}$$

will have error $O(h^{m_2})$.

Romberg integration: The application of Richardson's extrapolation to the trapezoidal method.

Exercises

1. One can use Taylor Polynomials to show that

$$\pi = \frac{1}{h}\sin(h\pi) + K_2h^2 + K_4h^4 + K_6h^6 + \cdots.$$

Therefore, $N(h) = \frac{1}{h}\sin(h\pi)$ is an $O(h^2)$ approximation of π. Use Richardson's extrapolation to derive an $O(h^4)$ approximation of π. [A]

2. It is interesting to note that we can reverse engi-

neer Richardson refinements in order to approximate the c_i of equation 4.5.5 on page 149. For example, $\tilde{e}(h) = e + c_1 h + O(h^2)$, and we assume the $O(h^2)$ term is relatively small, so we can rearrange this equation to find

$$\frac{\tilde{e}(h) - e}{h} \approx c_1.$$

To take a specific example, $\frac{\tilde{e}(.005)-e}{.005} = \frac{2.711517122929293-e}{.005} \approx -1.35$ so $c_1 \approx -1.35$. If we pay careful attention to how the constants are affected as we refine our initial approximations, we can find c_2, c_3, and c_4 as well.

$$\begin{aligned}\tilde{e}_1(h) &= 2\tilde{e}\left(\frac{h}{2}\right) - \tilde{e}(h)\\ &= 2e + c_1 h + \frac{c_2}{2}h^2 + \frac{c_3}{4}h^3 + \frac{c_4}{8}h^4 + O(h^5)\\ &\quad -(e + c_1 h + c_2 h^2 + c_3 h^3 + c_4 h^4 + O(h^5))\\ &= e - \frac{c_2}{2}h^2 - \frac{3c_3}{4}h^3 - \frac{7c_4}{8}h^4 + O(h^5).\end{aligned}$$

Therefore, $\tilde{e}_1(h) - e \approx -\frac{c_2}{2}h^2$, from which we conclude

$$\frac{-2(\tilde{e}_1(h) - e)}{h^2} \approx c_2.$$

(a) Use this formula and the values in 4.5.1 to verify that $c_2 \approx 1.24$.

(b) Approximate c_3 using values in 4.5.2.

(c) Approximate c_4 using values in 4.5.3.

(d) Compare these approximations of c_1, c_2, c_3, c_4 to the exact values in crumpet 28.

3. Suppose N approximates M according to $N(h) = M + K_1 h^3 + K_2 h^5 + K_3 h^7 + \cdots$. Of what order will $N_3(h)$ (the third generation Richardson's extrapolation) be? [A]

4. Suppose N approximates M according to $N(h) = M + K_1 h^2 + K_2 h^4 + K_3 h^6 + \cdots$. What would you expect the value of

$$\frac{|M - N(h/3)|}{|M - N(h/4)|}$$

to be for small h, approximately? [A]

5. $N(h) = \frac{1-\cos h}{h^2}$ can be used to approximate [A]

$$\lim_{h\to 0} \frac{1 - \cos h}{h^2}$$

(a) Compute $N(1.0)$ and $N(0.5)$.

(b) Compute $N_1(1.0)$, the first Richardson's extrapolation, assuming

i. $N(h)$ has an error of the form $K_1 h + K_2 h^2 + K_3 h^3 + \cdots$

ii. $N(h)$ has an error of the form $K_2 h^2 + K_4 h^4 + K_6 h^6 + \cdots$

(c) Which of the assumptions in part 5b do you think gives the correct error and why?

6. The backward difference formula can be expressed as

$$\begin{aligned}f'(x_0) = \frac{1}{h}[f(x_0) - f(x_0 - h)]\\ + \frac{h}{2}f''(x_0) - \frac{h^2}{6}f'''(x_0) + O(h^3)\end{aligned}$$

(a) Use Richardson's extrapolation to derive an $O(h^2)$ formula for $f'(x_0)$.

(b) The formula you derived should look familiar. What formula does it look like? Is it exactly the same? Why or why not?

7. Derive an $O(h^3)$ formula for approximating M that uses $N(h)$, $N(\frac{h}{2})$, and $N(\frac{h}{3})$, and is based on the assumption that [S]

$$M = N(h) + K_1 h + K_2 h^2 + K_3 h^3 + \cdots.$$

8. The following data give estimates of the integral $M = \int_0^{3\pi/2} \cos x\, dx$.

$$N(h) = 2.356194 \qquad N(h/2) = -0.4879837$$
$$N(h/4) = -0.8815732 \qquad N(h/8) = -0.9709157$$

Assuming $M - N(h) = K_1 h^2 + K_2 h^4 + K_3 h^6 + \cdots$, find a third Richardson's extrapolation for M. [S]

9. Suppose that $N(h)$ is an approximation of M for every $h > 0$ and that

$$M - N(h) = K_1 h + K_2 h^2 + K_3 h^3 + \cdots$$

for some constants $K_1, K_2, K_3, \ldots$. Use the values $N(h)$, $N(h/3)$, and $N(h/9)$ to produce an $O(h^3)$ approximation of M. [A]

10. Use Romberg integration to compute the integral with tolerance 10^{-4}.

(a) $\displaystyle\int_1^3 \ln(\sin(x))dx$ [S]

(b) $\displaystyle\int_5^7 \sqrt{x\cos x}\,dx$

(c) $\displaystyle\int_1^4 \frac{e^x \ln(x)}{x}dx$ [A]

(d) $\displaystyle\int_{10}^{13} \sqrt{1 + \cos^2 x}\,dx$

(e) $\displaystyle\int_{\ln 3}^{\ln 7} \frac{e^x}{1+x}dx$ [A]

(f) $\displaystyle\int_0^1 \frac{x^2-1}{x^2+1}dx$

(g) $\displaystyle\int_0^2 x^2 \ln(x^2+1)dx$ [A]

11. Write a Romberg integration function on the computer. [A]

12. (i) Use your code from question 11 to approximate the integral using $tol = 10^{-5}$. (ii) Calculate the actual error of the approximation. (iii) Is the approximation accurate to within 10^{-5} as requested?

(a) $\displaystyle\int_0^{2\pi} x\sin(x^2)dx$ [A]

(b) $\displaystyle\int_{0.1}^2 \frac{1}{x}\,dx$

(c) $\displaystyle\int_0^2 x^2 \ln(x^2+1)\,dx$

NOTE: $\int_0^2 x^2 \ln(x^2+1)\,dx = \frac{24\ln(5) - 6\tan^{-1}(2) - 4}{9}$.

13. Compare the results of question 12 with those of question 30 on page 147.

Answers

$\lim\limits_{h\to 0}(1+h)^{1/h}=e$**:** Begin by noting $\ln\left[(1+h)^{1/h}\right]=\frac{\ln(1+h)}{h}$. Set

$$\begin{aligned} L &= \lim_{h\to 0}\frac{\ln(1+h)}{h} \\ &= \lim_{h\to 0}\frac{\frac{d}{dh}(\ln(1+h))}{\frac{d}{dh}(h)} \\ &= \lim_{h\to 0}\frac{1}{1+h} \\ &= 1. \end{aligned}$$

Thus $L=1$, and due to continuity of the exponential function, e^x,

$$\begin{aligned} e=e^L=e^{\lim_{h\to 0}\frac{\ln(1+h)}{h}}=\lim_{h\to 0}e^{\frac{\ln(1+h)}{h}} &= \lim_{h\to 0}e^{\ln\left[(1+h)^{1/h}\right]} \\ &= \lim_{h\to 0}(1+h)^{1/h}. \end{aligned}$$

$\tilde{e}_4(h)$**:** We use formula 4.5.4 with $\alpha=\frac{1}{2}$, $m=4$, and $n=5$ to find

$$\begin{aligned} \tilde{e}_4(h) &= \frac{16\tilde{e}_3\left(\frac{h}{2}\right)-\tilde{e}_3(h)}{15} \\ &= e+O(h^5). \end{aligned}$$

Applying this formula to $\tilde{e}_3(0.01)$ and $\tilde{e}_3(0.005)$ we get

$$\begin{aligned} \tilde{e}_4(0.01) &= \frac{16(2.718281828448482)-2.718281828281785}{15} \\ &= 2.718281828459595, \end{aligned}$$

a value that is accurate to 13 significant digits!

error ratio: We should expect $\frac{\left|\int_0^1 \sin x\,dx-T_{10}\right|}{\left|\int_0^1 \sin x\,dx-T_{15}\right|}$ to be about $1.5^2=2.25$ because 15 (the number of intervals used in the approximation of the denominator) is 1.5 times 10 (the number of intervals used in the approximation of the numerator).

$T_{5,3}$ **and its error:** $\frac{\left|\int_0^1 \sin x\,dx-T_{5,2}\right|}{\left|\int_0^1 \sin x\,dx-T_{10,2}\right|}\approx\frac{1.522(10)^{-11}}{2.374(10)^{-13}}\approx 64$ so

$$\begin{aligned} T_{5,3} &= \frac{64T_{10,2}-T_{5,2}}{63}\approx .4596976941318606 \\ \left|\int_0^1 \sin x\,dx-T_{5,3}\right| &\approx 4(10)^{-16} \end{aligned}$$

Chapter 5

More Interpolation

5.1 Osculating Polynomials

The Taylor polynomials of Section 1.2 and interpolating polynomials of Chapter 3 represent opposite extremes in the spectrum of osculating polynomials. Taylor polynomials require the value of the polynomial at a single point while interpolating polynomials require the value of the polynomial at, generally anyway, multiple points. Taylor polynomials require the values of, generally anyway, multiple derivatives while interpolating polynomials do not allow derivative specification.

The set of osculating polynomials contains Taylor polynomials, interpolating polynomials, and hybrids. Any polynomial required to pass through any set of points with any number of derivatives specified at those points is called an osculating polynomial. Thus a Taylor polynomial is the special case of an osculating polynomial specified by one point and any number of derivatives at that point. An interpolating polynomial is the special case of an osculating polynomial specified by any number of points and no derivatives at any point. To be precise, an osculating polynomial is one that is required to pass through a set of points

$$(t_0, y_0), (t_1, y_1), \dots, (t_n, y_n)$$

with the first m_i derivatives specified at (t_i, y_i), $i = 0, 1, \dots, n$. As before, the $t_0, t_1, \dots, t_n$ are called nodes.

One useful type of osculating polynomial is the Hermite polynomial in which the value of the polynomial and its first derivative are both given at each node. Even more specifically, third degree, or cubic, Hermite polynomials play an important role in approximation theory. Since a third degree polynomial has four parameters, data—the ordinate and first derivative—at two nodes is sufficient to specify such a polynomial. So suppose we wish to find a polynomial p of degree at most three that passes through (t_0, y_0) and (t_1, y_1) with derivative $\dot{y}_0$ at t_0 and $\dot{y}_1$ at t_1.

Remembering the lessons of our study of interpolating polynomials, we might begin with the Lagrange form of the interpolating polynomial passing through (t_0, y_0) and (t_1, y_1) and worry about the derivatives later. That gives us $f(t) = \frac{t-t_1}{t_0-t_1} y_0 + \frac{t-t_0}{t_1-t_0} y_1$ to begin. Of course f passes through the required points, but it is not even potentially cubic, and its derivative is $f'(t) = \frac{y_0}{t_0-t_1} + \frac{y_1}{t_1-t_0}$, a constant. It would be nice if we could add to it, a third degree polynomial that has zeroes at t_0 and t_1 and whose derivatives we can control. Well, $g(t) = (t-t_0)(t-t_1)^2$, for example, is cubic, has zeroes at t_0 and t_1, and has derivative $(t-t_1)^2 + 2(t-t_0)(t-t_1)$, so we have at least some control over its derivative. Great, now let us look at it a little more closely:

$$g'(t) = (t-t_1)^2 + 2(t-t_0)(t-t_1) = (t-t_1)\left[(t-t_1) + 2(t-t_0)\right].$$

So $g'(t_1) = 0$ and $g'(t_0) = (t_0 - t_1)^2$ is nonzero. That should remind you of how we developed the Lagrange interpolating polynomial. Only, there, the value of the polynomial was either 0 or 1 at each node before we added an unknown coefficient. Of course, $\hat{g}(t) = \frac{g(t)}{(t_0-t_1)^2}$ has derivative 1 at t_1 and 0 at t_0. Putting it all together, $\hat{g}_a(t) = a\frac{(t-t_0)(t-t_1)^2}{(t_0-t_1)^2}$ has everything we need to control the derivative at t_0. Similarly, $\hat{h}_b(t) = b\frac{(t-t_0)^2(t-t_1)}{(t_1-t_0)^2}$ has everything we need to control the derivative at t_1. The sum of $\hat{g}_a$ and $\hat{h}_b$ is a degree at most three polynomial with zeroes at t_0 and t_1 and easily specified derivatives at t_0 and t_1. Finally, a polynomial p of the form

$$\begin{aligned} p(t) &= \frac{t-t_1}{t_0-t_1}y_0 + \frac{t-t_0}{t_1-t_0}y_1 + g_a(t) + h_b(t) \\ & \frac{t-t_1}{t_0-t_1}y_0 + \frac{t-t_0}{t_1-t_0}y_1 + a\frac{(t-t_0)(t-t_1)^2}{(t_0-t_1)^2} + b\frac{(t-t_0)^2(t-t_1)}{(t_1-t_0)^2} \end{aligned}$$

would be the Hermite polynomial we are after. The first two terms form the interpolating polynomial passing through the required points. The last two terms are zero at t_0 and t_1 so do not affect this interpolation. Moreover, the last two terms are chosen so that their derivatives are convenient at t_0 and t_1. The derivative of $\frac{(t-t_1)^2(t-t_0)}{(t_0-t_1)^2}$ is 1 at t_0 and 0 at t_1. The derivative of $\frac{(t-t_0)^2(t-t_1)}{(t_1-t_0)^2}$ is 0 at t_0 and 1 at t_1. These characteristics ensure simple values for a and b in terms of the specified derivatives. To find out exactly what they should be, it remains to force $\dot{p}(t_0) = \dot{y}_0$ and $\dot{p}(t_1) = \dot{y}_1$:

$$\dot{p}(x) = \frac{y_1-y_0}{t_1-t_0} + 2\frac{(t-t_1)(t-t_0)}{(t_0-t_1)^2}a + 2\frac{(t-t_0)(t-t_1)}{(t_1-t_0)^2}b + \frac{(t-t_1)^2}{(t_0-t_1)^2}a + \frac{(t-t_0)^2}{(t_1-t_0)^2}b$$

so

$$\dot{p}(t_0) = \frac{y_1-y_0}{t_1-t_0} + a$$

and

$$\dot{p}(t_1) = \frac{y_1-y_0}{t_1-t_0} + b.$$

Therefore, we need $c = \dot{y}_0 - \frac{y_1-y_0}{t_1-t_0}$ and $d = \dot{y}_1 - \frac{y_1-y_0}{t_1-t_0}$. The desired degree at most three Hermite (osculating) polynomial is

$$p(t) = \frac{t-t_1}{t_0-t_1}y_0 + \frac{t-t_0}{t_1-t_0}y_1 + \frac{(t-t_1)^2(t-t_0)}{(t_0-t_1)^2}(\dot{y}_0 - m) + \frac{(t-t_0)^2(t-t_1)}{(t_1-t_0)^2}(\dot{y}_1 - m) \tag{5.1.1}$$

where $m = \frac{y_1-y_0}{t_1-t_0}$.

This form of the Hermite cubic polynomial is convenient for humans. It is formulaic and requires very little computation to write down. We will call it the Human form of the Hermite cubic polynomial. A more computer-friendly form, which we will refer to as the Computer form of the Hermite cubic is obtained via divided differences. In general, for an osculating polynomial where the first k derivatives are specified at t_i, t_i and y_i must be repeated $k+1$ times in the divided differences table. Quotients that would otherwise be undefined as a result of the repetition are replaced by the specified derivatives, first derivatives for first divided differences, second derivatives for second divided differences, and so on.

For the cubic Hermite polynomial p passing through (t_0, y_0) and (t_1, y_1) with derivative $\dot{y}_0$ at t_0 and $\dot{y}_1$ at t_1, the table looks like so:

t_0	y_0	y_0'		
t_0	y_0			
t_1	y_1	y_1'		
t_1	y_1			

The four remaining entries are to be filled in by the usual divided difference method. Can you compute them in general (in terms of $t_0, t_1, y_0, y_1, \dot{y}_0, \dot{y}_1$)? Answers on page 165. Using the results, we write down the interpolating polynomial in two ways:

$$\begin{aligned} p(t) &= y_0 + [\dot{y}_0](t-t_0) + \left[\frac{y_1-y_0}{(t_1-t_0)^2} - \frac{\dot{y}_0}{t_1-t_0}\right](t-t_0)^2 \\ &\quad + \left[\frac{\dot{y}_1+\dot{y}_0}{(t_1-t_0)^2} - 2\frac{y_1-y_0}{(t_1-t_0)^3}\right](t-t_0)^2(t-t_1) \end{aligned}$$

and

$$\begin{aligned} p(t) &= y_1 + [\dot{y}_1](t-t_1) + \left[\frac{\dot{y}_1}{t_1-t_0} - \frac{y_1-y_0}{(t_1-t_0)^2}\right](t-t_1)^2 \\ &\quad + \left[\frac{\dot{y}_1+\dot{y}_0}{(t_1-t_0)^2} - 2\frac{y_1-y_0}{(t_1-t_0)^3}\right](t-t_1)^2(t-t_0). \end{aligned}$$

Just as we had for interpolating polynomials, we have two ways to find cubic Hermite osculating polynomials. One way is convenient for humans and the other for computers.

Bèzier Curves

Forcing $(x(0), y(0)) = (-1, 2)$, we need

$$\begin{aligned} x(0) &= a_x = -1 \\ y(0) &= a_y = 2. \end{aligned}$$

Forcing $(x(1), y(1)) = (5, -2)$, we need

$$\begin{aligned} x(1) &= a_x + b_x + c_x = -1 + b_x + c_x = 5 \\ y(1) &= a_y + b_y + c_y = 2 + b_y + c_y = -2 \end{aligned}$$

or

$$\begin{aligned} b_x + c_x &= 6 \\ b_y + c_y &= -4. \end{aligned}$$

Bèzier curves are parametric curves with parameter $t \in [0, 1]$ connecting two points. The simplest Bèzier curve is a straight line passing through the two points. For example, the simplest Bèzier curve from $(-1, 2)$ to $(5, -2)$ is given by the parametric linear functions

$$\begin{aligned} x(t) &= (1-t)(-1) + t(5) \\ y(t) &= (1-t)(2) + t(-2), \end{aligned}$$

which we choose to write down in Lagrange form. You can check that $x(0) = -1$, $x(1) = 5$, $y(0) = 2$, and $y(1) = -2$. In other words, x passes through $(0, -1)$ and $(1, 5)$ while y passes through $(0, 2)$ and $(1, -2)$. This parametrization is unique because x and y are interpolating polynomials.

One the other hand, if we allow x and y to be quadratic, there are infinitely many (parametric) pairs of functions connecting $(-1, 2)$ to $(5, -2)$ even if we require x and y to be interpolating polynomials and restrict the parameter t to the interval $[0, 1]$. That is not to say we do not have quadratic Bèzier curves, but rather that we need to specify more than just the two points to be connected. Allowing the parameter function to be quadratic, we have say

$$\begin{aligned} x(t) &= a_x + b_x t + c_x t^2 \\ y(t) &= a_y + b_y t + c_y t^2, \end{aligned}$$

giving six unknowns or undetermined coefficients, if you will. That leaves two conditions that may yet be imposed on the parameter functions.

Any particular quadratic Bèzier curve is prescribed by specifying a control point distinct from the two endpoints. The two linear Bèzier curves, one connecting $(-1, 2)$ to the control point and the other connecting the control point to $(5, -2)$, then determine the quadratic Bèzier curve. Suppose $\vec{B}_{1,0}(t)$ is the linear Bèzier curve from $(-1, 2)$ to the control point and $\vec{B}_{1,1}(t)$ is the linear Bèzier curve from the control point to $(5, -2)$. These two curves define a family of linear Bèzier curves, namely the set of linear Bèzier curves from $\vec{B}_{1,0}(t_0)$ to $\vec{B}_{1,1}(t_0)$, where $t_0 \in [0, 1]$. Letting $\vec{B}_{2,0,t_0}(t)$ be the linear Bèzier curve from $\vec{B}_{1,0}(t_0)$ to $\vec{B}_{1,1}(t_0)$, the point $\vec{B}_{2,0,t_0}(t_0)$ is on the quadratic Bèzier curve from $(-1, 2)$ to $(5, -2)$ via the given control point. The collection of all such points as t_0 varies from 0 to 1 is the quadratic Bèzier curve we are after. Different control points determine different quadratics. For example, if we have $(0, 4)$ as our control point, $\vec{B}_{1,0}$ is the linear Bèzier curve connecting $(-1, 2)$ to $(0, 4)$ and $\vec{B}_{1,1}$ is the linear Bèzier curve from $(0, 4)$ to $(5, -2)$:

$$\vec{B}_{1,0}(t) = \begin{pmatrix} (1-t)(-1) \\ (1-t)(2) + t(4) \end{pmatrix}$$

and

$$\vec{B}_{1,1}(t) = \begin{pmatrix} t(5) \\ (1-t)(4) + t(-2) \end{pmatrix}.$$

$\vec{B}_{2,0,t_0}$ is the linear Bèzier curve connecting $\vec{B}_{1,0}(t_0)$ to $\vec{B}_{1,1}(t_0)$. Therefore, $\vec{B}_{2,0,t_0}(t) = (1-t)\vec{B}_{1,0}(t_0) + t\vec{B}_{1,1}(t_0)$ or

$$\vec{B}_{2,0,t_0}(t) = (1-t) \begin{pmatrix} (1-t_0)(-1) \\ (1-t_0)(2) + t_0(4) \end{pmatrix} + t \begin{pmatrix} t_0(5) \\ (1-t_0)(4) + t_0(-2) \end{pmatrix}.$$

Then

$$\vec{B}_{2,0,t_0}(t_0) = (1-t_0)\left(\begin{array}{c} (1-t_0)(-1) \\ (1-t_0)(2)+t_0(4) \end{array}\right) + t_0\left(\begin{array}{c} t_0(5) \\ (1-t_0)(4)+t_0(-2) \end{array}\right).$$

Observe that $\vec{B}_{2,0,t_0}$ is quadratic as a function of t_0 and that $\vec{B}_{2,0,0}(0) = \begin{pmatrix}-1\\2\end{pmatrix}$ and $\vec{B}_{2,0,1}(1) = \begin{pmatrix}5\\-2\end{pmatrix}$.

But the notation $\vec{B}_{2,0,t_0}(t_0)$ is cumbersome and we are really interested in a parametrization of the quadratic anyway. Letting $\vec{B}_{2,0}(t) = \vec{B}_{2,0,t}(t)$, we get the quadratic Bèzier curve from $(-1,2)$ to $(5,-2)$ via control point $(0,4)$:

$$\vec{B}_{2,0}(t) = (1-t)\left(\begin{array}{c} (1-t)(-1) \\ (1-t)(2)+t(4) \end{array}\right) + t\left(\begin{array}{c} t(5) \\ (1-t)(4)+t(-2) \end{array}\right)$$

and we have cleaner notation.

With some algebra, the expression for $\vec{B}_{2,0}$ can be simplified, but leaving it unsimplified emphasizes whence it came. It is the result of nested linear interpolations. Higher order Bèzier curves are constructed by continued nesting. We now use this idea to define the Bèzier curve from $\vec{P}_0$ to $\vec{P}_n$ via control points $\vec{P}_1, \vec{P}_2, \ldots, \vec{P}_{n-1}$. Commonly, $\vec{P}_0$ and $\vec{P}_n$ are also considered control points and so this Bèzier curve is also referred to as the Bèzier curve with control points $\vec{P}_0, \vec{P}_1, \ldots, \vec{P}_n$. Such a Bèzier curve will have degree at most n.

We begin by defining the linear Bèzier curves

$$\vec{B}_{1,i}(t) = (1-t)\vec{P}_i + (t)\vec{P}_{i+1}, \qquad i = 0, 1, \ldots, n-1. \tag{5.1.2}$$

Note that $\vec{B}_{1,i}$ is the linear Bèzier curve from $\vec{P}_i$ to $\vec{P}_{i+1}$. Then

$$\vec{B}_{j,i}(t) = (1-t)\cdot\vec{B}_{j-1,i}(t) + (t)\cdot\vec{B}_{j-1,i+1}(t), \qquad j = 2, 3, \ldots, n;\ i = 0, 1, \ldots, n-j. \tag{5.1.3}$$

Note that $\vec{B}_{2,i}(t)$ is the quadratic Bèzier curve connecting $\vec{P}_i$ to $\vec{P}_{i+2}$ via control point $\vec{P}_{i+1}$. With a little algebra, you can confirm that $\vec{B}_{3,i}(t)$ is at-most-cubic and connects $\vec{P}_i$ to $\vec{P}_{i+3}$. An inductive proof will show that $\vec{B}_{j,i}(t)$ is an at-most-degree-j polynomial parametrization connecting $\vec{P}_i$ to $\vec{P_{i+j}}$. Can you provide it? Answer on page 5.1. It follows that $\vec{B}_{n,0}(t)$ is the degree at most n Bèzier curve connecting $\vec{P}_0$ to $\vec{P}_n$.

Returning to our previous example, we add the control point $(5,1)$ so we have now four control points:

$$\vec{P}_0 = \begin{pmatrix}-1\\2\end{pmatrix},\ \vec{P}_1 = \begin{pmatrix}0\\4\end{pmatrix},\ \vec{P}_2 = \begin{pmatrix}5\\1\end{pmatrix},\ \vec{P}_3 = \begin{pmatrix}5\\-2\end{pmatrix}.$$

By equation 5.1.2,

$$\begin{aligned}
\vec{B}_{1,0}(t) &= (1-t)\vec{P}_0 + (t)\vec{P}_1 = (1-t)\begin{pmatrix}-1\\2\end{pmatrix} + t\begin{pmatrix}0\\4\end{pmatrix} = \begin{pmatrix}-1+t\\2+2t\end{pmatrix}\\
\vec{B}_{1,1}(t) &= (1-t)\vec{P}_1 + (t)\vec{P}_2 = (1-t)\begin{pmatrix}0\\4\end{pmatrix} + t\begin{pmatrix}5\\1\end{pmatrix} = \begin{pmatrix}5t\\4-3t\end{pmatrix}\\
\vec{B}_{1,2}(t) &= (1-t)\vec{P}_2 + (t)\vec{P}_3 = (1-t)\begin{pmatrix}5\\1\end{pmatrix} + t\begin{pmatrix}5\\-2\end{pmatrix} = \begin{pmatrix}5\\1-3t\end{pmatrix}.
\end{aligned}$$

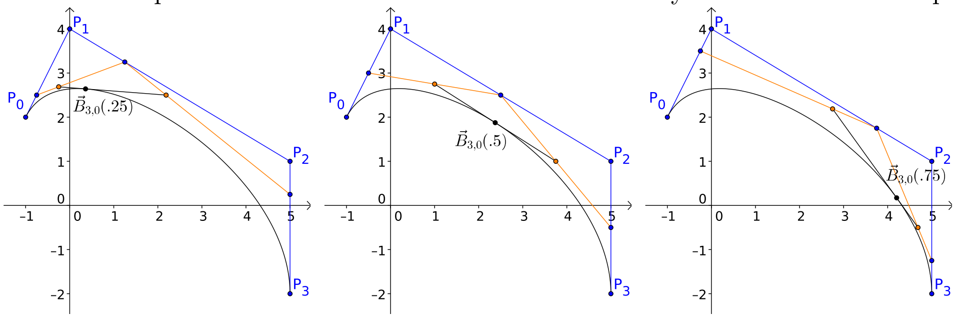
And by equation 5.1.3,

$$\begin{aligned}
\vec{B}_{2,0}(t) &= (1-t)\vec{B}_{1,0}(t) + (t)\vec{B}_{1,1}(t) = (1-t)\begin{pmatrix}-1+t\\2+2t\end{pmatrix} + t\begin{pmatrix}5t\\4-3t\end{pmatrix} = \begin{pmatrix}-1+2t+4t^2\\2+4t-5t^2\end{pmatrix}\\
\vec{B}_{2,1}(t) &= (1-t)\vec{B}_{1,1}(t) + (t)\vec{B}_{1,2}(t) = (1-t)\begin{pmatrix}5t\\4-3t\end{pmatrix} + t\begin{pmatrix}5\\1-3t\end{pmatrix} = \begin{pmatrix}10t-5t^2\\4-6t\end{pmatrix},
\end{aligned}$$

and

$$\begin{aligned}
\vec{B}_{3,0}(t) &= (1-t)\vec{B}_{2,0}(t) + (t)\vec{B}_{2,1}(t)\\
&= (1-t)\begin{pmatrix}-1+2t+4t^2\\2+4t-5t^2\end{pmatrix} + t\begin{pmatrix}10t-5t^2\\4-6t\end{pmatrix}\\
&= \begin{pmatrix}-1+3t+12t^2-9t^3\\2+6t-15t^2+5t^3\end{pmatrix}.
\end{aligned} \tag{5.1.4}$$

Figure 5.1.1: Three points on a cubic Bèzier curve constructed by recursive linear interpolation.



$\vec{B}_{3,0}(t)$ is the cubic Bèzier curve from $\begin{pmatrix}-1\\2\end{pmatrix}$ to $\begin{pmatrix}5\\-2\end{pmatrix}$ via control points $\begin{pmatrix}0\\4\end{pmatrix}$ and $\begin{pmatrix}5\\1\end{pmatrix}$. Figure 5.1.1 shows this Bèzier curve and the construction of three of its points via recursive linear interpolation. The blue points lie along the linear Bèzier curves $\vec{B}_{1,0}, \vec{B}_{1,1}, \vec{B}_{1,2}$. The orange points lie along the quadratic Bèzier curves $\vec{B}_{2,0}$ and $\vec{B}_{2,1}$. The black points lie along the cubic Bèzier curve. The graphs of the quadratics have been suppressed to avoid overcomplicating the figure.

Figure 5.1.1 may help you grasp the recursion, but maybe more importantly, may help you understand the relationship between the control points and the Bèzier curve. For example, upon close examination, you may be led to believe the line segments $\vec{B}_{1,0}$ and $\vec{B}_{1,2}$ are tangent to the cubic Bèzier curve $\vec{B}_{3,0}$ at $\vec{P}_0$ and $\vec{P}_3$, respectively. Close examination of the formulas will confirm it.

According to formulas 5.1.2 and 5.1.3, the (at most) cubic Bèzier curve with control points $\vec{P}_0, \vec{P}_1, \vec{P}_2, \vec{P}_3$ is computed thus:

$$\begin{aligned}\vec{B}_{1,0}(t) &= (1-t)\vec{P}_0 + (t)\vec{P}_1\\ \vec{B}_{1,1}(t) &= (1-t)\vec{P}_1 + (t)\vec{P}_2\\ \vec{B}_{1,2}(t) &= (1-t)\vec{P}_2 + (t)\vec{P}_3\end{aligned}$$

so

$$\begin{aligned}\vec{B}_{2,0}(t) &= (1-t)\vec{B}_{1,0}(t) + (t)\vec{B}_{1,1}(t) = (1-t)\left[(1-t)\vec{P}_0 + (t)\vec{P}_1\right] + t\left[(1-t)\vec{P}_1 + (t)\vec{P}_2\right]\\ &= (1-t)^2\vec{P}_0 + 2t(1-t)\vec{P}_1 + t^2\vec{P}_2\\ \vec{B}_{2,1}(t) &= (1-t)\vec{B}_{1,1}(t) + (t)\vec{B}_{1,2}(t) = (1-t)\left[(1-t)\vec{P}_1 + (t)\vec{P}_2\right] + t\left[(1-t)\vec{P}_2 + (t)\vec{P}_3\right]\\ &= (1-t)^2\vec{P}_1 + 2t(1-t)\vec{P}_2 + t^2\vec{P}_3\end{aligned}$$

so

$$\begin{aligned}\vec{B}_{3,0}(t) &= (1-t)\vec{B}_{2,0}(t) + (t)\vec{B}_{2,1}(t)\\ &= (1-t)\left[(1-t)^2\vec{P}_0 + 2t(1-t)\vec{P}_1 + t^2\vec{P}_2\right] + t\left[(1-t)^2\vec{P}_1 + 2t(1-t)\vec{P}_2 + t^2\vec{P}_3\right] \qquad (5.1.5)\\ &= (1-t)^3\vec{P}_0 + 3t(1-t)^2\vec{P}_1 + 3t^2(1-t)\vec{P}_2 + t^3\vec{P}_3.\end{aligned}$$

Hence, $\frac{d}{dt}\vec{B}_{3,0}(t) = -3(1-t)^2\vec{P}_0 + 3\left[(1-t)^2 - 2t(1-t)\right]\vec{P}_1 + 3\left[2t(1-t) - t^2\right]\vec{P}_2 + 3t^2\vec{P}_3$, from which it follows

$$\begin{aligned}\left.\frac{d}{dt}\vec{B}_{3,0}(t)\right|_{t=0} &= -3\vec{P}_0 + 3\vec{P}_1 = 3(\vec{P}_1 - \vec{P}_0)\\ \left.\frac{d}{dt}\vec{B}_{3,0}(t)\right|_{t=1} &= -3\vec{P}_2 + 3\vec{P}_3 = 3(\vec{P}_3 - \vec{P}_2).\end{aligned}$$

Indeed, the derivative of $\vec{B}_{3,0}$ at 0 is in the direction of the line segment from $\vec{P}_1$ to $\vec{P}_2$, and the derivative of $\vec{B}_{3,0}$ at 1 is in the direction of the line segment from $\vec{P}_2$ to $\vec{P}_3$. Moreover, these derivatives have magnitude exactly three times the magnitudes of the line segments.

Though we took a somewhat circuitous route, we now see another way to compute cubic Bèzier curves besides using recursion 5.1.2/5.1.3 or formula 5.1.5. Control points $\vec{P}_0$ and $\vec{P}_3$ give us two points x and y must pass through. Control points $\vec{P}_1$ and $\vec{P}_2$ give us $\dot{x}$ and $\dot{y}$ at those two points. Thus specified, x and y are cubic Hermite polynomials!

To be precise, let $\vec{P}_i = (x_i, y_i)$ for $i = 0, 1, 2, 3$. Then $x(t)$ is the cubic Hermite polynomial with $x(0) = x_0$, $\dot{x}(0) = 3(x_1 - x_0)$, $x(1) = x_3$, and $\dot{x}(1) = 3(x_3 - x_2)$; and $y(t)$ is the cubic Hermite polynomial with $y(0) = y_0$, $\dot{y}(0) = 3(y_1 - y_0)$, $y(1) = y_3$, and $\dot{y}(1) = 3(y_3 - y_2)$.

We close this section by computing the Bèzier curve from $\begin{pmatrix} -1 \\ 2 \end{pmatrix}$ to $\begin{pmatrix} 5 \\ -2 \end{pmatrix}$ via control points $\begin{pmatrix} 0 \\ 4 \end{pmatrix}$ and $\begin{pmatrix} 5 \\ 1 \end{pmatrix}$ using equation 5.1.1 and comparing our results to 5.1.4. With $x(0) = -1$, $\dot{x}(0) = 3$, $x(1) = 5$, and $\dot{x}(1) = 0$ (and the understood substitution of x for y), equation 5.1.1 gives $m = \frac{5+1}{1-0} = 6$ and

$$x(t) = \frac{t-1}{-1}(-1) + \frac{t}{1}(5) + \frac{(t-1)^2 t}{1}(3-6) + \frac{t^2(t-1)}{1}(-6)\,.$$

Using equation 5.1.1 with $y(0) = 2$, $\dot{y}(0) = 6$, $y(1) = -2$, and $\dot{y}(1) = -9$ gives $m = \frac{-2-2}{1-0} = -4$ and

$$y(t) = \frac{t-1}{-1}(2) + \frac{t}{1}(-2) + \frac{(t-1)^2 t}{1}(6+4) + \frac{t^2(t-1)}{1}(-9+4)\,.$$

While these equations are complete and correct, it is difficult to compare them to 5.1.4 without some simplification. Can you show

$$\begin{aligned} x(t) &= -1 + 3t + 12t^2 - 9t^3 \\ y(t) &= 2 + 6t - 15t^2 + 5t^3 \end{aligned}$$

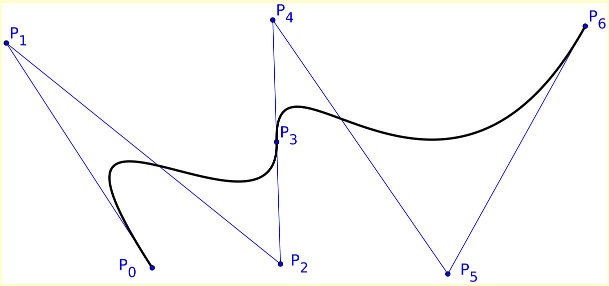
as required? Answer on page 165.

Crumpet 29: Bézier curves and CAGD

Bézier curves were originally developed around 1960 by employees at french automobile manufacturing companies. Paul de Casteljau of Citroën was first, but Pierre Bèzier of Renault popularized the method so has his name associated with the polynomials.

Nowadays, almost all computer aided graphic design, or CAGD, software uses Bèzier curves, particularly cubic, for drawing smooth objects. CAGD software with cubic Bèzier tools will display the four control points and allow the user to move them about. In fact, the software will draw the two linear Bézier curves at the endpoints as well. This gives the user "handles" to manipulate the curve. Some software will include the third linear Bèzier curve as well. The three linear Bèzier curves together form the so-called control polygon. Since the relationship between the control points and the curve is intuitive, manipulation of the control points, whether it be by handles or control polygons, provides a means for swift modeling of smooth shapes.

Some shapes are too intricate to model with a single cubic Bèzier curve, however. To handle such shapes, CAGD software allows a user to string cubic Bèzier curves together end to end, forming a composite, or piecewise, Bèzier curve, such as that shown here.



This particular curve is made of two cubic Bèzier curves, one with control points $\vec{P}_0, \vec{P}_1, \vec{P}_2, \vec{P}_3$ and the other with control points $\vec{P}_3, \vec{P}_4, \vec{P}_5, \vec{P}_6$. Since Bèzier curves are intended to model smooth objects, software will provide the option of forcing derivative matching at a common point such as $\vec{P}_3$. This is done by making sure the common point is on the line segment between its two adjacent control points ($\vec{P}_2$ and $\vec{P}_4$ in this diagram). You may view an interactive version of this diagram at the companion website.

Free open source software such as Inkscape, LibreOffice Drawing, and Dia provide Bezier curve drawing tools, but not all of them use the technical term. Inkscape has a Bezier curve tool by that name, but LibreOffice Drawing's Bezier curve tool is simply called "curve", and Dia's tool for single Bezier curves only, not composite, goes by the name of "Bezierline".

References [1, 10, 9, 15, 27, 32]

Key Concepts

osculating polynomial: A polynomial whose graph is required to pass through a set of prescribed points

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

and whose first m_i derivatives may also be specified at x_i.

Hermite polynomial: An osculating polynomial required to pass through two points with its first derivative specified at each point.

Bèzier curve: A curve connecting two points via parametric osculating polynomials.

Exercises

1. Find the cubic Hermite polynomial interpolating the data.

x	$f(x)$	$f'(x)$
1	2	1
5	3	-1

2. Find the Hermite polynomial of degree (at most) 5 interpolating the data.

x	$f(x)$	$f'(x)$
0	2	1
0.5	2	0
1	2	1

3. Let $g(x) = (\sqrt{2})^x$.

   (a) Using $x_0 = 1$ and $x_1 = 2$, find a Hermite interpolating polynomial for g.

   (b) Use the Hermite polynomial to approximate $g(1.5)$.

   (c) Calculate the actual error of this approximation, and compare it to the error you got in question 15 of section 3.2 on page 98.

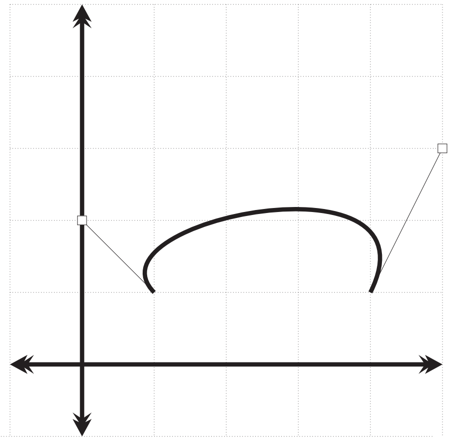
   (d) Which polynomial approximated $g(1.5)$ with smaller absolute error, the Hermite or the Lagrange interpolating polynomial?

4. Find a polynomial that passes through the points $(0, 0)$ and $(4, -3)$ and whose derivative passes through the points $(0, 1)$ and $(4, 1)$.

5. Construct the Hermite interpolating polynomial for the given data. Do this using a pencil, paper and calculator, or a spreadsheet. Do not use code.

x	$f(x)$	$f'(x)$
0.1	-0.29004996	-2.8019975
0.2	-0.56079734	-2.6159201
0.3	-0.81401972	-2.4533949

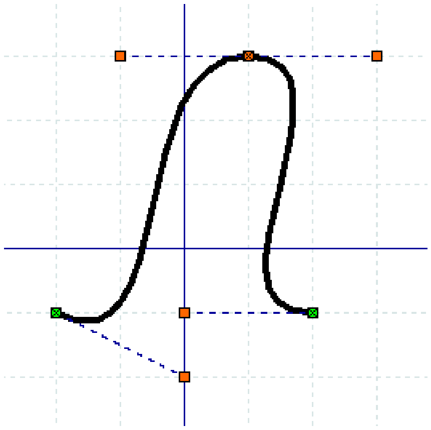
6. Find parametric equations for the cubic Bèzier curve. The ends of the "handles" are the four control points.



7. Write down the parametric equations of the Bèzier curve with control points $(-1, 2)$, $(-3, 2)$, $(3, 1)$, and $(3, 0)$. It is not necessary to simplify your answer.

8. Construct the parametric equations for the Bèzier curve with control points $(1, 1), (2, 1.5), (7, 1.5), (6, 2)$.

9. Find equations for the cubic polynomials that make up the composite Bézier curve.



10. The data in question 5 were generated using $f(x) = x^2 \cos(x) - 3x$.

(a) Approximate $f(0.18)$ using the polynomial from question 5.

(b) Calculate the absolute error of this approximation.

11. Suppose $H(x) = x^5 - 3x^4 + 2x^3 - 6x + 2$ is a Hermite polynomial interpolating the data

x	$f(x)$	$f'(x)$
0	2	-6
1	-4	
2	-10	2

collected from a function f. Find the missing datum.

12. A Hermite polynomial $H(x)$ is constructed using the data

x	0.3	0.5	0.6	0.8
$f(x)$	0.8	0.6	0.3	0.5
$f'(x)$	1.5	-1.2	-5.3	-2

(a) Find $(H \circ H)'(0.6)$. That is, the derivative of $H(H(x))$ evaluated at $x = 0.6$.

(b) Find $(f \circ f)'(0.8)$.

13. The Hermite interpolating polynomial for the following data has the form $H(x) = a_0 + a_1(x - 0.3) + a_2(x - 0.3)^2 + \ldots$.

x	$f(x)$	$f'(x)$
0.30	0.295	-0.155
0.32	0.314	-0.149
0.35	0.342	-0.139

(a) Fill in the missing part of the form of $H(x)$.

(b) What is the maximum possible degree of $H(x)$?

(c) Find a_0 and a_1.

14. Construct the divided differences table that led to the Hermite polynomial

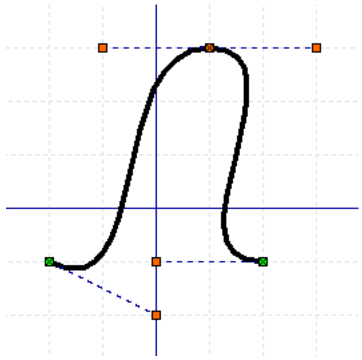
$$p(x) = 2 - (x - 1) + \frac{1}{4}(x - 1)^2 + \frac{1}{4}(x - 1)^2(x - 3).$$

15. The Bèzier Curve

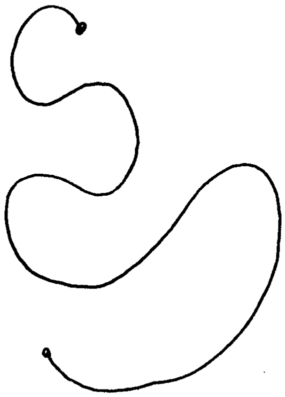
$$\begin{aligned} x(t) &= 11t^3 - 18t^2 + 3t + 5 \\ y(t) &= t^3 + 1 \end{aligned}$$

has control points $(5, 1)$, $(6, 1)$, and $(1, 2)$. Find the fourth control point.

16. What is the minimum number of cubic Bèzier curves in the diagram, and why?



17. Refer to the following graph.



(a) The graph can not be the graph of a single cubic Bèzier curve. Why not?

(b) The graph is that of a composite cubic Bèzier curve. At least how many cubic Bèzier curves have been spliced together, and why?

18. Give three reasons that might make you use a Bèzier curve rather than a Lagrange polynomial to model a certain graph.

19. The osculating polynomial $p(x)$ passing through $(x_0, f(x_0))$ with $P'(x_0) = f'(x_0)$, $P''(x_0) = f''(x_0)$, and $P'''(x_0) = f'''(x_0)$ is also called what? Be as specific as you can.

20. A cubic polar Bèzier curve is the unique (parametrized) cubic polar function $(r(t), \theta(t))$ satisfying the following data.

t	$r(t)$	$\theta(t)$	$\dot{r}(t)$	$\dot{\theta}(t)$
0	r_0	θ_0	δ_0	μ_0
1	r_1	θ_1	δ_1	μ_1

(a) A standard cubic Bèzier curve is given by the control points $(0, 0)$, $(2, 0)$, $(0, 1)$, and $(0, 3)$ (in that order). Convert this data into polar coordinate data. Recall that the conversion from Cartesian coordinates to polar coordinates involves the formulas

$$r = \sqrt{x^2 + y^2} \quad \text{and} \quad \tan\theta = \frac{y}{x}.$$

(b) Find the cubic polar Bèzier curve based on your results from (a).

21. Write a function to compute Hermite polynomials.

22. A car traveling along a straight road is clocked at a number of points. The data from the observations are given in the following table, where the time is in seconds, the distance is in feet, and the speed is in feet per second.

Time	0	3	5	8	13
Distance	0	225	383	623	993
Speed	75	77	80	74	72

(a) Compute a Hermite interpolating polynomial for the data.

(b) Use your polynomial from part (a) to predict the position (distance) of the car and its speed when $t = 10$ seconds.

(c) Determine whether the car ever exceeds the 55 mph speed limit on the road. If so, what is the first time the car exceeds this speed?

(d) What is the predicted maximum speed for the car?

NOTES: Speed is the derivative of distance.

$$\begin{aligned} 55\frac{\text{miles}}{\text{hour}} &= 55\frac{\text{miles}}{\text{hour}} \times \frac{5280 \text{ feet}}{\text{mile}} \times \frac{1 \text{ hour}}{3600 \text{ seconds}} \\ &\approx 80.67\frac{\text{feet}}{\text{second}} \end{aligned}$$

Answers

Hermite polynomial computer form: The four remaining entries are

$$\begin{aligned} f_{1,1} &= \frac{y_1 - y_0}{t_1 - t_0} \\ f_{0,2} &= \frac{f_{1,1} - \dot{y}_0}{t_1 - t_0} = \frac{y_1 - y_0}{(t_1 - t_0)^2} - \frac{\dot{y}_0}{t_1 - t_0} \\ f_{1,2} &= \frac{\dot{y}_1 - f_{1,1}}{t_1 - t_0} = \frac{\dot{y}_1}{t_1 - t_0} - \frac{y_1 - y_0}{(t_1 - t_0)^2} \\ f_{0,3} &= \frac{f_{1,2} - f_{0,2}}{t_1 - t_0} = \frac{\dot{y}_1 + \dot{y}_0}{(t_1 - t_0)^2} - 2\frac{y_1 - y_0}{(t_1 - t_0)^3} \end{aligned}$$

Bezier curve $\vec{B}_{j,i}(t)$ is an at-most-degree-j polynomial connecting $\vec{P_i}$ to $\vec{P_{i+j}}$: *Proof.* We proceed by induction on j, beginning with $j = 1$: Since

$$\vec{B}_{1,i}(t) = (1-t)\vec{P_i} + (t)\vec{P}_{i+1}, \qquad i = 0, 1, \ldots, n-1,$$

$\vec{B}_{1,i}(0) = \vec{P_i}$ and $B_{1,i}(1) = \vec{P}_{i+1}$ so $\vec{B}_{1,i}$ connects $\vec{P_i}$ to $\vec{P}_{i+1}$. Furthermore, $\vec{B}_{1,i}(t) = \vec{P_i} + t(\vec{P}_{i+1} - \vec{P_i})$, so $\vec{B}_{1,i}$ is an at-most-degree-1 polynomial. Now assume $\vec{B}_{j,i}(t)$ is an at-most-degree-j polynomial connecting $\vec{P_i}$ to $\vec{P_{i+j}}$ for some $j \geq 1$ (and all applicable i). By definition, $\vec{B}_{j+1,i}(0) = \vec{B}_{j,i}(0)$ and $\vec{B}_{j+1,i}(1) = \vec{B}_{j,i+1}(1)$. By the inductive hypothesis, $\vec{B}_{j,i}(0) = \vec{P_i}$ and $\vec{B}_{j,i+1}(1) = \vec{P}_{i+j+1}$, so $\vec{B}_{j+1,i}$ connects $\vec{P_i}$ to $\vec{P}_{i+j+1}$. Furthermore,

$$\vec{B}_{j+1,i}(t) = (1-t) \cdot \vec{B}_{j,i}(t) + (t) \cdot \vec{B}_{j,i+1}(t)$$

has degree at most $j+1$ because $\vec{B}_{j,i}(t)$ and $\vec{B}_{j,i+1}(t)$ have at most degree j (by the inductive hypothesis). This completes the proof. □

Bézier curve via Hermite cubics: The simplification may be done as follows.

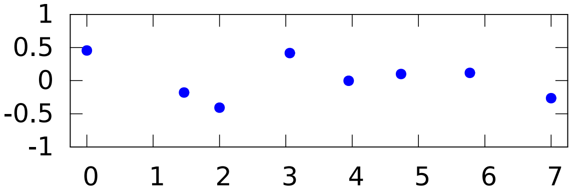
$$\begin{aligned} x(t) &= \frac{t-1}{-1}(-1) + \frac{t}{1}(5) + \frac{(t-1)^2 t}{1}(3-6) + \frac{t^2(t-1)}{1}(-6) \\ &= (t-1) + 5t - 3t(t-1)^2 - 6t^2(t-1) \\ &= 6t - 1 - 3t(t^2 - 2t + 1) - 6t^3 + 6t^2 \\ &= 6t - 1 - 3t^3 + 6t^2 - 3t - 6t^3 + 6t^2 \\ &= -9t^3 + 12t^2 + 3t - 1 \end{aligned}$$

and

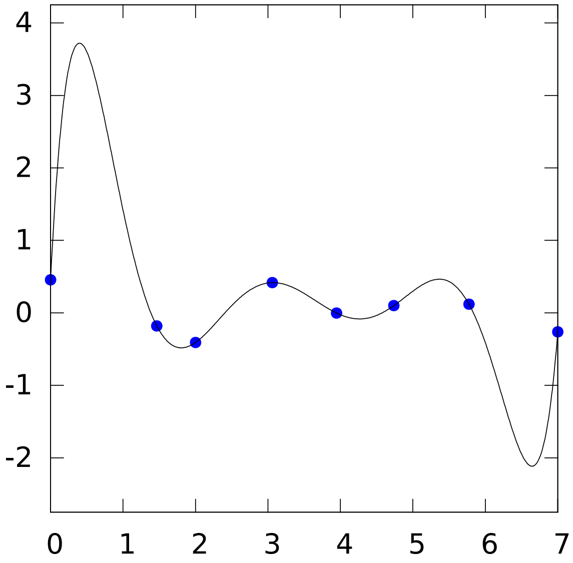
$$\begin{aligned} y(t) &= \frac{t-1}{-1}(2) + \frac{t}{1}(-2) + \frac{(t-1)^2 t}{1}(6+4) + \frac{t^2(t-1)}{1}(-9+4) \\ &= -2(t-1) - 2t + 10t(t-1)^2 - 5t^2(t-1) \\ &= -2t + 2 - 2t + 10t(t^2 - 2t + 1) - 5t^3 + 5t^2 \\ &= 2 - 4t + 10t^3 - 20t^2 + 10t - 5t^3 + 5t^2 \\ &= 5t^3 - 15t^2 + 6t + 2. \end{aligned}$$

5.2 Splines

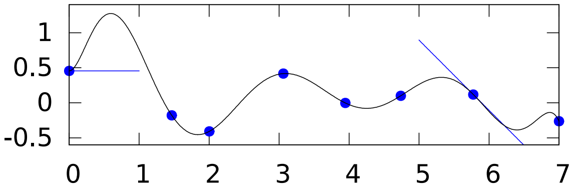
Osculating polynomials have limited use in applications where a curve is required to pass through a large number of points. And large may mean only *a half dozen* or so. Take the following innocuous-looking set of points.



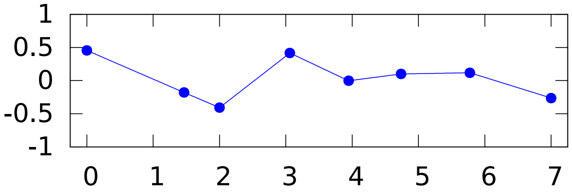
It is easy to imagine an equally innocuous function passing through these eight points, but actually finding such a function poses a slight challenge. The interpolating polynomial of least degree oscillates too widely.



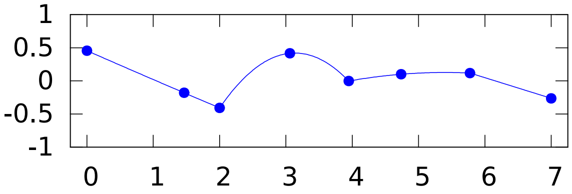
This is a common problem with high-degree interpolating polynomials. There is no control over their oscillations, and the oscillations are most often undesirable. The oscillations can be tamed to some degree by finding the osculating polynomial through these points with, say, a first derivative of 0 at 0 and of $-\frac{1}{2}$ at the seventh point from the left (the one whose x-coordinate is between 5 and 6).



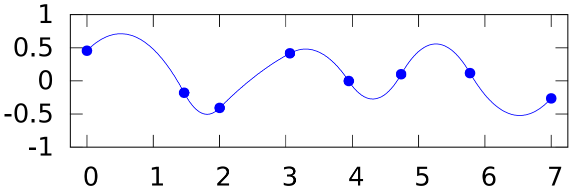
That's better, but still leaves something to be desired. And the business of setting the first derivatives at two of the points strictly for the purpose of reducing the oscillations is a bit arbitrary—better to let the nature of the problem dictate. The oscillations of the previous attempts make them far too distinctive and interesting for the vapid set of points with which we began. A rightfully trite way to interpolate the data is by connecting consecutive points by line segments.



This forms what is known as the piecewise linear interpolation of the data set. This type of graph is often seen in public media. Many applications, especially those from engineering, require some smoothness, however. Connecting sets of three consecutive points by quadratic functions helps.



That takes care of smoothness at three of the points, but still lacks differentiability at the points common to consecutive quadratics. Moreover, using the first three points for the first quadratic (which looks linear to the naked eye), the third through fifth points for the second quadratic, and the fifth through seventh points for the third quadratic (which also looks linear to the naked eye) leaves only the seventh and eighth points for what would presumably be a fourth quadratic. With only two points, however, a line segment is used instead. A smoother solution to the problem is to make sure the first derivatives of consecutive quadratics match at their common point. With that in mind, it makes sense to fit only two points per parabola, leaving one coefficient (of the three in any quadratic) for matching the derivative of the neighboring quadratic.



That's better! This piecewise parabolic function has continuous first derivative, but there is still something arbitrary about it. The seven parabolas have, all together, 21 coefficients. Making each parabola pass through two points gives 14 conditions on those coefficients. Having adjacent parabolas match first derivatives at their common points gives 6 more conditions, one at each of the 6 interior points. That leaves one "free" coefficient. Specifying one last condition seems a bit arbitrary, and is. The graph shows the result when the derivative at 0 is set to 1. Notice there is no control over the derivative at the right end. Besides the arbitrariness, this asymmetry is bothersome. If only we had one more degree of freedom...

Piecewise polynomials

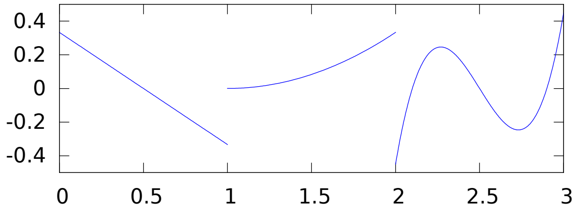
A piecewise-defined function whose pieces are all polynomials is called a piecewise polynomial. It takes the form

$$p(x) = \begin{cases} p_1(x), & x \in [x_0, x_1] \\ p_2(x), & x \in (x_1, x_2] \\ & \vdots \\ p_n(x) & x \in (x_{n-1}, x_n] \end{cases}$$

where $p_i(x)$ is a polynomial for each $i = 1, 2, \ldots, n$ and $x_0 < x_1 < \cdots < x_n$; or some variant where $p(x_j)$ is defined by exactly one of the p_i. If each p_i is a linear function, p is called piecewise linear. If each p_i is a quadratic function, p is called piecewise quadratic. If each p_i is a cubic function, p is called piecewise cubic. And so on. Examples of piecewise linear and piecewise quadratic functions appear in the introduction to this section.

Splines

Nothing about the definition of piecewise polynomials requires one to be differentiable or even continuous. The following function is a piecewise polynomial.



Most applications of piecewise polynomials require continuity or differentiability, however. Any piecewise polynomial with at least one continuous derivative is called a spline. The points separating adjacent pieces, the x_j, $j = 1, 2, \ldots, n-1$, are called knots or joints.

The last graph in the introduction to this section shows a quadratic spline. Each piece of the piecewise function is a quadratic, and the quadratics are chosen so that their derivatives match at the joints. As pointed out there, though, we needed to supply one unnatural condition—the derivative at the left endpoint. It could have been the derivative at any of the points, or even the second derivative at one of the points. In a very real sense, the choice was arbitrary. It was not governed naturally by the question at hand. Consequently, there is a family of solutions to the problem of connecting those eight points with a continuously differentiable piecewise quadratic.

Cubic splines

The most common spline in use is the cubic spline. As with the quadratic spline, a cubic spline is computed by matching derivatives at the joints. In fact, there are enough coefficients in the set of cubics that both first and second derivatives are matched. Note that, according to our definition of spline, matching both first and second derivatives at the joints is not strictly necessary, however. Other sources will give a more restrictive definition of spline where matching both derivatives is required. As a matter of convention, we focus on such splines.

A cubic spline required to interpolate $n+1$ points has $n-1$ joints and n pieces. It follows that the set of cubics has $4n$ coefficients. Requiring each cubic to pass through 2 points gives $2n$ conditions on the coefficients. Requiring first derivative matching at the joints gives $n-1$ more conditions. Requiring second derivative matching at the joints gives an additional $n-1$ conditions for a grand total of $4n-2$ conditions. That leaves 2 “free” coefficients. Mathematically speaking, we have a family of splines with two degrees of freedom. To find any specific spline, we need to enforce two more conditions on the coefficients. These conditions may include the first, second, or third derivative at two of the nodes, both the first and second derivative at a single node, or some other combination of two derivative requirements.

Guided perhaps by knowledge of draftsman’s splines, convention leads us to supply endpoint conditions. That is, we require something of some derivative at x_0 and at x_n. Supplying the first derivative is akin to pointing the draftsmen’s spline in a particular direction at its ends. Setting the second derivative equal to 0 is akin to allowing the ends of a draftsman’s spline to freely point in whatever direction physics takes them. These models of draftsman’s splines are not particularly accurate, but they are motivational.

A cubic spline with its first derivative specified at both endpoints is called a clamped spline. A cubic spline with its second derivative set equal to zero at both endpoints is called a natural or free spline. A hybrid where the first derivative is specified at one end and the second derivative is set to zero at the other has no special name. To be precise, we have the following definitions.

Let $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ be $n+1$ points where $x_0 < x_1 < \cdots < x_n$ and let $S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$ for $i = 1, 2, \ldots, n$. Then S, defined by

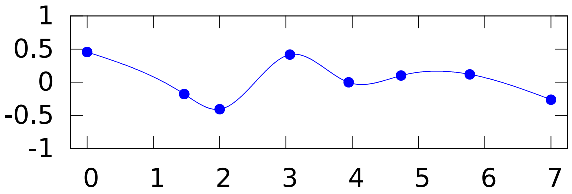
$$S(x) = \begin{cases} S_1(x), & x \in [x_0, x_1] \\ S_2(x), & x \in [x_1, x_2] \\ & \vdots \\ S_n(x), & x \in [x_{n-1}, x_n] \end{cases},$$

is a cubic spline if it satisfies the following three conditions.

1. $S_i(x_{i-1}) = y_{i-1}$ and $S_i(x_i) = y_i$ for $i = 1, 2, \ldots, n$ (interpolation)
2. $S_i'(x_i) = S_{i+1}'(x_i)$ and $S_i''(x_i) = S_{i+1}''(x_i)$ for $i = 1, 2, \ldots, n-1$ (derivative matching)
3. One of the following is satisfied (endpoint conditions)
   (a) $S_1''(x_0) = S_n''(x_n) = 0$
   (b) $S_1'(x_0) = m_0$ and $S_n'(x_n) = m_n$ for some m_0 and m_n
   (c) $S_1'(x_0) = m_0$ for some m_0 and $S_n''(x_n) = 0$
   (d) $S_1''(x_0) = 0$ and $S_n'(x_n) = m_n$ for some m_n

If endpoint condition 3a is satisfied, S is called a free spline or natural spline. If endpoint condition 3b is satisfied, S is called a clamped spline.

The natural (cubic) spline passing through the eight points presented in the introduction to this section looks like this.



Finally, a function that is as unspectacular as the data set itself! How was it calculated, you ask? The short answer is, the 28 simultaneous equations resulting from the definition of natural cubic spline were solved. The solution provided the coefficients a_i, b_i, c_i, d_i, $i = 1, 2, \dots, 7$.

Setting up the equations

The long answer is, well, a bit longer to tell, but really only differs from the short version in the level of detail. To begin, the requirement that $S_i(x_i) = y_i$ immediately gives us the values of n of the coefficients:

$$S_i(x_i) = a_i = y_i.$$

The requirement that $S_i(x_{i-1}) = y_{i-1}$ gives us the n equations

$$S_i(x_{i-1}) = y_i + b_i(x_{i-1} - x_i) + c_i(x_{i-1} - x_i)^2 + d_i(x_{i-1} - x_i)^3 = y_{i-1} \tag{5.2.1}$$

for $i = 1, 2, \dots, n$. The derivative requirements give us $n - 1$ equations each:

$$S_i'(x_i) = S_{i+1}'(x_i) \quad \Rightarrow \quad b_i = b_{i+1} + 2c_{i+1}(x_i - x_{i+1}) + 3d_{i+1}(x_i - x_{i+1})^2 \tag{5.2.2}$$
$$S_i''(x_i) = S_{i+1}''(x_i) \quad \Rightarrow \quad 2c_i = 2c_{i+1} + 6d_{i+1}(x_i - x_{i+1}) \tag{5.2.3}$$

for $i = 1, 2, \dots, n - 1$. Finally, the endpoint conditions give us the two equations

$$S_1''(x_0) = 2c_1 + 6d_1(x_0 - x_1) = 0 \tag{5.2.4}$$
$$S_n''(x_n) = 2c_n = 0. \tag{5.2.5}$$

Without much ado, we have the values of the a_i and of c_n. The remaining $3n - 1$ coefficients are found by solving the remaining $3n - 1$ simultaneous equations. Though a computer can certainly handle the solution from here, finding a bit of the general solution by hand gives a much more efficient algorithm.

Solving the equations

Essentially, we now have three equations with three unknowns. Equations 5.2.1, 5.2.2, and 5.2.3 are written in the variables b_i, c_i, d_i. Equation 5.2.3 can easily be solved for d_i in terms of c_i and equation 5.2.1 can easily be solved for b_i. The resulting expressions can be substituted into equation 5.2.2 to get an equation in only c_i. It is a straightforward matter to complete the calculation. At this point, it becomes convenient to define $h_i = x_{i-1} - x_i$.

$$(5.2.3) \quad \Rightarrow \quad d_{i+1} = \frac{c_i - c_{i+1}}{3h_{i+1}}, \quad i = 1, 2, \dots, n-1$$
$$\Rightarrow \quad d_i = \frac{c_{i-1} - c_i}{3h_i}, \quad i = 2, 3, \dots, n. \tag{5.2.6}$$
$$(5.2.1) \quad \Rightarrow \quad b_i = \frac{y_{i-1} - y_i}{h_i} - c_i h_i - d_i h_i^2, \quad i = 1, 2, \dots, n$$
$$\Rightarrow \quad b_i = \frac{y_{i-1} - y_i}{h_i} - c_i h_i - \frac{(c_{i-1} - c_i)h_i}{3}, \quad i = 2, 3, \dots, n$$
$$\Rightarrow \quad b_i = \frac{y_{i-1} - y_i}{h_i} - \frac{(c_{i-1} + 2c_i)h_i}{3}, \quad i = 2, 3, \dots, n \tag{5.2.7}$$
$$\Rightarrow \quad b_{i+1} = \frac{y_i - y_{i+1}}{h_{i+1}} - \frac{(c_i + 2c_{i+1})h_{i+1}}{3}, \quad i = 1, 2, \dots, n-1.$$

Substituting into equation 5.2.2,

$$\frac{y_{i-1} - y_i}{h_i} - \frac{(c_{i-1} + 2c_i)h_i}{3} = \frac{y_i - y_{i+1}}{h_{i+1}} - \frac{(c_i + 2c_{i+1})h_{i+1}}{3} + 2c_{i+1}h_{i+1} + (c_i - c_{i+1})h_{i+1}$$

for $i = 2, 3, \ldots, n-1$. With a bit of simplification, this becomes

$$h_i c_{i-1} + 2(h_i + h_{i+1})c_i + h_{i+1}c_{i+1} = 3\left(\frac{y_{i-1} - y_i}{h_i} - \frac{y_i - y_{i+1}}{h_{i+1}}\right), \quad i = 2, 3, \ldots, n-1. \tag{5.2.8}$$

We now have $n-2$ equations in the n unknown c_i. These equations hold for any cubic spline with any endpoint conditions. But equation 5.2.2 has not been used with index $i = 1$. Hence, we still have to incorporate

$$b_1 = b_2 + 2c_2h_2 + 3d_2h_2^2 \tag{5.2.9}$$

into the solution. It remains to replace b_1, b_2, and d_2 by expressions in c_i.

To begin, equations 5.2.7 and 5.2.6 with $i = 2$ give

$$\begin{aligned} b_2 &= \frac{y_1 - y_2}{h_2} - \frac{(c_1 + 2c_2)h_2}{3} \\ d_2 &= \frac{c_1 - c_2}{3h_2}. \end{aligned}$$

Making the substitutions for b_2 and d_2, equation 5.2.9 becomes

$$\begin{aligned} b_1 &= \frac{y_1 - y_2}{h_2} - \frac{(c_1 + 2c_2)h_2}{3} + 2c_2h_2 + (c_1 - c_2)h_2 \\ &= \frac{y_1 - y_2}{h_2} + \frac{2}{3}h_2c_1 + \frac{1}{3}h_2c_2. \end{aligned} \tag{5.2.10}$$

We have not used the endpoint conditions yet, so this equation is good for any cubic spline. Whatever endpoint conditions are given must result in an expression for b_1 in terms of c_i plus one other equation in the c_i.

In the case of the free spline, endpoint condition 5.2.5 gives $c_n = 0$. This is the first of the final two equations. Endpoint condition 5.2.4 gives $d_1 = -\frac{c_1}{3h_1}$. This relationship is not directly useful since we are looking for an expression for b_1. However, equation 5.2.1 with $i = 1$ gives $b_1 = \frac{y_0 - y_1}{h_1} - c_1h_1 - d_1h_1^2$ so we can use it to find

$$b_1 = \frac{y_0 - y_1}{h_1} - \frac{2}{3}c_1h_1.$$

Finally, substituting into equation 5.2.10, the final equation in c_i is $\frac{y_0 - y_1}{h_1} - \frac{2}{3}c_1h_1 = \frac{y_1 - y_2}{h_2} + \frac{2}{3}h_2c_1 + \frac{1}{3}h_2c_2$, which simplifies to

$$2(h_1 + h_2)c_1 + h_2c_2 = 3\left(\frac{y_0 - y_1}{h_1} - \frac{y_1 - y_2}{h_2}\right). \tag{5.2.11}$$

Equations 5.2.8, 5.2.11, and $c_n = 0$ are n equations which can be solved for the n coefficients c_i. Back-substitution will give the values of the b_i and d_i.

Other endpoint conditions lead to a different pair of final equations, but the process is the same. We need to substitute an expression for b_1 into 5.2.10 and come up with one other equation.

An application of natural cubic splines?

"For many important applications, this mathematical [cubic spline] model of the draftsman's spline is highly realistic."[1] Claims such as this rely on the assumptions that a draftsman's spline is aptly modeled by a thin beam and that beam deflections are small. But the shapes modeled by splines often include large deflections, and unless the draftsman's spline is damaged in some way, its shape will be an infinitely differentiable curve. Cubic splines generally lack continuity in their third derivative, hence, do not have higher order derivatives. Moreover, the endpoint conditions $S_0''(x_0) = S_n''(x_n) = 0$ do not translate well to the physical situation. These conditions imply the shape of the spline has zero curvature (concavity) at the endpoints while nothing about the physical situation points to that conclusion.

Despite the cubic spline's ineffective use as a model for a draftsman's spline, it can be used with great efficacy in design applications. At Boeing, the airplane manufacturer, for example, they are used in computer-aided graphic design, computer-aided manufacturing, engineering analysis and simulation, and as a key component in Boeing's Automated Flight Manual system. By 2005, it was estimated that Boeing's use of splines involved about 500 million spline evaluations every day![2]

[1] Ahlberg and Nilson, The Theory of Splines and their Applications, Elsevier, 1967.

[2] SIAM News, volume 38, number 4, May 2005.

Exercises

1. What problem with polynomial interpolation does cubic spline interpolation address?

2. Write down the system of equations that would need to be solved in order to find the cubic spline through $(0,-9)$, $(1,-13)$, and $(2,-29)$ with free boundary conditions. Do not attempt to solve the system. [S]

3. Set up but do not solve the equations which could be solved to find the free cubic spline through the points $(1,1)$, $(2,3)$, and $(4,2)$.

4. List three reasons that might make you use a cubic spline rather than a Lagrange polynomial to model a certain graph.

5. Write down a system of equations that could be solved in order to find the free cubic spline through the following data points. Do not solve the system.

x	$f(x)$
0.1	-0.62
0.2	-0.28
0.3	0.0066
0.4	0.24

6. Write down the system of equations that would need to be solved in order to find the cubic spline through $(0,-9)$, $(1,-13)$, and $(2,-29)$ with clamped boundary conditions $S'(0) = 1$ and $S'(2) = -1$. Do not attempt to solve the system.

7. Set up but do not solve the equations which could be solved to find the clamped cubic spline through the points $(1,1)$, $(2,3)$, and $(4,2)$ with $S'(1) = S'(4) = 0$. [S]

8. Write down a system of equations that could be solved in order to find the clamped cubic spline through the following data points with $S'(0.1) = 0.5$ and $S'(0.4) = 0.1$. Do not solve the system.

x	$f(x)$
0.1	-0.62
0.2	-0.28
0.3	0.0066
0.4	0.24

9. Find the spline described in question

   (a) 2 [S]

   (b) 3

   (c) 5 [A]

   (d) 6

   (e) 7 [S]

   (f) 8 [A]

   (a) 9a [S]

   (b) 9b

   (c) 9c [A]

10. Modify the code presented in this section so that it computes the coefficients for a clamped cubic spline. [S]

11. Use your code from question 10 to check your answer to question

    (a) 9d

    (b) 9e [S]

    (c) 9f [A]

12. Modify the code presented in this section so that it computes the coefficients for a cubic spline with mixed endpoint conditions 3c (page 168).

13. Use your code from question 12 to find the cubic spline through $(0,-9)$, $(1,-13)$, and $(2,-29)$ with mixed boundary conditions $S'(0) = 1$ and $S''(2) = 0$.

14. Use your code from question 12 to find the cubic spline through the points $(1,1)$, $(2,3)$, and $(4,2)$ with $S'(1) = S''(4) = 0$.

15. Suppose $n+1$ points are given ($n > 1$). How many endpoint conditions are needed to fit the points with a

    (a) quadratic spline with first derivative matching at each joint?

    (b) cubic spline with first and second derivative matching at each joint?

    (c) quartic spline with first, second, and third derivative matching at each joint?

    (d) a degree k spline ($k > 1$) with derivative matching up to degree $k-1$ at each joint?

16. Suppose a spline S is to be fit to the four points (x_i, y_i), $i = 0,1,2,3$ where $x_0 < x_1 < x_2 < x_3$. Further suppose S is to be linear on $[x_0, x_1]$, quadratic on $[x_1, x_2]$, and cubic on $[x_2, x_3]$. Finally suppose S is to have one continuous derivative. How many endpoint conditions are needed to specify the spline uniquely? Argue that any such endpoint conditions must be specified at x_3 and not x_0.

17. Let $f(x) = \sin x$ and $x_0 = 0$, $x_1 = \pi/4$, $x_2 = \pi/2$, $x_3 = 3\pi/4$, and $x_4 = \pi$.

    (a) Find the cubic (clamped) spline through $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_4, f(x_4))$ with $S'(0) = f'(0)$ and $S'(\pi) = f'(\pi)$.

    (b) Approximate $f(\pi/3)$ by computing $S(\pi/3)$.

    (c) Approximate $f(7\pi/8)$ by computing $S(7\pi/8)$.

    (d) Calculate the absolute errors in the approximations.

Chapter 6

Ordinary Differential Equations

The gate and key to the sciences is mathematics.
–Roger Bacon (*Opus Majus)*

If I were again beginning my studies, I would follow the advice of
Plato and start with mathematics.
–Galileo Galilei

6.1 The Motion of a Pendulum

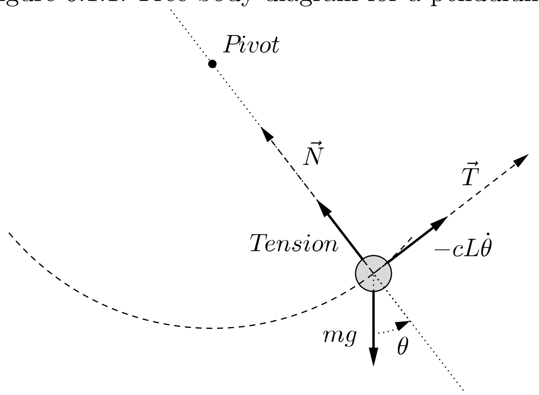
A brief history

Christiaan Huygens (1629-1695) is credited with inventing the pendulum clock in 1656, and Galileo Galilei (1564-1642) is credited with the first scientific study of the properties of pendula.[25, 33] In a famous letter to Guidobaldo del Monte in 1602, Galileo asserts that the period of a swinging pendulum (the time it takes to swing one way and back) is independent of the amplitude of the swing (how far it swings left and right). Del Monte famously argued that the physical evidence did not support the claim.[20] And he was right—it does not, and Galileo's claim is actually false. The period of a pendulum varies with the amplitude of its swing (all else equal).

Historians are generally willing to forgive Galileo for this error, though, likely due, in part, to the fact that the period of a pendulum is nearly constant for small amplitudes, and in part, to the fact that Galileo was the main figure in the scientific revolution (the birth of modern science) in the 17th century. His results regarding pendular motion account for only a small part of his total contribution to the sciences. The way he utilized idealized mathematical models of the physical world to inform his claims and experiments, a method of scientific study that directly contrasted with the generally held wisdom of his day, forms the basis for the scientific revolution, and as such was at least as important to science as any of his individual scientific discoveries. As for the pendulum, he put in motion the investigations which would one day (some years after his death) lead to a method of determining longitude at sea, an accomplishment that would change the world! With the ability to calculate their longitude, sailors were able to sail the seas, discover new places, and map the globe. Perhaps the biggest impact was the European colonization of foreign lands.

The thought of a pendulum today most likely brings to mind the grandfather clock. While arguably less important than its contribution to science and navigation, the timekeeping accuracy that pendulum clocks brought to the world had a substantial impact on broad society. With accurate timekeeping, time-based labor, transit and trade schedules, announced starting times for religious or other meetings, and every other clock-based phenomenon we take for granted today became possible. In the 17th century, these things were novel. To put into some perspective just how important the clock, and therefore the pendulum became to society, consider Mumford's claim: "the clock, not the steam-engine, is the key-machine of the modern industrial age."[24]

Figure 6.1.1: Free body diagram for a pendulum.



Crumpet 30: The Pendulum Clock

Galileo never implemented the pendulum as a timekeeping mechanism. It was around 15 years after Galileo's death that the pendulum clock became a reality. Even though his first pendulum clock (1656) was more accurate than any other clock at the time, Huygens strived to improve upon its design. During his quest, he built a clock with a modified pendulum and published the classic work, *Horologium Oscillatorium*, where mathematical details of the isochronism of the cycloid were laid out for the first time, in 1673.[33, 21]

Today, we take for granted that the cycloid is the path a falling object must follow in order for its travel to a given point to happen in the same time regardless of its starting position. And we also take for granted that the period of a simple pendulum varies with its amplitude. We have over 400 years of physical and mathematical hindsight that tell us so!

The equation of motion

Hopefully having justified an interest in the pendulum, let us turn to a modern derivation of the motion of a pendulum by appealing to the free body diagram, a mechanical engineering mainstay. In a free body diagram, a body, in this case the bob of a pendulum, is isolated from everything except the forces acting on it. Those forces are indicated by vectors, and Newton's second law of motion (the acceleration of an object is directly proportional to the magnitude of the net force applied to the object, in the same direction as the net force, and inversely proportional to the mass of the object, or $F = ma$) is applied. Figure 6.1.1 shows the three forces acting on a pendulum—the force of gravity; the tension in the rod or string holding the bob to the pivot; and a third force called drag, which is due to air resistance—along with the directions normal ($\vec{N}$) and tangential ($\vec{T}$) to the path of the pendulum. Technically only the bob and the three forces are part of the free body diagram. Nothing else is part of the free body diagram, but is added in dashed lines to help describe the motion. The length of the pendulum is taken to be ℓ, and we will apply Newton's second law in the direction tangent to the motion. That is, in the direction $\vec{T}$.

The speed of the bob is the product of the length of the pendulum and the angular speed, $\ell\dot{\theta}$. The acceleration of the bob, the derivative of speed, is $\frac{d}{dt}\left(\ell\dot{\theta}\right) = \ell\ddot{\theta}$. Therefore, the ma (mass times acceleration) term of Newton's second law for the motion of a pendulum is $ml\ddot{\theta}$.

Gravity causes a constant downward force on the bob with magnitude equal to the weight of the bob, mg. The magnitude of this force in the $\vec{T}$ direction, however, is $mg\sin\theta$. It is worth taking a moment to make sure we have the correct sign. For values of θ between 0 and π, the bob is to the right of the pivot, so the force of gravity tends to accelerate the bob in the clockwise (negative with respect to θ) direction. Since $mg\sin\theta$ is positive for values of θ between 0 and π, the force due to gravity is actually $-mg\sin\theta$. For values of θ between $-\pi$ and 0, the bob is to the left of the pivot, so the force of gravity tends to accelerate the bob in the counterclockwise (positive with respect to θ) direction. Since $mg\sin\theta$ is negative for values of θ between $-\pi$ and 0, the force due to gravity is again $-mg\sin\theta$. Similar analysis for any other angle will lead to the same conclusion.

The damping or drag force (air resistance) is taken as a force proportional to the speed of the bob, $\ell\dot{\theta}$, so has magnitude $c\ell\dot{\theta}$. Damping forces are always taken to directly oppose the motion, so the magnitude of damping in the direction of $\vec{T}$, is its entirety. It only remains to choose the right sign. Since $\dot{\theta}$ indicates the direction of motion, the damping force must have the opposite sign. The damping constant c is taken to be positive, and of course ℓ is positive, so the damping force must be $-c\ell\dot{\theta}$.

The tension acting on the bob is irrelevant because it is always perpendicular to the motion. The component of tension in the tangential direction is always zero.

Substituting the sum of these tangential forces for F, Newton's second law applied to the pendulum becomes $-mg\sin\theta - c\ell\dot{\theta} - 0 = m\ell\ddot{\theta}$ or

$$\ddot{\theta} + \frac{c}{m}\dot{\theta} + \frac{g}{\ell}\sin\theta = 0. \tag{6.1.1}$$

Equation 6.1.1 is known as a differential equation because it is an equation that involves derivatives (or differentials). To be more precise, it is a second degree ordinary differential equation (o.d.e.). Second degree because the highest degree derivative is the second and ordinary because it involves only one independent variable (time t).

The simplest differential equations are considered in calculus, though the term "differential equation" is rarely used. When first discussing the idea of antidifferentiation, the question of "What function has a derivative equal to ... ?" inevitably comes up. For example, one might be faced with the question of what function's derivative equals x? This question can also be asked, what function y satisfies the (differential) equation $y' = x$? The answer can be arrived at by integrating the equation:

$$\begin{aligned} \int y'\,dx &= \int x\,dx \\ y &= \frac{1}{2}x^2 + C \end{aligned}$$

(don't forget the constant of integration!).

Forces in a free body diagram

The derivation of the equation of motion for the pendulum touches on three forces typically found in a free body diagram: gravity, drag, and tension. There are several other forces that may creep into a free body diagram. Most typical is the normal force a surface applies to a body lying upon it. In summary, here are the forces that should be considered when constructing a free body diagram.

Gravity: always acts directly downward with magnitude equal to the weight of the body, mg.

Drag: always acts directly opposite the direction of motion with a magnitude approximated in different ways depending on the application. This force is perhaps the most complicated to account for. It depends on the geometry of the body, the speed of the body, and the viscosity of the fluid relative to which the body moves. For slowly moving objects in low viscosity fluids, such as pendula in air, drag (air resistance) is taken proportional to the speed of the object. For faster moving objects in low viscosity fluids, drag is often taken proportional to the square of the speed of the object. In reality, drag is not exactly proportional to any power of speed, but rather varies in a very complicated way as the body moves through the fluid. For sake of tractability, though, it is almost always modeled as proportional to an appropriate power of speed. For our purposes, that power will simply be given.

Tension/compression: tension is transmited through a rope, wire, chain, or other similar object by pulling on its ends (in opposite directions). The magnitude of the tension is constant within the object assuming, as we often do, that the rope, wire, or chain is massless. Tension is always directed along the rope, wire, or chain. The opposite of tension is compression. Rigid objects such as rods, dowels, or poles are capable of transmitting compressive forces by pushing on their ends. Ropes, wires, chains, and other objects that simply slacken when pushed are not capable of transmitting compression.

Spring: a spring exerts a force proportional to the deflection of the spring, in the direction opposite the deflection.

Normal: when a body lies atop a solid surface and the body is not floating away from the surface nor sinking into the surface, there must be a balance between the forces perpendicular (normal) to the surface. The force that the surface applies to a body to keep it from sinking into the surface is called the normal force and always acts normal (perpendicular) to and away from the surface. The magnitude of the normal force is always equal to the net magnitude of all other forces in the normal direction. Often the normal component of gravity is the only other force acting normal to the surface.

Friction: when a body lies in contact with a surface, friction opposes motion with a magnitude proportional to the normal force. The constant of proportionality is called the coefficient of friction and is denoted by μ. For any body/surface combination, there are two types of friction to consider—static friction and kinetic friction. A body at rest on a surface is capable of resisting a greater force than is the same body sliding across the same surface (with the same normal force). You may be familiar with this phenomenon if you've ever tried to slide an oven into or out of its usual position in a kitchen. It's much harder to get it started moving than it is to keep it moving. Whether the friction is static or kinetic, it always resists motion tangential to the surface.

Applied: a force that is applied to a body by another body, such as a person pushing a sofa or an engine accelerating a vehicle.

Crumpet 31: Anti-lock braking systems

The anti-lock braking system (ABS) of an automobile is designed to take advantage of the fact that the static friction between a tire and the road can stop a car more quickly than the kinetic friction between the same tire and the same road. A tire that is not skidding is capable of applying a greater braking (frictional) force than the same tire skidding. When the ABS senses that a wheel has locked (ceased rotation) while the car is still moving, it forces the driver to let up on the brake enough so the wheel will start spinning again, though very briefly. If the driver continues to hold down the brake hard enough to skid, the ABS will force the driver to let up again. The ABS rapidly alternates between forcing the driver to let up and allowing the driver to do as (s)he will. The quick alternation between making the driver let up and allowing the driver to brake hard is what causes the vibration or pulsing you feel when the ABS kicks in. If the ABS is working properly, a vehicle will come to a halt more quickly than it would have if it were allowed to skid to a stop. Also, it's much easier to steer a car when it is not skidding than when it is skidding!

Solutions of ordinary differential equations

The solution of a differential equation is, in one way, very much like the solution of an algebraic equation but, in another way, entirely different. For an algebraic equation in x, for example, we say that we have a solution $x = s$ if substituting s for x in the equation makes the equation true. Likewise, for a differential equation in θ, for example, we say that we have a solution $\theta = s$ if substituting s for θ in the equation makes the equation true. The difference is s is a *number* in the case of an algebraic equation while s is a *function* in the case of a differential equation. We would say that $x = 2$ is a solution of the algebraic equation $3x^2 - 8x + 4 = 0$ since, substituting 2 for x gives

$$3(2)^2 - 8(2) + 4 = 0,$$

a true statement. Analogously, we would say that $\theta = e^{2t}$ is a solution of the differential equation $3\ddot{\theta} - 8\dot{\theta} + 4\theta = 0$ since, substituting e^{2t} for θ gives

$$3(4e^{2t}) - 8(2e^{2t}) + 4(e^{2t}) = 0,$$

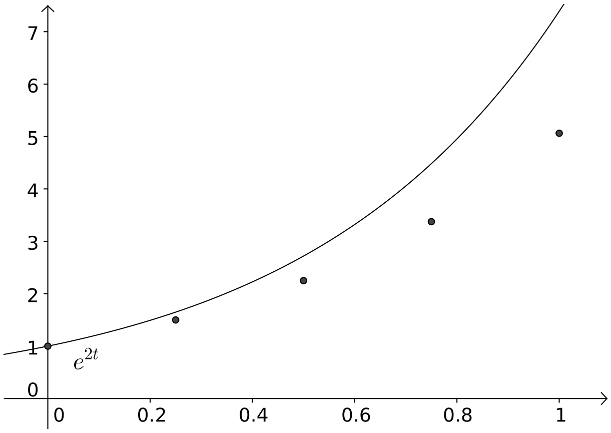
again a true statement. Notice that the derivatives $\dot{\theta}$ and $\ddot{\theta}$ need to be calculated in order to complete the substitution.

Approximate solutions of differential equations, then, must be approximations of functions. In fact, for any given ode, we settle for the crudest approximation, a set of points that, if our approximation is good, lie near the graph of an exact solution. Hence the set $\{(0, 1), (.25, 1.5), (.5, 2.25), (.75, 3.375), (1, 5.0625)\}$ might qualify as an approximate solution of the equation $3\ddot{\theta} - 8\dot{\theta} + 4\theta = 0$ for $t \in [0, 1]$. See figure 6.1.2. The approximation is good for values of t near zero but not as good for values of t near 1.

Initial Value Problems

As with algebraic equations, differential equations may have more than one solution. We already saw that $\theta = e^{2t}$ is a solution of $3\ddot{\theta} - 8\dot{\theta} + 4\theta = 0$. So are $\theta = 5e^{2t}$, $\theta = -2.1e^{2t}$, and $\theta = \sqrt{7\pi}e^{2t}$. In fact, $\theta = ce^{2t}$ is a solution for

Figure 6.1.2: Approximate solution of $3\ddot{\theta} - 8\dot{\theta} + 4\theta = 0$.



any constant c. The ode $3\ddot{\theta} - 8\dot{\theta} + 4\theta = 0$ has infinitely many solutions! It is a straightforward exercise to check. For $\theta = ce^{2t}$, $\dot{\theta} = 2ce^{2t}$ and $\ddot{\theta} = 4ce^{2t}$, so

$$\begin{aligned} 3\ddot{\theta} - 8\dot{\theta} + 4\theta &= 3(4ce^{2t}) - 8(2ce^{2t}) + 4(ce^{2t}) \\ &= 12c(e^{2t}) - 16c(e^{2t}) + 4c(e^{2t}) \\ &= (12c - 16c + 4c)e^{2t} \\ &= 0. \end{aligned}$$

Even more, $\theta = ae^{2t/3}$ is a solution for any constant a. This solution can be verified just as the solution $\theta = ce^{2t}$ was verified. Can you do it? Answer on page 179. Finally, $\theta = ce^{2t} + ae^{2t/3}$ is also a solution for any pair of constants c and a! Can you show it? Answer on page 180. It is not uncommon for a differential equation to have infinitely many solutions.

Another differential equation with infinitely many solutions is

$$\dot{y} = \frac{t}{y}.$$

The solutions are $y = \sqrt{t^2 + c}$ and $y = -\sqrt{t^2 + a}$, valid for any constants c and a as long as $y \neq 0$. Complex solutions are valid! However, if we also require $y(0) = 1$, there is only one solution! $y = -\sqrt{t^2 + c}$ is no longer a solution because it gives negative values of y for all values of t. And $y = \sqrt{t^2 + c}$ is only a solution if $c = 1$. The *one and only* solution is $y = \sqrt{t^2 + 1}$.

The requirement $y(0) = 1$ is called an initial value, or initial condition, and the pair of equations

$$\begin{aligned} \dot{y} &= \frac{t}{y} \\ y(0) &= 1 \end{aligned}$$

is called an initial value problem. More generally, the pair of equations

$$\begin{aligned} \dot{y} &= f(y, t) \\ y(t_0) &= y_0 \end{aligned}$$

forms what is knows as a first order initial value problem.

Crumpet 32: There is exactly one solution of $\dot{y} = \frac{t}{y}$ such that $y(0) = 1$.

Setting $y = \sqrt{t^2 + 1}$, $\dot{y} = \frac{1}{2}\frac{1}{\sqrt{t^2+1}}(2t) = \frac{t}{\sqrt{t^2+1}}$. Hence the equation $\dot{y} = \frac{t}{y}$ becomes

$$\frac{t}{\sqrt{t^2 + 1}} = \frac{t}{\sqrt{t^2 + 1}},$$

an undeniably true statement. Hence $y = \sqrt{t^2+1}$ is a solution of $\dot{y} = \frac{t}{y}$. Moreover $y(0) = \sqrt{0^2+1} = 1$, so the particular solution $y = \sqrt{t^2+1}$ satisfies the requirement that $y(0) = 1$ also. Hence $y = \sqrt{t^2+1}$ is *one* solution—and the only solution of the form $y = \sqrt{t^2+c}$ or $y = -\sqrt{t^2+a}$. But is it *the only* solution of any form? Perhaps there are other functions that satisfy the differential equation. A little bit of calculus should help settle the issue. The demonstration hinges on showing that $y = \sqrt{t^2+c}$ and $y = -\sqrt{t^2+a}$ are *the only* solutions of $\dot{y} = \frac{t}{y}$. The following sequence of equations show it. Each line implies the next.

$$\begin{aligned}
\frac{dy}{dt} &= \frac{t}{y}, \qquad y \neq 0 \\
y\,dy &= t\,dt, \qquad y \neq 0 \\
\int y\,dy &= C + \int t\,dt, \qquad y \neq 0 \\
\frac{1}{2}y^2 &= C + \frac{1}{2}t^2, \qquad y \neq 0 \\
y^2 &= 2C + t^2, \qquad y \neq 0 \\
y &= \pm\sqrt{t^2+2C}, \qquad y \neq 0.
\end{aligned}$$

Replacing the constant $2C$ with c or a does not change the fact that the term is an arbitrary constant, so $y = \sqrt{t^2+c}$ and $y = -\sqrt{t^2+a}$ are the only solutions of $\dot{y} = \frac{t}{y}$. This method of solving the differential equation is called separation of variables.

Key Concepts

Approximate solution of a differential equation: a set of points that, ideally, lie near the graph of an exact solution.

Degree of a differential equation: equal to the highest order derivative appearing in the equation.

Differential equation: an equation with derivatives (or differentials) in it.

Free body diagram: An engineering diagram consisting of only a body and the forces acting on it.

Initial value problem: a differential equation coupled with a required value of the solution.

Newton's second law of motion: the acceleration of an object is directly proportional to the magnitude of the net force applied to the object, in the same direction as the net force, and inversely proportional to the mass of the object—often summarized by the equation $F = ma$. This equation assumes the mass of the object is constant.

Ordinary differential equation (o.d.e.): a differential equation with only one independent variable.

Solution of a differential equation: a function that, when substituted for the dependent variable, makes the equation a true statement.

Exercises

1. State the degree of the differential equation.

   (a) $\dot{y} = y$ [A]

   (b) $y'' = 6x + \sin x$

   (c) $\ddot{s} + \dot{s} + s = 0$ [A]

   (d) $f' + \frac{f}{x} = x^2$ [S]

   (e) $(2h + x)h' + h = 4x$

   (f) $\ddot{r}\dot{r}t^2 = -\frac{1}{8}$ [A]

2. Verify that the function is a solution of the differential equation.

   (a) $y(t) = e^t$; $\dot{y} = y$ [A]

   (b) $y(x) = x^3 - 26.83x - \sin x$; $y'' = 6x + \sin x$

   (c) $s(t) = e^{-t/2}\sin\left(\frac{\sqrt{3}}{2}t\right)$; $\ddot{s} + \dot{s} + s = 0$ [A]

   (d) $f(x) = \frac{x^3}{4} + \frac{4}{x}$, $x > 0$; $f' + \frac{f}{x} = x^2$ [S]

   (e) $h(x) = -2x$; $(2h + x)h' + h = 4x$

   (f) $r(t) = \sqrt{t}$, $t > 0$; $\ddot{r}\dot{r}t^2 = -\frac{1}{8}$ [A]

3. Verify that the function is a solution of the initial value problem.

   (a) $y(t) = 4e^t$; $\dot{y} = y$, $y(0) = 4$ [A]

   (b) $y(x) = x^3 - \sin x - \pi^3$; $y' = 3x^2 - \cos x$, $y(\pi) = 0$

(c) $s(t) = \frac{1}{2}\left(1 + e^{-t^2}\right)$; $\dot{s} = (1-2s)t$, $s(0) = 1$ [𝔸]

(d) $f(x) = \frac{x^3}{4} + \frac{16}{x}$, $x > 0$; $f' = -\frac{f}{x} + x^2$, $f(4) = 20$ [𝕊]

(e) $h(x) = -2x - 1$; $h' = \frac{1+4x-h}{2h+x+1}$, $h(0) = -1$

(f) $r(t) = \sqrt{t} - 3$, $t > 0$; $\ddot{r}\dot{r}t^2 = -\frac{1}{8}$, $r(9) = 0$, $\dot{r}(9) = \frac{1}{6}$. [𝔸]HINT: The solution must satisfy the o.d.e. and both conditions, $r(9) = 0$ and $\dot{r}(9) = \frac{1}{6}$.

4. Solve the differential equation.

(a) $y' = 5x^4$ [𝔸]

(b) $y' = 3xe^{x^2}$

(c) $\dot{y} = t - \sin t$ [𝕊]

(d) $\dot{y} = \frac{1}{t}$, $t < 0$ [𝔸]

(e) $s' = 1 - \ln x$

(f) $\dot{s} = 3te^t$ [𝔸]

5. Given are an initial value problem, its exact solution, and an approximate solution. Comment on how well the approximate solution approximates the exact solution.

(a) $\dot{y} = y$, $y(0) = 4$; $y(t) = 4e^t$; $\{(0,4), (.25,5), (.5,6.3), (.75,7.8), (1,9.8)\}$ [𝔸]

(b) $y' = 3x^2 - \cos x$, $y(\pi) = 0$; $y(x) = x^3 - \sin x - \pi^3$; $\{(\pi,0), (\frac{5}{4}\pi,30), (\frac{3}{2}\pi,74), (\frac{7}{4}\pi,135), (2\pi,216)\}$

(c) $\dot{s} = (1-2s)t$, $s(0) = 1$; $s(t) = \frac{1}{2}\left(1 + e^{-t^2}\right)$; $\{(0,1), (.5,1), (1,.75), (1.5,.5), (2,.5)\}$ [𝔸]

(d) $f' = -\frac{f}{x} + x^2$, $f(4) = 20$; $f(x) = \frac{x^3}{4} + \frac{16}{x}$; $\{(4,20), (4.25,23), (4.5,26), (4.75,30), (5,34)\}$ [𝕊]

(e) $h' = \frac{1+4x-h}{2h+x+1}$, $h(0) = -1$; $h(x) = -2x - 1$; $\{(0,-1), (.25,-1.5), (.5,-2), (.75,-2.5), (1,-3)\}$

(f) $\ddot{r}\dot{r}t^2 = -\frac{1}{8}$, $r(9) = 0$, $\dot{r}(9) = -\frac{1}{6}$; $r(t) = \sqrt{t} - 3$; $\{(9,0), (10,.16), (11,.31), (12,.46), (13,.61)\}$ [𝔸]

6. Draw a free body diagram for the situation.

(a) Pendular motion ignoring air resistance (no damping). [𝔸]

(b) A block sliding down an inclined plane. [𝔸]

(c) A block sitting on an inclined plane (not moving). [𝕊]

(d) A block being pushed up an inclined plane.

(e) A sofa being pushed across a level floor where the applied force is parallel to the floor. [𝔸]

(f) A sofa being pushed across a level floor where the applied force is not parallel to the floor. [𝕊]

(g) A sofa being pushed up an old, slanted hardwood floor. The applied force may or may not be parallel to the floor. [𝔸]

(h) A sledder has reached the bottom of a hill (and is now traveling on level snow) and is coasting to a stop. [𝔸]

(i) A sledder sledding down a hill. [𝔸]

(j) A hockey puck sliding across an ice rink. [𝔸]

(k) A hockey puck sliding across ice at constant speed (ignoring friction).

(l) A sky diver falling. [𝔸]

(m) A sky diver whose parachute just opened. [𝕊]

(n) A sky diver whose parachute just opened while a constant breeze is blowing sideways. [𝔸]

(o) A football originally kicked at a 40 degree angle just as it reaches its peak, ignoring drag. [𝔸]

(p) A football moving up and to the right approaching its peak, ignoring drag.

7. Use the free body diagram from question 6 to find the equation of motion in the tangential direction for (6a)-(6k), and in the vertical direction for (6l)-(6p). [𝕊][𝔸]

8. How much easier is it to slide a sofa by pushing parallel to the floor as opposed to slightly toward the floor? Compare the kinetic friction for a sofa being pushed parallel to the floor to one being pushed at an angle of 20 degrees from parallel. Then calculate the necessary applied force to overcome kinetic friction in each case. Assume the floor is level. [𝔸]

Answers

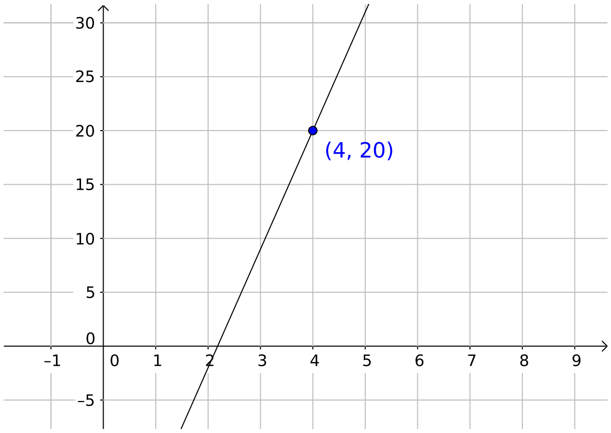
$\theta = ae^{2t/3}$ **is a solution of** $3\ddot{\theta} - 8\dot{\theta} + 4\theta = 0$: $\dot{\theta} = \frac{2}{3}ae^{2t/3}$ and $\ddot{\theta} = \frac{4}{9}ae^{2t/3}$ so

$$\begin{aligned}
3\ddot{\theta} - 8\dot{\theta} + 4\theta &= 3\left(\frac{4}{9}ae^{2t/3}\right) - 8\left(\frac{2}{3}ae^{2t/3}\right) + 4\left(ae^{2t/3}\right) \\
&= \frac{4}{3}a(e^{2t/3}) - \frac{16}{3}a(e^{2t/3}) + \frac{12}{3}a(e^{2t/3}) \\
&= \left(\frac{4}{3}a - \frac{16}{3}a + \frac{12}{3}a\right)e^{2t/3} \\
&= 0.
\end{aligned}$$

$\theta = ce^{2t} + ae^{2t/3}$ **is a solution of** $3\ddot{\theta} - 8\dot{\theta} + 4\theta = 0$**:** $\dot{\theta} = 2ce^{2t} + \frac{2}{3}ae^{2t/3}$ and $\ddot{\theta} = 4ce^{2t} + \frac{4}{9}ae^{2t/3}$ so

$$\begin{aligned}
3\ddot{\theta} - 8\dot{\theta} + 4\theta &= 3\left(4ce^{2t} + \frac{4}{9}ae^{2t/3}\right) - 8\left(2ce^{2t} + \frac{2}{3}ae^{2t/3}\right) + 4\left(ce^{2t} + ae^{2t/3}\right) \\
&= 12c(e^{2t}) + \frac{4}{3}a(e^{2t/3}) - 16c(e^{2t}) - \frac{16}{3}a(e^{2t/3}) + 4c(e^{2t}) + \frac{12}{3}a(e^{2t/3}) \\
&= (12c - 16c + 4c)e^{2t} + \left(\frac{4}{3}a - \frac{16}{3}a + \frac{12}{3}a\right)e^{2t/3} \\
&= 0.
\end{aligned}$$

Figure 6.2.1: Beginning a numerical solution with the initial condition



6.2 Taylor Methods

The exact solution of the initial value problem

$$\begin{aligned} \dot{y} &= -\frac{y}{t} + t^2 \\ y(4) &= 20 \end{aligned} \tag{6.2.1}$$

is $y(t) = \frac{t^3}{4} + \frac{16}{t}$, $t > 0$, as verified in exercise 3d on page 179. For the time being, let us try to forget that we know the exact solution, and study a method for approximating it. We will recall that we have the exact solution when we are ready to check how the approximation is going. The initial condition, $y(4) = 20$, means that the graph of the exact solution passes through $(4, 20)$. What a great place to start an approximate solution—at a point that is on the graph of the exact solution! Thus the approximation is seeded by the initial condition. There are numerous ways to proceed from there. Perhaps the simplest way is to use the differential equation to compute the exact slope (derivative) of y at $(4, 20)$:

$$\begin{aligned} \dot{y}(4) &= -\frac{y(4)}{4} + 4^2 \\ &= -\frac{20}{4} + 4^2 \\ &= 11. \end{aligned}$$

You might imagine a graph like that in figure 6.2.1. The graph is that of the first order Taylor polynomial expanded about $t_0 = 4$. According to Taylor's theorem, $y(t) = 20 + 11(t-4) + \frac{\ddot{y}(\xi)}{2}(t-4)^2$ for t near 4 and some ξ, depending on t. So, $y(2) \approx T_1(2) = 20 + 11(2-4) = -2$ and $y(5) \approx T_1(5) = 20 + 11(5-4) = 31$ (as long as y has two derivatives on an open interval containing $[2, 5]$), and so on. As always, there is the concern of how good these approximations are.

In section 4.4, two different approximations for the same number were used to estimate error in the adaptive methods. A similar tack may be used here. We will compare approximations given by T_1 and T_2. The differential equation can be used to compute $\ddot{y}$, in terms of y and t. Implicitly differentiating the differential equation gives

$$\ddot{y} = -\frac{\dot{y}t - y}{t^2} + 2t.$$

But $\dot{y} = -\frac{y}{t} + t^2$, so we may substitute into and simplify the expression for $\ddot{y}$:

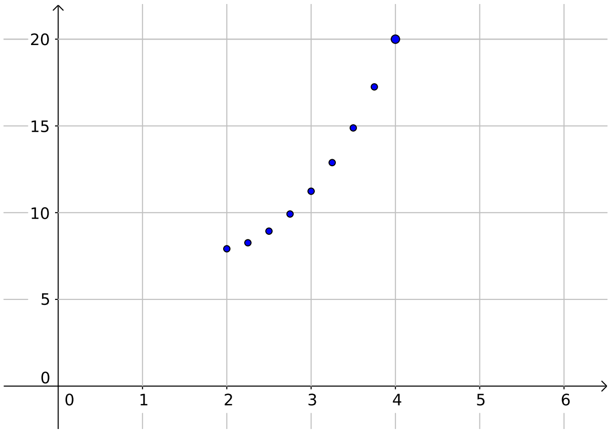
$$\begin{aligned} \ddot{y} &= -\frac{(-\frac{y}{t} + t^2)t - y}{t^2} + 2t \\ &= -\frac{-y + t^3 - y}{t^2} + 2t \\ &= \frac{2y}{t^2} - \frac{t^3}{t^2} + 2t \\ &= \frac{2y}{t^2} + t. \end{aligned}$$

Table 6.1: Comparing first and second order polynomial approximations

t	$T_1(t)$	$T_2(t)$
2	-2	11
5	31	34.25

Figure 6.2.2: A repetitive numerical calculation (truncated to 5 decimal places)

t_0	$y(t_0)$
4	20
3.75	17.25
3.5	14.88437
3.25	12.88504
3	11.23557
2.75	9.92187
2.5	8.93323
2.25	8.26406
2	7.91666



Now we know $\ddot{y}(4) = \frac{2y(4)}{4^2} + 4 = \frac{2(20)}{16} + 4 = \frac{13}{2}$, so $T_2(t) = 20 + 11(t-4) + \frac{13}{4}(t-4)^2$. Finally, we can compare values of T_1 to corresponding values of T_2, as in Table 6.1. $T_1(2)$ and $T_2(2)$ disagree wildly, so we should assume that neither approximation is to be trusted. $T_1(5)$ and $T_2(5)$ differ by only around 10%, so these approximations may be reasonable. To further hone the approximation of $y(2)$, it is possible to calculate $T_3(2)$ and again compare. Can you do it? Answer on page 186.

Another way to approximate $y(2)$ is to take things a little more slowly. We could use the initial condition to approximate $y(3.75)$ first. Then we could use this approximation to approximate $y(3.5)$, which we could, in turn, use to approximate $y(3.25)$, and so on until we ultimately use the approximation of $y(2.25)$ to approximate $y(2)$. We humans may think the prospect of doing all these calculations is repugnant, but with a little computer code, the burden is placed on the machine. It is the ability to understand the process well enough to write that code that now becomes the focus.

We know that $y(4) = 20$ and we are interested in approximating $y(3.75)$. Since the difference between 4 and 3.75 is only .25, perhaps using T_1 will be sufficiently accurate. From before, we know the Taylor polynomial expanded about $t_0 = 4$ is $T_1(t) = 20 + 11(t-4)$, so $T_1(3.75) = 20 + 11(-.25) = 17.25$. Now we can use $y(3.75) = 17.25$ as a "new" initial condition. $\dot{y}(3.75) = -\frac{17.25}{3.75} + 3.75^2 = 9.4625$. We can use this information to approximate the Taylor polynomial for y expanded about 3.75: $T_1(t) \approx 17.25 + 9.4625(t - 3.75)$, and use this expansion to approximate $y(3.5)$: $y(3.5) \approx T_1(3.5) \approx 17.25 + 9.4625(3.5 - 3.75) = 14.884375$. We then can use $y(3.5) = 14.884375$ as an initial condition, approximating the Taylor polynomial for y expanded about 3.5. Continuing in this vein leads to the tabular and graphical results in Figure 6.2.2. Can you reproduce these results? Details on page 186.

The method of repeated calculation leads to $y(2) \approx 7.91$, but more importantly, illuminates an algorithm for approximating solutions of differential equations. Calling the initial condition (t_0, y_0), and succeeding points (t_1, y_1),(t_2, y_2),$(t_3, y_3)\ldots$, the same procedure is used to calculate (t_1, y_1) from (t_0, y_0) as is used to calculate (t_2, y_2) from (t_1, y_1) as is used to calculate (t_3, y_3) from (t_2, y_2), and so on. It remains to capture that procedure as a formula of some sort. To summarize, the procedure is to use a given point, call it (t_i, y_i) to

1. calculate $\dot{y}(t_i, y_i)$;

2. use the three values t_i, y_i, and $\dot{y}(t_i, y_i)$ to form $T_1(t)$ expanded about t_i; and finally

3. set $y_{i+1} = T_1(t_{i+1})$, which gives a new point, (t_{i+1}, y_{i+1}).

But $T_1(t_{i+1}) = y_i + \dot{y}(t_i, y_i) \cdot (t_{i+1} - t_i)$, so the procedure really boils down to setting

$$y_{i+1} = y_i + \dot{y}(t_i, y_i) \cdot (t_{i+1} - t_i). \tag{6.2.2}$$

The method of using formula (6.2.2) repeatedly to compute a sequence of points approximately on the solution of an ordinary differential equation is most often called Euler's method.[7] It may also be referred to as the Taylor method of degree 1 since it uses Taylor polynomials of degree 1 at each step. The value $t_{i+1} - t_i$ is called the step size and is often held constant, so you are likely to see Euler's method written as

$$y_{i+1} = y_i + h \cdot \dot{y}(t_i, y_i) \tag{6.2.3}$$

where $h = t_{i+1} - t_i$ is the constant step size.

Euler's Method (pseudo-code)

As is most common, Euler's method will be coded for a constant step size.

```
Assumptions: The solution of the o.d.e. exists and is unique on the interval from $t_0$ to $t_1$.
Input: Differential equation $\dot{y} = f(t, y)$; initial condition $y(t_0) = y_0$; numbers $t_0$ and $t_1$; number of steps $N$.
Step 1: Set $t = t_0$; $y = y_0$; $h = (t_1 - t_0)/N$
Step 2: For $j = 1 \ldots N$ do Steps 3-4:
    Step 3: Set $y = y + hf(t, y)$
    Step 4: Set $t = t_0 + \frac{i}{N}(t_1 - t_0)$
Output: Approximation $y$ of the solution at $t = t_1$.
```

Higher Degree Taylor Methods

Taylor methods of higher degree are rarely used in practice because they require computation of derivatives, a task that is not always easy or even possible. Nonetheless, it is not a huge stretch from what we have already done to consider higher degree methods. Rewriting the steps outlined in the enumeration that leads to 6.2.2, the third degree Taylor method can be summarized by

1. calculate $\dot{y}(t_i, y_i)$ and $\ddot{y}(t_i, y_i)$ and $\dddot{y}(t_i, y_i)$;

2. use the ~~three~~ five values t_i, y_i, ~~and~~ $\dot{y}(t_i, y_i)$, $\ddot{y}(t_i, y_i)$, and $\dddot{y}(t_i, y_i)$ to form ~~$T_1(t)$~~ $T_3(x)$ expanded about t_i; and finally

3. set ~~$y_{i+1} = T_1(t_{i+1})$~~ $y_{i+1} = T_3(t_{i+1})$, which gives a new point, (t_{i+1}, y_{i+1}).

Now written without all the markup, the procedure is

1. calculate $\dot{y}(t_i, y_i)$, $\ddot{y}(t_i, y_i)$, and $\dddot{y}(t_i, y_i)$;

2. use the five values t_i, y_i, $\dot{y}(t_i, y_i)$, $\ddot{y}(t_i, y_i)$, and $\dddot{y}(t_i, y_i)$ to form $T_3(x)$ expanded about t_i; and finally

3. set $y_{i+1} = T_3(t_{i+1})$, which gives a new point, (t_{i+1}, y_{i+1}).

Higher degree Taylor methods require higher derivatives in step 1 and a higher degree Taylor polynomial in steps 2 and 3. As should be expected, higher degree methods are generally more accurate than lower degree methods as long as the formula for $\dot{y}(t, y)$ is sufficiently smooth. To illustrate the point, we now compare approximate solutions of 6.2.1.

Taylor's Method of Degree 3 (pseudo-code)

Taylor's method of degree 3 will be coded for a constant step size.

```
Assumptions: The solution of the o.d.e. exists and is unique on the interval from $t_0$ to $t_1$.
Input: Differential equation $\dot{y} = f(t, y)$; formulas $\ddot{y}(t, y)$ and $\dddot{y}(t, y)$; initial condition $y(t_0) = y_0$; numbers
    $t_0$ and $t_1$; number of steps $N$.
Step 1: Set $t = t_0$; $y = y_0$; $h = (t_1 - t_0)/N$
Step 2: For $j = 1 \ldots N$ do Steps 3-4:
    Step 3: Set $y = y + hf(t, y) + \frac{1}{2}h^2\ddot{y}(t, y) + \frac{1}{6}h^3\dddot{y}(t, y)$
    Step 4: Set $t = t_0 + \frac{i}{N}(t_1 - t_0)$
Output: Approximation $y$ of the solution at $t = t_1$.
```

Table 6.2: Approximate values of $y(2)$ from solving 6.2.1

	$h = 0.5$	error	$h = 0.25$	error	$h = 0.125$	error
Euler's method	6.1	3.9	7.91666	2.08333	8.91911	1.08088
Taylor's degree 3 method	9.975765	0.024234	9.996280	0.003719	9.999485	0.000514

Using code based on the pseudo-code presented in this section, Table 6.2 summarizes the approximate solution of 6.2.1 using Euler's method and Taylor's method of degree 3 to approximate $y(2)$.

Now is a good time to say something about the error of Taylor methods. Remember a Taylor polynomial of degree n has an error of order $n+1$, so Euler's method uses a Taylor polynomial with error of order 2 and Taylor's degree 3 method uses a Taylor polynomial with error of order 4. But how does that translate into an error term for the Taylor *method*?

Though we will not answer this question completely here, we can get some idea what to expect from Table 6.2. From the Euler's method row, we see the error decrease from (roughly) 3.9 to 2.08 to 1.08 as the step size is reduced by a factor of one half. Since

$$\frac{2.08}{3.9} \approx \frac{1.08}{2.08} \approx \left(\frac{1}{2}\right)^1,$$

we conclude that Euler's method is of first order. Considering the row on Taylor's degree 3 method, we see the error decrease from about .024 to .0037 to .00051 as the step size is reduced by a factor of one half. Since

$$\frac{.0037}{.024} \approx \frac{.00051}{.0037} \approx \frac{1}{8} = \left(\frac{1}{2}\right)^3,$$

we conclude that Taylor's degree 3 method is of order 3.

Notice the similarity between this observation and the observation we made about composite integration. In section 4.4, we argued that the error term for a composite integration formula had order one less than that of a single application of the underlying integration formula. The same thing happens here. When the truncation error for the underlying Taylor polynomial has order n, the corresponding o.d.e. solver has order $n-1$, an order equal to the degree of the Taylor polynomial itself.

Reducing a second order equation to a first order system

Taylor's methods and the upcoming Runge-Kutta methods are all designed to work on first order differential equations. However, all the equations of motion we have developed are second order differential equations. To resolve this disconnect, a second order o.d.e. can be reduced to a first order system. The idea is straightforward. Suppose y is the dependent variable in a second order o.d.e. and we have an equation of the form $y'' = f(y', y, x)$. We introduce an auxiliary variable u and set $u = y'$. Consequently, $u' = y'' = f(y', y, x) = f(u, y, x)$. We thus have the first order system

$$\begin{aligned} u' &= f(u, y, x) \\ y' &= u \end{aligned}$$

which can be solved using a numerical method for first order differential equations.

For example, the equation of a pendulum (6.1.1) can be rearranged as $\ddot{\theta} = -\frac{c}{m}\dot{\theta} - \frac{g}{\ell}\sin\theta$. If we substitute the auxiliary variable $u = \dot{\theta}$ into the equation, it becomes $\dot{u} = -\frac{c}{m}u - \frac{g}{\ell}\sin\theta$, and the system

$$\begin{aligned} \dot{u} &= -\frac{c}{m}u - \frac{g}{l}\sin\theta \\ \dot{\theta} &= u \end{aligned}$$

is equivalent to (6.1.1). Euler's method, for example, can be applied to this system in the following way:

$$\begin{aligned} u_{n+1} &= u_n + h\left(-\frac{c}{m}u_n - \frac{g}{l}\sin\theta_n\right) \\ \theta_{n+1} &= \theta_n + hu_n \\ t_{n+1} &= t_n + h \end{aligned}$$

where u_0, θ_0, and t_0 are taken from the initial conditions.

Key Concepts

Taylor method: A method for approximating the solution of a first order o.d.e. in which a Taylor polynomial of some predetermined order is used at each step to compute the next.

Euler's method: Another name for the first order Taylor method, having formula $y_{i+1} = y_i + h \cdot \dot{y}(t_i, y_i)$.

Exercises

1. Use Euler's method with step size $h = 0.5$ to approximate $y(2)$.

   (a) [S]

   $$\begin{aligned} \frac{dy}{dx} &= 3x - 2y \\ y(1) &= 1 \end{aligned}$$

   (b)

   $$\begin{aligned} \frac{dy}{dx} &= 3x^3 - y \\ y(1) &= 3 \end{aligned}$$

   (c) [A]

   $$\begin{aligned} \dot{y} &= ty \\ y(1) &= 0.5 \end{aligned}$$

   (d) [S]

   $$\begin{aligned} \cos(x)y' + \sin(x)y &= 2\cos^3(x)\sin(x) - 1 \\ y(1) &= 0 \end{aligned}$$

   (e)

   $$\begin{aligned} 7\dot{y} + 3y &= 5 \\ y(1) &= 2 \end{aligned}$$

2. Repeat exercise 1 using Taylor's method of order 2. [S] [A]

3. Repeat exercise 1 using Taylor's method of order 3. [S] [A]

4. Execute two steps of Euler's method for solving $\dot{y} = ty$ with $y(1) = -0.5$ and $h = 0.25$, thus approximating $y(1.5)$. [A]

5. Write pseudo-code for Taylor's method of order 2. [A]

6. Write pseudo-code for Taylor's method of order 4.

7. C Write computer code that implements Euler's method. [S]

8. C Write computer code that implements Taylor's method of degree 2. [A]

9. C Write computer code that implements Taylor's method of degree 3.

10. C Write computer code that implements Taylor's method of degree 4.

11. Use your code from exercise 8 to calculate $y(2)$ for the o.d.e. in 1a uisng $h = 0.5,\ 0.25,\ 0.125,$ and 0.0625. Use your calculations and the fact that the exact value of $y(2)$ is $\frac{9+e^{-2}}{4}$ to verify that Taylor's method of degree 2 is an order 2 numerical method. [A]

12. Use your code from exercise 9 to calculate $y(2)$ for the o.d.e. in 1a uisng $h = 0.5,\ 0.25,\ 0.125,$ and 0.0625. Use your calculations and the fact that the exact value of $y(2)$ is $\frac{9+e^{-2}}{4}$ to verify that Taylor's method of degree 3 is an order 3 numerical method.

13. Use your code from exercise 10 to calculate $y(2)$ for the o.d.e. in 1a uisng $h = 0.5,\ 0.25,\ 0.125,$ and 0.0625. Use your calculations and the fact that the exact value of $y(2)$ is $\frac{9+e^{-2}}{4}$ to verify that Taylor's method of degree 4 is an order 4 numerical method.

14. Write the equation of motion you derived in exercise 7 on page 179 as a first order system. [S] [A]

15. Given the following parameter values and initial conditions for the referenced system, use Euler's method with a step size $h = 0.25$ to compute $s(0.5)$ or $\theta(0.5)$ as appropriate.

    14a: $g = 9.81$ m/s^2; $\ell = .31$ m; $\theta(0) = \frac{\pi}{3}$; $\dot{\theta}(0) = 0$ [A]

    14b: $g = 32.2$ ft/s^2; $\mu = .21$; $\alpha = .25$ rad; $s(0) = 0$; $\dot{s}(0) = .3$ ft/s [A]

    14c: $g = 32.2$ ft/s^2; $\mu = .21$; $\alpha = .25$ rad; $s(0) = 0$; $\dot{s}(0) = 0$ [S]

    14d: $g = 32.2$ ft/s^2; $\mu = .21$; $\alpha = .25$ rad; $m = .19$ lbm; $F_{applied} = 15$ lb; $s(0) = 0$; $\dot{s}(0) = 1$ ft/s

    14e: $g = 9.81$ m/s^2; $\mu = .15$; $m = 35$ kg; $F_{applied} = 75$ N; $s(0) = 0$; $\dot{s}(0) = .03$ m/s [A]

    14f: $g = 9.81$ m/s^2; $\mu = .15$; $\beta = \frac{\pi}{10}$ rad; $m = 35$ kg; $F_{applied} = 75$ N; $s(0) = 0$; $\dot{s}(0) = .03$ m/s [S]

    14g: $g = 9.81$ m/s^2; $\mu = .15$; $\alpha = .05$ rad; $\beta = \frac{\pi}{10}$ rad; $m = 35$ kg; $F_{applied} = 90$ N; $s(0) = 0$; $\dot{s}(0) = .03$ m/s [A]

    14h: $g = 32.2$ ft/s^2; $\mu = .01$; $s(0) = 0$; $\dot{s}(0) = 30$ ft/s [A]

    14i: $g = 32.2$ ft/s^2; $\mu = .01$; $\alpha = \frac{\pi}{6}$ rad; $s(0) = 0$; $\dot{s}(0) = 10$ ft/s [A]

    14j: $g = 32.2$ ft/s^2; $\mu = .003$; $s(0) = 0$; $\dot{s}(0) = 88$ ft/s [A]

    14k: $g = 32.2$ ft/s^2; $\mu = 0$; $s(0) = 0$; $\dot{s}(0) = 88$ ft/s

    14l: $g = 9.81$ m/s^2; $c = 4.5$; $m = 70$ kg; $s(0) = 10000$; $\dot{s}(0) = -10$ m/s [A]

    14m: $g = 9.81$ m/s^2; $c = 26$; $m = 70$ kg; $s(0) = 2000$; $\dot{s}(0) = -55$ m/s [S]

16. Find a formula for the angle at which a stationary block on an inclined plane (whose angle of inclination is increasing) will start moving.

17. Find a formula for the angle at which a block moving down an inclined plane (whose angle of inclination is decreasing) will stop moving.

18. **Undetermined Coefficients.** For each differential equation, a solution with undetermined coefficients is suggested. Find values for the coefficients that make the suggested solution an actual solution.

    (a) [S]$y'' + 5y' - 8y = 3x^2$; $y(x) = Ax^2 + Bx + C$

    (b) $2y''' - 5y'' + 3y' + 5y = x + 1$; $y(x) = Ax + B$

    (c) [A]$3y' + 2y = 3x + 2$; $y(x) = Ax + B$

    (d) [A]$y'' - 14y' + 7y = 2x^2 + 3x - 1$; $y(x) = Ax^2 + Bx + C$

    (e) [A]$2\dot{y} + y = t^4 + 1$; $y(t) = A + Bt + Ct^2 + Dt^3 + Et^4$

    (f) $\ddot{x} + 2\dot{x} - x = 1 + te^t$; $x(t) = Ate^t + Be^t + C$

    (g) [A]$\dot{\theta} - \theta = e^{-t}\sin t$; $\theta(t) = Ae^{-t}\sin t + Be^{-t}\cos t$

    (h) [S]$\ddot{\theta} + \frac{1}{10}\dot{\theta} + \theta = t\cos t$; $\theta(t) = At\cos t + Bt\sin t + C\cos t + D\sin t$

    (i) [A]$\ddot{x} - 2\dot{x} - 35x = e^{7t} + 1$; $x(t) = Ate^{7t} + Be^{7t} + C$

Answers

$T_3(2)$**:** Begin by calculating $\dddot{y} = \frac{d}{dt}\ddot{y}$.

$$\begin{aligned}
\dddot{y} &= \frac{d}{dt}\left(\frac{2y}{t^2} + t\right) \\
&= \frac{2\dot{y}t^2 - 4ty}{t^4} + 1 \\
&= \frac{2\left(-\frac{y}{t} + t^2\right)t^2 - 4ty}{t^4} + 1 \\
&= \frac{-2ty + 2t^4 - 4ty}{t^4} + 1 \\
&= \frac{-6y}{t^3} + 3
\end{aligned}$$

so $\dddot{y}(4) = \frac{-6(20)}{4^3} + 3 = 3 - \frac{120}{64} = \frac{9}{8}$. Therefore, $T_3(t) = 20 + 11(t-4) + \frac{13}{4}(t-4)^2 + \frac{3}{16}(t-4)^3$, and $T_3(2) = 9.5$ so it is close to $T_2(2) = 11$. We can start to believe that $y(2)$ is somewhere around 9.5 or 11.

Details:

t_0	$y(t_0)$	$\dot{y}(t_0)$	T_1 expanded about t_0	$T_1(t_0 - .25)$
4	20	11	$20 + 11(t-4)$	17.25
3.75	17.25	9.4625	$17.25 + 9.4625(t - 3.75)$	14.88437
3.5	14.88437	7.99732	$14.88437 + 7.99732(t - 3.5)$	12.88504
3.25	12.88504	6.59787	$12.88504 + 6.59787(t - 3.25)$	11.23557
3	11.23557	5.25480	$11.23557 + 5.25480(t - 3)$	9.92187
2.75	9.92187	3.95454	$9.92187 + 3.95454(t - 2.75)$	8.93323
2.5	8.93323	2.67670	$8.93323 + 2.67670(t - 2.5)$	8.26406
2.25	8.26406	1.38958	$8.26406 + 1.38958(t - 2.25)$	7.91666
2	7.91666			

6.3 Foundations for Runge-Kutta Methods

In section 6.2, derivatives were used to generate approximate solutions of ordinary differential equations. However, approximate solutions can also be generated by integrating, a much more stable numerical process. An o.d.e. of the form

$$\begin{aligned} \dot{y} &= f(t,y) \\ y(t_0) &= y_0 \end{aligned}$$

has an exact solution that can be written in terms of an integral. For any value $\tilde{t}$, and assuming existence of a solution over the interval from t_0 to $\tilde{t}$, we can find a value for $y(\tilde{t})$ by integrating both sides of $\dot{y} = f(t,y)$ with respect to t:

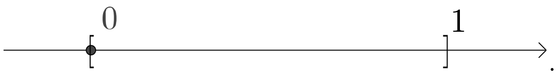
$$\begin{aligned} \int_{t_0}^{\tilde{t}} \dot{y}\,dt &= \int_{t_0}^{\tilde{t}} f(t,y)\,dt \\ y(\tilde{t}) - y(t_0) &= \int_{t_0}^{\tilde{t}} f(t,y)\,dt \\ y(\tilde{t}) &= y(t_0) + \int_{t_0}^{\tilde{t}} f(t,y)\,dt. \qquad (6.3.1) \end{aligned}$$

When t_0 and $\tilde{t}$ are not close to one another, which is what we normally assume, we need to proceed in small steps as done in section 6.2.

Substituting t_1 for $\tilde{t}$ in equation 6.3.1, $y(t_1) = y(t_0) + \int_{t_0}^{t_1} f(t,y)\,dt$, so we can add $\int_{t_0}^{t_1} f(t,y)\,dt$ to the known value $y(t_0)$ to get $y(t_1)$, our first small step on the way to approximating $y(\tilde{t})$. Now substituting t_1 for t_0 and t_2 for $\tilde{t}$ in equation 6.3.1, $y(t_2) = y(t_1) + \int_{t_1}^{t_2} f(t,y)\,dt$. So, we can compute $y(t_2)$ from knowledge of $y(t_1)$. Similarly we can compute $y(t_3)$ from knowledge of $y(t_2)$, $y(t_4)$ from knowledge of $y(t_3)$, and so on, eventually computing $y(t_n) = y(\tilde{t})$. With this in mind, we rewrite the integral representation in terms of t_i and t_{i+1} instead of t_0 and $\tilde{t}$:

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t,y)\,dt. \qquad (6.3.2)$$

This formula suggests that finding one approximation, $y(t_{i+1})$, from the previous, $y(t_i)$, boils down to approximating $\int_{t_i}^{t_{i+1}} f(t,y)\,dt$. That should not be too challenging at this point. About half of chapter 4 is dedicated to exactly this task! Every numerical integration formula is a candidate for use here, but let's start simple. We know $y(t_i)$, the value of the function at the left endpoint of integration, at least approximately, so it makes sense to use a stencil that includes the left endpoint of integration as one of the nodes. And to make our first stab as easy as possible, let's let that node be the only one! That is, let's find an integration formula for the stencil



Using the method of undetermined coefficients, we calculate the left hand side of system 4.2.4 (which for us will only be one equation since we only have one node):

$$\int_a^b p_0(x)dx = \int_{x_0}^{x_0+h} p_0(x)dx \quad = \quad \int_{x_0}^{x_0+h} 1dx = (x-x_0)|_{x_0}^{x_0+h} = h$$

and the right hand side:

$$\sum_{i=0}^{0} (\theta_i h)^0 a_i = a_0.$$

So $a_0 = h$ and we get the formula

$$\int_{x_0}^{x_0+h} f(x)dx \approx hf(x_0).$$

Consequently, $\int_{t_i}^{t_{i+1}} f(t,y)\,dt \approx (t_{i+1} - t_i)f(t_i, y(t_i))$, and equation 6.3.2 becomes

$$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i)) \cdot (t_{i+1} - t_i).$$

Adopting the notation $y_i = y(t_i)$ and $f = \dot{y}$ from section 6.2, this formula becomes

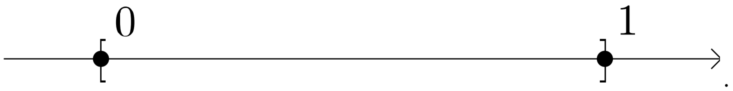
$$y_{i+1} = y_i + \dot{y}(t_i, y_i) \cdot (t_{i+1} - t_i).$$

Wait a minute! We've seen this before. This is exactly equation 6.2.2.

The search for new methods of approximating solutions of o.d.e.s by integrating has not yielded anything new yet. It has to be different, however. Integration formulas include evaluation of the integrand at various points while Taylor methods involve evaluation of derivatives at a single point. Let's push on. Perhaps the next simplest integration formula that includes the left endpoint of integration is the trapezoidal rule (see section 4.3),

$$\int_{x_0}^{x_0+h} f(x)dx = \frac{h}{2}\left[f(x_0) + f(x_0+h)\right] + O(h^3 f''(\xi_h))$$

over the stencil



Translating the trapezoidal rule to the current notation,

$$\int_{t_i}^{t_{i+1}} f(t,y)\,dt = \frac{t_{i+1}-t_i}{2}\left[f(t_i,y_i) + f(t_{i+1},y_{i+1})\right] + O((t_{i+1}-t_i)^3).$$

Therefore our new approximation formula is

$$y_{i+1} = y_i + \frac{t_{i+1}-t_i}{2}\left[f(t_i,y_i) + f(t_{i+1},y_{i+1})\right].$$

This equation is great except the right hand side includes y_{i+1}, the quantity we are trying to approximate! One theory is to leave it at that. The equation for y_{i+1} is implicit in nature and that's alright. Some root finding method could be used to determine y_{i+1} for each step of the method. While this path is not impossible, it is also not the simplest solution. Since the step size $(t_{i+1} - t_i)$ is likely to be small, perhaps using Euler's method to approximate y_{i+1} on the right side will not cause irreparable harm to the overall approximation. Giving it a shot, we let $y_{i+1} = y_i + (t_{i+1} - t_i) \cdot f(t_i, y_i)$ on the right hand side to get the new formula

$$y_{i+1} = y_i + \frac{t_{i+1}-t_i}{2}\left[f(t_i,y_i) + f(t_{i+1}, y_i + (t_{i+1}-t_i)\cdot f(t_i,y_i))\right].$$

Pausing for a moment to consider what we have, we might conclude the formula is getting a little unwieldy. Let's see if we can tidy it up a bit. First, substituting h for $t_{i+1} - t_i$ makes it a little nicer:

$$y_{i+1} = y_i + \frac{h}{2}\left[f(t_i,y_i) + f(t_{i+1}, y_i + h\cdot f(t_i,y_i))\right].$$

Second, letting $k_1 = f(t_i, y_i)$ and $k_2 = f(t_{i+1}, y_i + h \cdot f(t_i, y_i)) = f(t_{i+1}, y_i + h \cdot k_1)$, we get a nice, neat, three-step computation:

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f(t_{i+1}, y_i + hk_1) \\ y_{i+1} &= y_i + \frac{h}{2}(k_1 + k_2). \end{aligned} \tag{6.3.3}$$

But before getting too carried away with the clean formulation, it would be nice to have some evidence that this "advanced" method gives a reasonable approximation of the solution to an o.d.e. as expected. Let's have the computer compute approximate solutions of o.d.e. 6.2.1 using both Euler's method and this method based on the trapezoidal rule, and compare them to the exact solution, $y(t) = \frac{t^3}{4} + \frac{16}{t}$. The following code snippet, while specific to this one task can be generalized to find approximate solutions of other o.d.e.s as well.

O.D.E. solver test code

```
t=4;
h=-1/4;
f=inline("-y/t+t^2");
exact=inline("t^3/4+16/t");
euler=20;
trap=20;
disp('             Euler   Trapezoid       Exact   Euler err    Trap err')
disp('        -------------------------------------------------------------')
for i=1:8
  euler=euler+h*f(t,euler);
  k1=f(t,trap);
  k2=f(t+h,trap+h*k1);
  trap=trap+h/2*(k1+k2);
  t=t+h;
  x=exact(t);
  sprintf('%12.5g%12.5g%12.5g%12.5g%12.5g',euler,trap,x,abs(euler-x),abs(trap-x))
end%for
```

This test code may be downloaded at the companion website (`rungeKuttaDemo.m`). The only part of this code that may appear unfamiliar to you at this point is the `sprintf()` command. The first argument,

`'%12.5g%12.5g%12.5g%12.5g%12.5g',`

is the formatting string. This particular string means to string together 5 floating point numbers using 12 spaces each and displaying 5 significant digits. In the `sprintf` command, `%12.5g` means "general" formatting of a floating point number with 12 spaces and 5 significant figures. The computer will decide whether to use scientific notation in the output. Since it is repeated 5 times, this particular command will format five such floating point values. The rest of the arguments are the five numbers to print. The command `sprintf` should not be read as "sprint-eff" but rather "ess-print-eff" or "string print formatted". The `s` is for string and the `f` is for formatted. If you're thinking this command seems a bit arcane, you're right. This type of print formatting command originated in the C programming language during the 1970s![1] The output of running this code is

```
             Euler   Trapezoid       Exact   Euler err    Trap err
        -------------------------------------------------------------
ans =        17.25      17.442       17.45     0.20026   0.0080729
ans =       14.884      15.273       15.29      0.4058    0.016741
ans =       12.885      13.479      13.505     0.62006    0.026142
ans =       11.236      12.047      12.083     0.84776    0.036458
ans =       9.9219      10.969      11.017      1.0955     0.04794
ans =       8.9332      10.245      10.306       1.373    0.060938
ans =       8.2641      9.8828      9.9588      1.6947    0.075955
ans =       7.9167      9.9062          10      2.0833     0.09375
```

Our method based on the trapezoidal rule, which we will call trapezoidal-ode for now, seems to do a better job of approximating the solution of this o.d.e. than does Euler's method. The last two columns contain the absolute errors for each approximation. The errors in trapeziodal-ode are roughly 0.01 to 0.1 while the errors for Euler's method are roughly 0.2 to 2. All of the errors in trapezoidal-ode are smaller than all the errors in Euler's method. Of course trapezoidal-ode requires two evaluations of f per step, so it better deliver better results for the extra work if it is to be useful at all.

Buoyed by this success, perhaps it is worth investing some time in other integration formulas, like Simpson's rule, for example. Recall from section 4.3, Simpson's rule states

$$\int_{x_0}^{x_0+2h} f(x)dx = \frac{h}{3}\left[f(x_0) + 4f(x_0+h) + f(x_0+2h)\right] + O(h^5 f^{(4)}(\xi_h)),$$

[1] See `https://en.wikipedia.org/wiki/Printf_format_string` for some details.

which in the notation of this section we might write as

$$\int_{t_i}^{t_{i+1}} f(t,y)\,dt = \frac{h}{6}\left[f(t_i,y_i) + 4f(t_{i+1/2},y_{i+1/2}) + f(t_{i+1},y_{i+1})\right],$$

ignoring the error term, and using the notation $t_{i+1/2}$ to mean $t_i + \frac{1}{2}h$ and $y_{i+1/2}$ to mean $y(t_i + \frac{1}{2}h)$. So an o.d.e. solver based on Simpson's rule might look like

$$y_{i+1} = y_i + \frac{h}{6}\left[f(t_i,y_i) + 4f(t_{i+1/2},y_{i+1/2}) + f(t_{i+1},y_{i+1})\right].$$

Again, this is an implicit formula. Again, we can use Euler's method to estimate y_{i+1}, and, in fact, we can use Euler's method to estimate $y_{i+1/2}$ too! Since $t_{i+1/2}$ is closer to t_i than is t_{i+1}, we estimate $y_{i+1/2}$ first. That is, we replace $y_{i+1/2}$ by $y_i + \frac{h}{2}f(t_i,y_i)$. Using a multiple-step calculation as before, that gives us

$$\begin{aligned} k_1 &= f(t_i,y_i) \\ k_2 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \end{aligned}$$

so far. This takes care of the first two terms in brackets. Now we estimate y_{i+1} by approximating $f(t_{i+1},y_{i+1})$. But we now have an estimate of f at $t_i + \frac{h}{2}$, and $t_i + \frac{h}{2}$ is closer to t_{i+1} than is t_i. So, even though we could use $y_i + hf(t_i,y_i) = y_i + hk_1$ to approximate y_{i+1} (as done before), we might expect $y_i + hk_2$ to be a better estimate. With this hope in hand, we complete the method by calculating as follows:

$$\begin{aligned} k_1 &= f(t_i,y_i) \\ k_2 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \\ k_3 &= f(t_{i+1}, y_i + hk_2) \\ y_{i+1} &= y_i + \frac{h}{6}\left[k_1 + 4k_2 + k_3\right]. \end{aligned}$$

For now, we will refer to this method as Simpson's-ode.

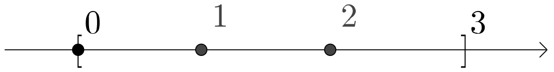
Before trying to assess whether this new method is better than the previous ones, let's derive a couple more, and compare them all together. The formula

$$\int_{x_0}^{x_0+3h} f(x)dx = \frac{3h}{2}\left[f(x_0+h) + f(x_0+2h)\right] + O(h^3 f''(\xi_h))$$

(an open Newton-Cotes formula from section 4.3) leads to the method

$$\begin{aligned} k_1 &= f(t_i,y_i) \\ k_2 &= f\left(t_i + \frac{h}{3}, y_i + \frac{h}{3}k_1\right) \\ k_3 &= f\left(t_i + \frac{2h}{3}, y_i + \frac{2h}{3}k_2\right) \\ y_{i+1} &= y_i + \frac{h}{2}\left[k_2 + k_3\right]. \end{aligned}$$

Can you fill in the steps to derive this method? Answer on page 193. We will call this method open-ode. Finally, we use the stencil



to derive yet another integration formula. This is not an open Newton-Cotes formula nor is it a closed Newton-Cotes formula. It is not one that was covered in section 4.3. Perhaps it might be called a "clopen" (half closed and half open) Newton-Cotes formula. Can you derive the corresponding integration method? Details on page 194. The result is

$$\int_{x_0}^{x_0+3h} f(x)dx \approx \frac{3h}{4}\left[f(x_0) + 3f(x_0+2h)\right],$$

disregarding the error term. This leads to the o.d.e. solver

$$\begin{aligned}
k_1 &= f(t_i, y_i)\\
k_2 &= f\left(t_i + \frac{h}{3}, y_i + \frac{h}{3}k_1\right)\\
k_3 &= f\left(t_i + \frac{2h}{3}, y_i + \frac{2h}{3}k_2\right)\\
y_{i+1} &= y_i + \frac{h}{4}\left[k_1 + 3k_3\right].
\end{aligned}$$

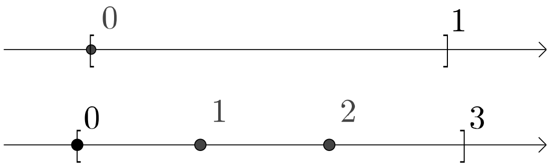
We will call this method clopen-ode. Notice two things. First, even though k_2 is not used in the final line, it is still computed since it is used to compute k_3. Second, the calculations of k_1, k_2, and k_3 are identical to those in the open-ode method. The only difference is how the k_j are combined. The integration methods combine the values of the function at the nodes differently. This idea of using the same k_j for different purposes will come up again!.

So now we have three new methods to test out—one based on Simpson's rule (Simpson's-ode), one based on an open Newton-Cotes formula (open-ode), and a third based on a "clopen" Newton-Cotes formula (clopen-ode). Can you write test code for comparing the three new formulas (similar to the code used to compare Euler's method with trapezoidal-ode)? Answer on page 195. Results are summarized in the following output:

```
        Simpsons  Open      Clopen    Simp err  Open err  Clop err
        ------------------------------------------------------------
ans =   17.44806  17.44999  17.45022   0.00220   0.00028   0.00004
ans =   15.28557  15.28953  15.29008   0.00461   0.00065   0.00010
ans =   13.49781  13.50395  13.50494   0.00730   0.00116   0.00017
ans =   12.07297  12.08146  12.08307   0.01036   0.00187   0.00027
ans =   11.00347  11.01450  11.01700   0.01393   0.00290   0.00040
ans =   10.28804  10.30185  10.30566   0.01821   0.00440   0.00059
ans =    9.93523   9.95208   9.95789   0.02354   0.00669   0.00088
ans =    9.96952   9.98969   9.99866   0.03048   0.01031   0.00134
```

Simpson's-ode does the poorest job of finding an approximate solution and clopen-ode does the best. But why?

We've done a pretty thorough job of sweeping error analysis under the rug up until now. The bulk of that investigation will happen in the next section, but we can do a quick analysis here. From section 4.3, we know that the trapezoidal rule and the open Newton-Cotes formula we used here both have error terms of $O(h^3)$, while Simpson's rule has error term $O(h^5)$. The integration methods based on the stencils



(which led to Euler's method and the clopen method) have yet undetermined error terms. Can you show that their error terms are $O(h^2)$ and $O(h^4)$, respectively? Answer on page 195. Based on the error terms of the underlying integration methods, we should expect these o.d.e. solvers to be, in order from least accurate to most accurate, Euler's method (based on a $O(h^2)$ integration formula), open-ode (based on a $O(h^3)$ integration formula), clopen-ode (based on a $O(h^4)$ integration formula), and Simpson's-ode (based on a $O(h^5)$ integration formula); with trapezoidal-ode to be on par with open-ode. Table 6.3 shows the errors in calculating $y(2)$ for 6.2.1 for the five methods of this section using various values of h. Since the value of h in each row is half that of the previous row, we would expect the ratio of the errors in consecutive rows to be approximately $\left(\frac{1}{2}\right)^\ell$ where the rate of convergence for the method is $O(h^\ell)$. For Euler's method, dividing the error in row 3 by that of row 2, we get $\left(\frac{1}{2}\right)^\ell \approx \frac{.55114}{1.0809} \approx \frac{1}{2}$ and dividing the error in row 6 by that in row 5, we get $\left(\frac{1}{2}\right)^\ell \approx \frac{.07013}{.1399} \approx \frac{1}{2}$, for example. This evidence suggests that $\ell = 1$ for Euler's method, and therefore, Euler's method has an $O(h)$ convergence. Repeating the same calculation for the other methods yields Table 6.4.

With the exception of Simpson's-ode, Table 6.4 suggests that o.d.e. solvers have an error term of one less degree than their underlying (single step) integration formula. In section 4.4 we noted that composite integration formulas also have error terms of one less degree than their corresponding single-step integration formulas (and we made a

Table 6.3: A comparison of absolute errors for five o.d.e. solvers

h	Euler's	Trap-ode	Open-ode	Clopen-ode	Simpson's-ode
$-\frac{1}{4}$	2.0833	0.09375	0.010311	0.0013444	0.030482
$-\frac{1}{8}$	1.0809	0.023437	0.0025929	0.00017446	0.0077168
$-\frac{1}{16}$	0.55114	0.0058594	0.00064977	$2.2207(10)^{-5}$	0.0019412
$-\frac{1}{32}$	0.27837	0.0014648	0.00016261	$2.8008(10)^{-6}$	0.00048679
$-\frac{1}{64}$	0.1399	0.00036621	$4.0672(10)^{-5}$	$3.5166(10)^{-7}$	0.00012188
$-\frac{1}{128}$	0.07013	$9.1553(10)^{-5}$	$1.017(10)^{-5}$	$4.4055(10)^{-8}$	$3.0494(10)^{-5}$

Table 6.4: The error terms of five o.d.e solvers and their underlying integration methods

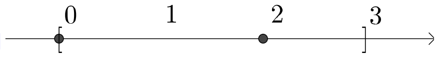
	Euler's	Trap-ode	Open-ode	Clopen-ode	Simpson's-ode
Integration method	$O(h^2)$	$O(h^3)$	$O(h^3)$	$O(h^4)$	$O(h^5)$
O.D.E. solver	$O(h)$	$O(h^2)$	$O(h^2)$	$O(h^3)$	$O(h^2)$

similar observation about Taylor methods in section 6.2). There is reason to believe in this parallel as the methods proposed in this section are essentially composite integration techniques. So, it should be a little troubling that Simpson's-ode does not fit the pattern. A deeper exploration of the error term is needed to explain this anomaly.

Exercises

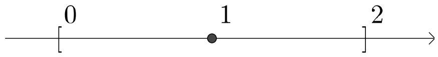
1. Derive an o.d.e. solver based on the stencil and corresponding integration formula.

(a) [S]



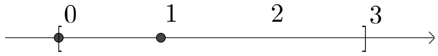
$$\frac{h}{4}\left(f(x_0)+3f\left(x_0+\frac{2}{3}h\right)\right)+O(h^4)$$

(b) [A]



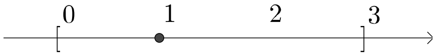
$$hf\left(x_0+\frac{1}{2}h\right)+O(h^3)$$

(c) [A]



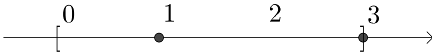
$$\frac{h}{2}\left(3f\left(x_0+\frac{1}{3}h\right)-f(x_0)\right)+O(h^3)$$

(d)



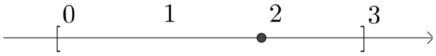
$$hf\left(x_0+\frac{1}{3}h\right)+O(h^2)$$

(e) [S]



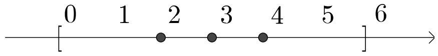
$$\frac{h}{4}\left(3f\left(x_0+\frac{1}{3}h\right)+f(x_0+h)\right)+O(h^4)$$

(f)



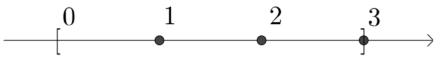
$$hf\left(x_0+\frac{2}{3}h\right)+O(h^2)$$

(g) [A]



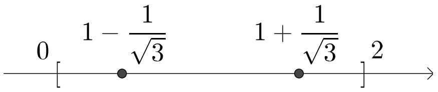
$$\frac{h}{2}\left(3f\left(x_0+\frac{1}{3}h\right)-4f\left(x_0+\frac{1}{2}h\right)+3hf\left(x_0+\frac{2}{3}h\right)\right)+O(h^5)$$

(h)



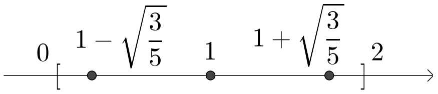
$$\frac{h}{4}\left(3f\left(x_0+\frac{1}{3}h\right)+f(x_0+h)\right)+O(h^4)$$

(i)



$$\frac{h}{2}\left(f\left(x_0+\frac{\sqrt{3}-1}{2\sqrt{3}}h\right)+f\left(x_0+\frac{\sqrt{3}+1}{2\sqrt{3}}h\right)\right)+O(h^5)$$

(j) [A]



$$\frac{h}{18}\left(5f\left(x_0+\frac{\sqrt{5}-\sqrt{3}}{2\sqrt{5}}h\right)+8f\left(x_0+\frac{1}{2}h\right)5f\left(x_0+\frac{\sqrt{5}+\sqrt{3}}{2\sqrt{5}}h\right)\right)+O(h^7)$$

2. Conduct a numerical experiment on test o.d.e. 6.2.1 to determine the rate of convergence of the method derived in question 1. Based on the error term of the integration formula, is the rate of convergence of the o.d.e. solver as expected?
3. Write computer code that implements Euler's method. [A]
4. Write computer code that implements trapezoidal-ode.
5. Write computer code that implements clopen-ode.
6. Write computer code that implements the solver you derived in exercise 1b. This is called the midpoint method or the modified Euler method. It is based on the midpoint rule for integration. [A]
7. Write computer code that implements the solver you derived in exercise 1a. This is called Ralston's method. [A]
8. Use your code from exercise 3 to compute $y(2)$ for the o.d.e. in exercise 1 on page 185 using step size $h = 0.05$. [S] [A]
9. Use your code from exercise 4 to compute $y(2)$ for the o.d.e. in exercise 1 on page 185 using step size $h = 0.05$. [S] [A]
10. Use your code from exercise 5 to compute $y(2)$ for the o.d.e. in exercise 1 on page 185 using step size $h = 0.05$. [S] [A]
11. Use your code from exercise 6 to compute $y(2)$ for the o.d.e. in exercise 1 on page 185 using step size $h = 0.05$. [S] [A]
12. Use your code from exercise 7 to compute $y(2)$ for the o.d.e. in exercise 1 on page 185 using step size $h = 0.05$. [S] [A]

Answers

Filling in the gaps: Beginning with the integration formula

$$\int_{x_0}^{x_0+3h} f(x)dx = \frac{3h}{2}\left[f(x_0+h)+f(x_0+2h)\right]+O(h^3f''(\xi_h)),$$

we "shrink" the interval of integration to $[x_0, x_0+s]$ by making the substitution $s = 3h$:

$$\int_{x_0}^{x_0+s} f(x)dx = \frac{s}{2}\left[f(x_0+\frac{1}{3}s)+f(x_0+\frac{2}{3}s)\right]+O(s^3f''(\xi_k)).$$

With the integration formula rephrased in terms of step size s, the o.d.e. solving method is

$$y_{i+1} = y_i + \frac{h}{2}\left[f(t_{i+1/3}, y_{i+1/3}) + f(t_{i+2/3}, y_{i+2/3})\right],$$

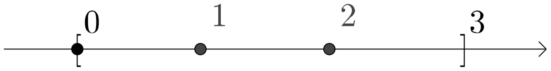
where we revert to using h for step size. We then use Euler's method to estimate $y_{i+1/3}$ and $y_{i+2/3}$, starting with $y_{i+1/3}$. That is, we replace $y_{i+1/3}$ by $y_i + \frac{h}{3}f(t_i, y_i)$. Then we estimate $y_{i+2/3}$. Using a multiple-step calculation as before, that gives us

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f\left(t_i + \frac{h}{3}, y_i + \frac{h}{3}k_1\right), \end{aligned}$$

taking care of the first term in brackets. It remains to estimate $f(t_{i+2/3}, y_{i+2/3})$. But we now have an estimate of f (the derivative of y) at $t_i + \frac{h}{3}$, and $t_i + \frac{h}{3}$ is closer to $t_{i+2/3}$ than is t_i. So, we approximate $y_{i+2/3}$ by $y_i + \frac{2}{3}hk_2$:

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f\left(t_i + \frac{h}{3}, y_i + \frac{h}{3}k_1\right) \\ k_3 &= f(t_i + \frac{2h}{3}, y_i + \frac{2h}{3}k_2) \\ y_{i+1} &= y_i + \frac{h}{2}\left[k_2 + k_3\right]. \end{aligned}$$

Clopen Newton-Cotes:



For this stencil, $a = x_0$, $b = x_0 + 3h$, and $\theta_i = ih$, $i = 0, 1, 2$. Therefore, we will have a system of three equations in the three unknowns. First, the left-hand sides:

$$\begin{aligned} \int_a^b p_0(x)dx = \int_{x_0}^{x_0+3h} p_0(x)dx &= \int_{x_0}^{x_0+3h} 1dx = (x - x_0)|_{x_0}^{x_0+3h} = 3h \\ \int_a^b p_1(x)dx = \int_{x_0}^{x_0+3h} p_1(x)dx &= \int_{x_0}^{x_0+3h} (x - x_0)dx = \frac{1}{2}(x - x_0)^2\Big|_{x_0}^{x_0+3h} = \frac{9}{2}h^2 \\ \int_a^b p_2(x)dx = \int_{x_0}^{x_0+3h} p_2(x)dx &= \int_{x_0}^{x_0+3h} (x - x_0)^2dx = \frac{1}{3}(x - x_0)^3\Big|_{x_0}^{x_0+3h} = 9h^3 \end{aligned}$$

Now putting them together with the right-hand sides (and swapping sides):

$$\begin{aligned} \sum_{i=0}^{2}(\theta_i h)^0 a_i &= a_0 + a_1 + a_2 = 3h \\ \sum_{i=0}^{2}(\theta_i h)^1 a_i &= ha_1 + 2ha_2 = \frac{9}{2}h^2 \\ \sum_{i=0}^{2}(\theta_i h)^2 a_i &= h^2a_1 + 4h^2a_2 = 9h^3 \end{aligned}$$

This system is small enough to solve by hand (without the use of a computer algebra system):

$$\begin{array}{ccccc} & h^2a_1 & +4h^2a_2 & = & 9h^3 \\ - & (h^2a_1 & +2h^2a_2 & = & \frac{9}{2}h^3) \\ \hline & & 2h^2a_2 & = & \frac{9}{2}h^3 \end{array} \Rightarrow a_2 = \tfrac{9}{4}h.$$

Substituting $a_2 = \frac{9}{4}h$ into $ha_1 + 2ha_2 = \frac{9}{2}h^2$, we can solve for a_1:

$$\begin{aligned} ha_1 + 2h \cdot \frac{9}{4}h &= \frac{9}{2}h^2 \\ ha_1 + \frac{9}{2}h^2 &= \frac{9}{2}h^2 \qquad \Rightarrow a_1 = 0. \\ ha_1 &= 0 \end{aligned}$$

Substituting $a_1 = 0$ and $a_2 = \frac{9}{4}h$ into $a_0 + a_1 + a_2 = 3h$, we can solve for a_0:

$$\begin{aligned} a_0 + 0 + \frac{9}{4}h &= 3h \\ a_0 &= 3h - \frac{9}{4}h \quad \Rightarrow a_0 = \frac{3}{4}h. \end{aligned}$$

Therefore, $\sum_{i=0}^{2} a_i f(x_0 + \theta_i h) = \frac{3}{4}h \cdot f(x_0) + 0 \cdot f(x_0 + h) + \frac{9}{4}h \cdot f(x_0 + 2h)$ and the integration formula is

$$\int_{x_0}^{x_0+3h} f(x)dx \approx \frac{3h}{4}\left[f(x_0) + 3f(x_0 + 2h)\right].$$

Test code: Comparing Simpson's, open, and clopen methods:

```
t=4;
h=-1/4;
f=inline("-y/t+t^2");
exact=inline("t^3/4+16/t");
simp=20;
open=20;
clop=20;
disp('        Simpsons  Open      Clopen    Simp err  Open err  Clop err')
disp('        ------------------------------------------------------------')
for i=1:8
  k1simp=f(t,simp);
  k1open=f(t,open);
  k1clop=f(t,clop);
  k2simp=f(t+h/2,simp+h/2*k1simp);
  k2open=f(t+h/3,open+h/3*k1open);
  k2clop=f(t+h/3,clop+h/3*k1clop);
  k3simp=f(t+h,simp+h*k2simp);
  k3open=f(t+2*h/3,open+2*h/3*k2open);
  k3clop=f(t+2*h/3,clop+2*h/3*k2clop);
  simp=simp+h/6*(k1simp+4*k2simp+k3simp);
  open=open+h/2*(k2open+k3open);
  clop=clop+h/4*(k1clop+3*k3clop);
  t=t+h;
  x=exact(t);
  sierr=abs(simp-x);
  operr=abs(open-x);
  clerr=abs(clop-x);
  sprintf('%12.5g%12.5g%12.5g%12.5g%12.5g%12.5g',simp,open,clop,sierr,operr,clerr)
end%for
```

This test code may be downloaded at the companion website (`rungeKuttaDemo2.m`).

Error terms: The error term for

$$\int_{x_0}^{x_0+3h} f(x)dx \approx \frac{3h}{4}\left[f(x_0) + 3f(x_0 + 2h)\right]$$

is derived in the section 4.3 solutions. See page **??**. The error term for

$$\int_{x_0}^{x_0+h} f(x)dx \approx hf(x_0)$$

is derived similarly. We are given that the error is $O(h^2)$, so we can skip the discovery. Expanding $f(x)$ in a Taylor polynomial with error term,

$$f(x) = f(x_0) + (x - x_0)f'(\xi_x).$$

So

$$\begin{aligned}\int_{x_0}^{x_0+h} f(x)dx - hf(x_0) &= \int_{x_0}^{x_0+h} \left(f(x_0) + (x-x_0)f'(\xi_x)\right) dx - hf(x_0) \\ &= xf(x_0)|_{x_0}^{x_0+h} + \int_{x_0}^{x_0+h} (x-x_0)f'(\xi_x)dx - hf(x_0) \\ &= hf(x_0) + \int_{x_0}^{x_0+h} (x-x_0)f'(\xi_x)dx - hf(x_0) \\ &= \int_{x_0}^{x_0+h} (x-x_0)f'(\xi_x)dx.\end{aligned}$$

By the weighted mean value theorem, there exists $c \in (x_0, x_0+h)$ such that $\int_{x_0}^{x_0+h}(x-x_0)f'(\xi_x)dx = f'(c)\int_{x_0}^{x_0+h}(x-x_0)dx = \frac{1}{2}f'(c)h^2$. Hence

$$\int_{x_0}^{x_0+h} f(x)dx - hf(x_0) = \frac{1}{2}f'(c)h^2 \leq Mh^2 f'(\xi_h)$$

where we have replaced c by ξ_h.

6.4 Error Analysis

Section 6.3 ended with the mysterious (and unsettling?) observation that Simpson's-ode did not live up to expectations. Based on other o.d.e. solvers, we would expect the rate of convergence of Simpson's-ode to be $O(h^4)$ since Simpson's rule, on which Simpson's-ode is based, has local truncation error $O(h^5)$.

The explanation is rooted in the fact that we are solving an o.d.e. of the form $\dot{y} = f(t, y)$, in which the derivative is a function of two variables, t and y. To understand the error analysis, heavy use of partial derivatives and the chain rule are required. As ever, we consult Taylor's theorem and write

$$y(t_0 + h) = y(t_0) + h\dot{y}(t_0) + \frac{1}{2}h^2\ddot{y}(t_0) + \frac{1}{6}h^3\dddot{y}(t_0) + \cdots .$$

Each derivative of y can be replaced by some function of f and its partial derivatives, starting with $\dot{y}$, which is given by the o.d.e. we are trying to solve.

$$\begin{aligned} \dot{y} &= f(t,y) \\ \ddot{y} = \frac{d}{dt}\dot{y} &= \frac{d}{dt}f(t,y) = f_t(t,y) + f_y(t,y)\dot{y} = f_t(t,y) + f_y(t,y)\cdot f(t,y) \\ &\vdots \end{aligned}$$

Eliminating the explicit use of arguments t and y,

$$\begin{aligned} \dot{y} &= f \\ \ddot{y} &= f_t + f_y f \\ \dddot{y} &= f_{tt} + f_{ty}f + (f_{yt} + f_{yy}f)f + f_y(f_t + f_y f) \\ &= f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f \\ &\vdots \end{aligned}$$

so $y(t_0 + h) = y(t_0) + h\dot{y}(t_0) + \frac{1}{2}h^2\ddot{y}(t_0) + \frac{1}{6}h^3\dddot{y}(t_0) + \cdots$ in terms of f is

$$y(t_0 + h) = y(t_0) + hf + \frac{1}{2}h^2(f_t + f_y f) + \frac{1}{6}h^3(f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f) + \cdots ,$$

and as an o.d.e. solver (replacing $y(t_0)$ by y_i and $y(t_0 + h)$ by y_{i+1}),

$$y_{i+1} = y_i + hf + \frac{1}{2}h^2(f_t + f_y f) + \frac{1}{6}h^3(f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f) + \cdots . \tag{6.4.1}$$

Rewriting high degree Taylor polynomials in terms of f quickly becomes complicated. We will focus on analysis requiring only $\dot{y}$, $\ddot{y}$, and $\dddot{y}$.

The o.d.e. solvers of section 6.3 have the form

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f\left(t_i + \beta_2 h, y_i + \beta_2 h k_1\right) \\ k_3 &= f(t_i + \beta_3 h, y_i + \beta_3 h k_2) \\ &\vdots \\ k_s &= f(t_i + \beta_s h, y_i + \beta_s h k_{s-1}) \\ y_{i+1} &= y_i + h\left[\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \cdots + \alpha_s k_s\right] . \end{aligned} \tag{6.4.2}$$

We did not actually see any o.d.e. solvers with $s > 3$ in section 6.3, but the process we followed would clearly require it should there be more than three nodes in the underlying integration formula.

The difference between $y(t_0 + h)$ from (6.4.1) and y_{i+1} from (6.4.2) is the local truncation error of the o.d.e. solver (the error in taking a single step). In order to write this truncation error in the form $O(h^\ell)$, though, we need to expand each k_j in its Taylor polynomial. Taylor's theorem in two variables is needed.

Theorem 8. *Suppose $f(t,y)$ and all its partial derivatives of order $n+1$ and lower are continuous on the rectangle $D = \{(t,y) : a \leq t \leq b, c \leq y \leq d\}$, and let $(t_0, y_0) \in D$. Then for every $(t,y) \in D$, there exist $\xi \in (a,b)$ and $\mu \in (c,d)$ such that*

$$\begin{aligned} f(t,y) &= f(t_0,y_0) + [(t-t_0)\cdot f_t(t_0,y_0) + (y-y_0)\cdot f_y(t_0,y_0)] \\ &\quad + \frac{1}{2}\left[(t-t_0)^2 f_{tt}(t_0,y_0) + 2(t-t_0)(y-y_0)\cdot f_{ty}(t_0,y_0) + (y-y_0)^2 f_{yy}(t_0,y_0)\right] \\ &\quad + \cdots + \\ &\quad \frac{1}{n!}\left[\sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array}\right)(t-t_0)^{n-j}(y-y_0)^j \frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0,y_0)\right] \\ &\quad + \frac{1}{(n+1)!}\left[\sum_{j=0}^{n+1} \left(\begin{array}{c} n+1 \\ j \end{array}\right)(t-t_0)^{n+1-j}(y-y_0)^j \frac{\partial^{n+1} f}{\partial t^{n+1-j}\partial y^j}(\xi,\mu)\right]. \end{aligned}$$

As with Taylor's theorem (of one variable), the first $n+1$ terms form the Taylor polynomial and the last term is the remainder term.

To illustrate, we let $f(t,y) = -\frac{y}{t} + t^2$ and compute its second Taylor polynomial with remainder term expanded about $(t_0, y_0) = (1,1)$. For this, we will need all partial derivatives of f up to and including order 3.

$$\begin{aligned} f_t &= \frac{y}{t^2} + 2t \\ f_y &= -\frac{1}{t} \\ f_{tt} &= -2\frac{y}{t^3} + 2 \\ f_{ty} = f_{yt} &= \frac{1}{t^2} \\ f_{yy} &= 0 \\ f_{ttt} &= 6\frac{y}{t^4} \\ f_{tty} = f_{tyt} = f_{ytt} &= -\frac{2}{t^3} \\ f_{tyy} = f_{yty} = f_{yyt} &= 0 \\ f_{yyy} &= 0. \end{aligned}$$

It follows that

$$\begin{aligned} f(1,1) &= 0 \\ f_t(1,1) &= 3 \\ f_y(1,1) &= -1 \\ f_{tt}(1,1) &= 0 \\ f_{ty}(1,1) &= 1 \\ f_{yy}(1,1) &= 0 \\ f_{ttt}(\xi,\mu) &= 6\frac{\mu}{\xi^4} \\ f_{tty}(\xi,\mu) &= -\frac{2}{\xi^3} \\ f_{tyy}(\xi,\mu) &= 0 \\ f_{yyy}(\xi,\mu) &= 0. \end{aligned}$$

Therefore, the second Taylor polynomial for $f(t,y)$ is

$$\begin{aligned} T_2(t,y) &= f(1,1) + [(t-1)\cdot f_t(1,1) + (y-1)\cdot f_y(1,1)] \\ &\quad + \frac{1}{2}\left[(t-1)^2 f_{tt}(1,1) + 2(t-1)(y-1)\cdot f_{ty}(1,1) + (y-1)^2 f_{yy}(1,1)\right] \\ &= 0 + 3(t-1) - (y-1) + 0(t-1)^2 + (t-1)(y-1) + 0(y-1)^2 \\ &= 3(t-1) - (y-1) + (t-1)(y-1) \end{aligned}$$

with remainder term

$$\begin{aligned}
R_2(t,y) &= \frac{1}{6}\left[(t-1)^3 f_{ttt}(\xi,\mu) + 3(t-1)^2(y-1)f_{tty}(\xi,\mu) + 3(t-1)(y-1)^2 f_{tyy}(\xi,\mu) + (y-1)^3 f_{yyy}(\xi,\mu)\right] \\
&= \frac{1}{6}\left[(t-1)^3 \cdot 6\frac{\mu}{\xi^4} - 3(t-1)^2(y-1)\cdot\frac{2}{\xi^3} + 3(t-1)(y-1)^2\cdot 0 + (y-1)^3\cdot 0\right] \\
&= (t-1)^3\frac{\mu}{\xi^4} - (t-1)^2(y-1)\frac{1}{\xi^3}.
\end{aligned}$$

More generally, suppose we are interested in Taylor polynomial expansions of expressions like $f(t_i + \beta_j h, y_i + \beta_j h k_{j-1})$, as we have in our o.d.e. solvers. Expanding about (t_i, y_i), we let $t_0 = t_i$, $y_0 = y_i$, $t = t_i + \beta_j h$, and $y = y_i + \beta_j h k_{j-1}$. Thus $t - t_0 = \beta_j h$ and $y - y_0 = \beta_j h k_{j-1}$, and the second Taylor polynomial without explicit listing of the arguments t_i and y_i on the right-hand side is

$$f(t_i + \beta_j h, y_i + \beta_j h k_{j-1}) = f + h\beta_j\left[f_t + k_{j-1}f_y\right] + \frac{1}{2}h^2\beta_j^2\left[f_{tt} + 2k_{j-1}f_{ty} + k_{j-1}^2 f_{yy}\right]$$

with remainder term $O(h^3)$.

In particular, when we set $j = 1$, $\beta_j = \beta_1 = 0$, we get

$$k_1 = f(t_i, y_i) = f.$$

When we set $j = 2$,

$$\begin{aligned}
k_2 &= f\left(t_i + \beta_2 h, y_i + \beta_2 h k_1\right) \\
&= f + h\beta_2\left[f_t + f f_y\right] + \frac{1}{2}h^2\beta_2^2\left[f_{tt} + 2f f_{ty} + f^2 f_{yy}\right] + O(h^3).
\end{aligned}$$

The calculation of k_3 is a little bit messier since it involves k_2^2. Before diving in headlong, though, consider what we will do with k_3 first. After computing k_1, k_2, and k_3, we will substitute each into the formula

$$y_{i+1} = y_i + h\left[\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3\right] \tag{6.4.3}$$

and subtract the result from (6.4.1). For purposes of this discussion, we seek a method with local truncation error $O(h^4)$. Therefore, we need only retain constant terms and terms containing a factor of h^3, h^2, or h in equation (6.4.3). Terms with higher powers of h are irrelevant. They will be assumed (or should I say consumed?) by the $O(h^4)$. Since the sum $\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3$ is multiplied by h, we need only retain terms with factors of up to h^2 in k_1, k_2, and k_3. Taking a look at the expansion of k_3:

$$\begin{aligned}
k_3 &= f\left(t_i + \beta_3 h, y_i + \beta_3 h k_2\right) \\
&= f + h\beta_3\left[f_t + k_2 f_y\right] + \frac{1}{2}h^2\beta_3^2\left[f_{tt} + 2k_2 f_{ty} + k_2^2 f_{yy}\right]
\end{aligned}$$

we see only the term $\frac{1}{2}h^2\beta_3^2 \cdot k_2^2 f$ contains k_2^2, and it already has a factor of h^2. Consequently, we only need to include the constant term of k_2^2. The rest of the terms of k_2^2 become part of the $O(h^4)$. That's not so bad!

$$k_2^2 = f^2 + O(h).$$

Similarly, when we substitute expressions for k_2 into k_3, we will be careful to avoid any terms that would give a factor of h to any power greater than 2:

$$\begin{aligned}
k_3 &= f + h\beta_3\left[f_t + \left(f + h\beta_2\left[f_t + f f_y\right]\right) f_y\right] \\
&\quad + \frac{1}{2}h^2\beta_3^2\left[f_{tt} + 2(f)f_{ty} + \left(f^2\right) f_{yy}\right] + O(h^3) \\
&= f + h\beta_3 f_t + h\beta_3 f f_y + h^2\beta_2\beta_3(f_t f_y + f f_y^2) \\
&\quad + \frac{1}{2}h^2\beta_3^2\left[f_{tt} + 2f f_{ty} + f^2 f_{yy}\right] + O(h^3).
\end{aligned}$$

After all that detailed computation, now is a good time to lean back and take a look at what we have so far. We have expanded all the terms of (6.4.2) for $s = 3$ and are ready to compare the result to the Taylor expansion of the o.d.e. in (6.4.1). The difference of the two is the local truncation error, so we will be interested in the least power of h that remains after subtraction. Copying the two equations here for convenience, we are subtracting

$$y_{i+1} = y_i + hf + \frac{1}{2}h^2(f_t + f_y f) + \frac{1}{6}h^3(f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f) + O(h^4)$$

from

$$\begin{aligned}
y_{i+1} &= y_i + h\left[\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3\right] \\
&= y_i + h\alpha_1 k_1 + h\alpha_2 k_2 + h\alpha_3 k_3 \\
&= y_i + h\alpha_1 f \\
&\quad + h\alpha_2\left(f + h\beta_2\left[f_t + ff_y\right] + \frac{1}{2}h^2\beta_2^2\left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right] + O(h^3)\right) \\
&\quad + h\alpha_3\left(f + h\beta_3 f_t + h\beta_3 ff_y + h^2\beta_2\beta_3(f_t f_y + ff_y^2) + \frac{1}{2}h^2\beta_3^2\left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right] + O(h^3)\right).
\end{aligned}$$

The constant term (term containing no factor of h) for each equation is simply y_i, so no constant will remain after subtraction. The difference of the terms involving h is $hf - (h\alpha_1 f + h\alpha_2 f + h\alpha_3 f) = hf(1 - (\alpha_1 + \alpha_2 + \alpha))$, so if there is to be no h left in the difference, we must have

$$\alpha_1 + \alpha_2 + \alpha_3 = 1.$$

The difference of the terms involving $h^2 f_t$ is $\frac{1}{2}h^2 f_t - (h^2\alpha_2\beta_2 f_t + h^2\alpha_3\beta_3 f_t) = h^2 f_t(\frac{1}{2} - (\alpha_2\beta_2 + \alpha_3\beta_3))$, so if there is to be no $h^2 f_t$ left in the difference, we must have

$$\alpha_2\beta_2 + \alpha_3\beta_3 = \frac{1}{2}.$$

Similarly, we consider the differences of the rest of the terms to get the following conditions on the α_j and β_j.

term	leads to condition
$h^2 f_y f$	$\alpha_2\beta_2 + \alpha_3\beta_3 = \frac{1}{2}$
$h^3 f_{tt}$	$\alpha_2\beta_2^2 + \alpha_3\beta_3^2 = \frac{1}{3}$
$h^3 f_{ty} f$	$\alpha_2\beta_2^2 + \alpha_3\beta_3^2 = \frac{1}{3}$
$h^3 f_{yy} f^2$	$\alpha_2\beta_2^2 + \alpha_3\beta_3^2 = \frac{1}{3}$
$h^3 f_t f_y$	$\alpha_3\beta_2\beta_3 = \frac{1}{6}$
$h^3 f_y^2 f$	$\alpha_3\beta_2\beta_3 = \frac{1}{6}$

We have considered all 8 different terms, but have only arrived at 4 distinct conditions:

$$\begin{aligned}
\alpha_1 + \alpha_2 + \alpha_3 &= 1 \\
\alpha_2\beta_2 + \alpha_3\beta_3 &= \frac{1}{2} \\
\alpha_2\beta_2^2 + \alpha_3\beta_3^2 &= \frac{1}{3} \\
\alpha_3\beta_2\beta_3 &= \frac{1}{6}.
\end{aligned} \tag{6.4.4}$$

Since we have 5 variables and only 4 conditions, we should think that there are multiple o.d.e. solvers of the form (6.4.2) with $s = 3$ and local truncation error $O(h^4)$.

Evidence from section 6.3 suggests that clopen-ode should have local truncation error $O(h^4)$. Let's check. For that method, we have

$$\alpha_1 = \frac{1}{4}, \quad \alpha_2 = 0, \quad \alpha_3 = \frac{3}{4}$$
$$\beta_2 = \frac{1}{3}, \quad \beta_3 = \frac{2}{3},$$

so

$$\begin{aligned}
\alpha_1+\alpha_2+\alpha_3 &= \frac{1}{4}+0+\frac{3}{4}=1\\
\alpha_2\beta_2+\alpha_3\beta_3 &= 0\cdot\frac{1}{3}+\frac{3}{4}\cdot\frac{2}{3}=\frac{1}{2}\\
\alpha_2\beta_2^2+\alpha_3\beta_3^2 &= 0\left(\frac{1}{3}\right)^2+\frac{3}{4}\left(\frac{2}{3}\right)^2=\frac{1}{3}\\
\alpha_3\beta_2\beta_3 &= \frac{3}{4}\cdot\frac{1}{3}\cdot\frac{2}{3}=\frac{1}{6}.
\end{aligned}$$

Indeed, clopen-ode satisfies all the conditions of an o.d.e. solver with local truncation error (at least) $O(h^4)$. We would actually have to show that at least one term containing an h^4 remains in the difference to prove that the local truncation error is not of greater degree.

Before finally answering the question of what happened to Simpson's-ode, our hard work so far is sufficient to check that trapezoidal-ode and open-ode have local truncation error $O(h^3)$ and that Euler's method has local truncation error $O(h^2)$. For trapezoidal-ode, we have $\alpha_1=\frac{1}{2}$, $\alpha_2=\frac{1}{2}$, $\alpha_3=0$, $\beta_2=1$, and β_3 undefined (we may assign any particular number we choose since having $\alpha_3=0$ makes β_3 irrelevant to the method), which gives us

$$\begin{aligned}
\alpha_1+\alpha_2+\alpha_3 &= \frac{1}{2}+\frac{1}{2}+0=1\\
\alpha_2\beta_2+\alpha_3\beta_3 &= \frac{1}{2}\cdot 1+0=\frac{1}{2}\\
\alpha_2\beta_2^2+\alpha_3\beta_3^2 &= \frac{1}{2}\left(\frac{1}{3}\right)^2+0=\frac{1}{18}\neq\frac{1}{3}\\
\alpha_3\beta_2\beta_3 &= 0\neq\frac{1}{6}.
\end{aligned}$$

The first two conditions are satisfied, but the last two are not. Recall, though, that the first two conditions were derived from the h and h^2 terms while the last two conditions were derived from the h^3 terms. So, for trapezoidal-ode, the local truncation error is $O(h^3)$.

For Euler's method, we have $\alpha_1=1$, $\alpha_2=\alpha_3=0$, and β_2 and β_3 undefined (or whatever we choose), which gives us

$$\begin{aligned}
\alpha_1+\alpha_2+\alpha_3 &= 1+0+0=1\\
\alpha_2\beta_2+\alpha_3\beta_3 &= 0+0=0\neq\frac{1}{2}\\
\alpha_2\beta_2^2+\alpha_3\beta_3^2 &= 0+0=0\neq\frac{1}{3}\\
\alpha_3\beta_2\beta_3 &= 0\neq\frac{1}{6}.
\end{aligned}$$

The second equation, which was derived from terms involving h^2, is not satisfied but the first equation, which was derived from terms involving h, is, so the local truncation error for Euler's method is $O(h^2)$.

Finally, for Simpson's-ode, we have $\alpha_1=\frac{1}{6}$, $\alpha_2=\frac{2}{3}$, $\alpha_3=\frac{1}{6}$, $\beta_2=\frac{1}{2}$, and $\beta_3=1$, which gives us

$$\begin{aligned}
\alpha_1+\alpha_2+\alpha_3 &= \frac{1}{6}+\frac{2}{3}+\frac{1}{6}=1\\
\alpha_2\beta_2+\alpha_3\beta_3 &= \frac{2}{3}\cdot\frac{1}{2}+\frac{1}{6}\cdot 1=\frac{1}{2}\\
\alpha_2\beta_2^2+\alpha_3\beta_3^2 &= \frac{2}{3}\left(\frac{1}{2}\right)^2+\frac{1}{6}(1)^2=\frac{1}{3}\\
\alpha_3\beta_2\beta_3 &= \frac{1}{6}\cdot\frac{1}{2}\cdot 1\neq\frac{1}{6}.
\end{aligned}$$

The first two equations are satisfied, so the local truncation error is (at least) $O(h^3)$, but the last equation is not satisfied, so the local truncation error is no more than $O(h^3)$. No terms containing factors of h or h^2 (that don't also contain higher powers of h) appear in the local truncation error, but the term $h^3\alpha_3\beta_2\beta_3(f_tf_y+ff_y^2)=\frac{1}{6}h^3(f_tf_y+ff_y^2)$ does, so it is $O(h^3)$.

A Note About Convention and Practice

We have derived five o.d.e. solvers so far with little nod to established practice. It's time to fix that. What we have been calling trapezoidal-ode (since it was derived from the trapezoidal rule) is better known as the improved Euler method, though some will refer to it as the explicit trapezoidal method. What we have been calling clopen-ode is better known as Heun's third order method. These methods can easily be found in the literature. They are prototypical examples of efficient methods. The improved Euler method requires two function evaluations per step and gives a local truncation error $O(h^3)$. Heun's third order method requires three function evaluations per step and gives a local truncation error $O(h^4)$.

What we have been calling open-ode has not been named as it would never be used in practice. It is not an efficient method, requiring three function evaluations but having a local truncation error of only $O(h^3)$. Consequently, you are not likely to see it appear in the literature as it is not a useful method in practice. Heun's third order method or the improved Euler method would both be preferable to open-ode. Heun's third order method gives a smaller truncation error for the same amount of computation (three function evaluations) and the improved Euler's method gives the same truncation error for less computation (two function evaluations). Simpson's-ode has the same shortcomings as open-ode, and thus you are not likely to see it in the literature either. It is also an inefficient method.

Methods of the form (6.4.2) are part of a class of methods called Runge-Kutta methods, named after the German mathematicians Carl Runge and Martin Kutta. The basic idea for such methods was laid out by Runge in a paper published in 1895, where Runge introduced the improved Euler method and others. His work was continued by Heun, whose paper of 1900 brought us Heun's third order method and others. In 1901, Kutta derives the most famous Runge-Kutta method, what is sometimes now referred to as the classic Runge-Kutta method or the Runge-Kutta method of order 4, RK4. We will see shortly that it is a modification of Simpson's-ode.[7]

Higher Order Methods

Higher order Runge-Kutta methods can be derived by considering methods of the form (6.4.2) with a number of stages, $s > 3$. Of course higher order methods must satisfy more conditions. In fact, the number of conditions grows faster as the desired order increases than does the number of variables as the number of stages increases. In other words, there is a point where the number of stages to achieve order p exceeds p. Order 1 methods can be derived with one stage (Euler's method) and no less. Order 2 methods can be derived with two stages (improved Euler's method) and no less. Order 3 methods can be derived with three stages (Heun's third order method) and no less. Order 4 methods can be derived with four stages (example upcoming) and no less. However, order p methods with $p > 4$ require a number of stages $s > p$, which, in turn means more than p function evaluations. So, the most efficient methods are to be found with order 4 or less.

Simpson's-ode failed to live up to its potential because it did not have enough stages, not because there is no Simpson's-rule-derived formula with local truncation error $O(h^5)$. The classic Runge-Kutta method of order 4 (local truncation error $O(h^5)$) has four stages and is given by

$$\begin{aligned}
k_1 &= f(t_i, y_i)\\
k_2 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right)\\
k_3 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right)\\
k_4 &= f\left(t_i + h, y_i + hk_3\right)\\
y_{i+1} &= y_i + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right].
\end{aligned}$$

Compare this to Simpson's-ode:

$$\begin{aligned}
k_1 &= f(t_i, y_i)\\
k_2 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right)\\
k_3 &= f(t_{i+1}, y_i + hk_2)\\
y_{i+1} &= y_i + \frac{h}{6}\left[k_1 + 4k_2 + k_3\right].
\end{aligned}$$

They are very similar. If we separate the second stage of Simpson's-ode into two stages, we get Runge-Kutta's order 4 method. That is the difference. Two stages are used to approximate $\dot{y}(t_i + \frac{h}{2})$ instead of one!

Crumpet 33: Derivation of The (Classic) Runge-Kutta Order 4

To derive any Runge-Kutta method of order 4, the stages of the computation must be expanded in a third Taylor polynomial:

$$\begin{aligned} f(t_i+\beta_j h, y_i+\beta_j h k_{j-1}) &= f + h\beta_j\left[f_t + k_{j-1}f_y\right] + \frac{1}{2}h^2\beta_j^2\left[f_{tt} + 2k_{j-1}f_{ty} + k_{j-1}^2 f_{yy}\right] \\ &\quad + \frac{1}{6}h^3\beta_j^3\left[f_{ttt} + 3k_{j-1}f_{tty} + 3k_{j-1}^2 f_{tyy} + k_{j-1}^3 f_{yyy}\right] + O(h^4) \end{aligned}$$

and $f(t_0, y_0)$ must be expanded in a fourth Taylor polynomial:

$$y(t_0+h) = y(t_0) + h\dot{y}(t_0) + \frac{1}{2}h^2\ddot{y}(t_0) + \frac{1}{6}h^3\dddot{y}(t_0) + \frac{1}{24}h^4\ddddot{y}(t_0) + O(h^5).$$

But $\ddddot{y}$, in terms of f, is

$$\begin{aligned} \frac{d}{dt}(\dddot{y}) &= \frac{d}{dt}\left(f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f\right) \\ &= f_{yyy}f^3 + 3f_{tyy}f^2 + 4f_y f_{yy}f^2 + 3f_{tty}f + 5f_{ty}f_y f + f_y^3 f \\ &\quad +3f_t f_{yy}f + f_t f_y^2 + f_{tt}f_y + f_{ttt} + 3f_t f_{ty} \end{aligned}$$

so

$$\begin{aligned} y_{i+1} &= y_i + hf + \frac{1}{2}h^2(f_t + f_y f) + \frac{1}{6}h^3(f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f) \\ &\quad + \frac{1}{24}h^4\left(f_{yyy}f^3 + 3f_{tyy}f^2 + 4f_y f_{yy}f^2 + 3f_{tty}f + 5f_{ty}f_y f + f_y^3 f\right. \\ &\qquad \left. +3f_t f_{yy}f + f_t f_y^2 + f_{tt}f_y + f_{ttt} + 3f_t f_{ty}\right) + O(h^5). \end{aligned}$$

Furthermore,

$$k_1 = f(t_i, y_i) = f$$

and

$$\begin{aligned} k_2 &= f\left(t_i + \beta_2 h, y_i + \beta_2 h k_1\right) \\ &= f + h\beta_2\left[f_t + ff_y\right] + \frac{1}{2}h^2\beta_2^2\left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right] \\ &\quad + \frac{1}{6}h^3\beta_2^3\left[f_{ttt} + 3ff_{tty} + 3f^2 f_{tyy} + f^3 f_{yyy}\right] + O(h^4). \end{aligned}$$

Consequently, $k_2^2 = f^2 + 2h\beta_2\left[f_t + ff_y\right]f + O(h^2)$ and $k_2^3 = f^3 + O(h)$. Therefore

$$\begin{aligned} k_3 &= f + h\beta_3\left[f_t + k_2 f_y\right] + \frac{1}{2}h^2\beta_3^2\left[f_{tt} + 2k_2 f_{ty} + k_2^2 f_{yy}\right] \\ &\quad + \frac{1}{6}h^3\beta_3^3\left[f_{ttt} + 3k_2 f_{tty} + 3k_2^2 f_{tyy} + k_2^3 f_{yyy}\right] \\ &= f + h\beta_3\left[f_t + \left(f + h\beta_2\left[f_t + ff_y\right] + \frac{1}{2}h^2\beta_2^2\left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right]\right)f_y\right] \\ &\quad + \frac{1}{2}h^2\beta_3^2\left[f_{tt} + 2\left(f + h\beta_2\left[f_t + ff_y\right]\right)f_{ty} + \left(f^2 + 2h\beta_2\left[f_t + ff_y\right]f\right)f_{yy}\right] \\ &\quad + \frac{1}{6}h^3\beta_3^3\left[f_{ttt} + 3ff_{tty} + 3f^2 f_{tyy} + f^3 f_{yyy}\right] + O(h^4) \\ &= f + h\beta_3\left[f_t + ff_y\right] + h^2\beta_2\beta_3\left[f_t + ff_y\right]f_y + \frac{1}{2}h^2\beta_3^2\left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right] \\ &\quad + \frac{1}{2}h^3\beta_3\beta_2^2\left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right]f_y + h^3\beta_3^2\beta_2\left[f_t + ff_y\right]\left[f_{ty} + ff_{yy}\right] \\ &\quad + \frac{1}{6}h^3\beta_3^3\left[f_{ttt} + 3ff_{tty} + 3f^2 f_{tyy} + f^3 f_{yyy}\right] + O(h^4). \end{aligned}$$

So, $k_3^2 = f^2 + 2h\beta_3 \left[f_t + ff_y\right] f + O(h^2)$ and $k_3^3 = f^3 + O(h)$. Therefore

$$\begin{aligned}
k_4 &= f + h\beta_4 \left[f_t + k_3 f_y\right] + \frac{1}{2}h^2\beta_4^2 \left[f_{tt} + 2k_3 f_{ty} + k_3^2 f_{yy}\right] \\
&\quad + \frac{1}{6}h^3\beta_4^3 \left[f_{ttt} + 3k_3 f_{tty} + 3k_3^2 f_{tyy} + k_3^3 f_{yyy}\right] + O(h^4) \\
&= f + h\beta_4 \left[f_t + \left(f + h\beta_3 \left[f_t + ff_y\right] + h^2\beta_2\beta_3 \left[f_t + ff_y\right] f_y + \frac{1}{2}h^2\beta_3^2 \left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right]\right) f_y\right] \\
&\quad + \frac{1}{2}h^2\beta_4^2 \left[f_{tt} + 2\left(f + h\beta_3 \left[f_t + ff_y\right]\right) f_{ty} + \left(f^2 + 2h\beta_3 \left[f_t + ff_y\right] f\right) f_{yy}\right] \\
&\quad + \frac{1}{6}h^3\beta_4^3 \left[f_{ttt} + 3ff_{tty} + 3f^2 f_{tyy} + f^3 f_{yyy}\right] + O(h^4) \\
&= f + h\beta_4 \left[f_t + ff_y\right] + h^2\beta_3\beta_4 \left[f_t + ff_y\right] f_y + \frac{1}{2}h^2\beta_4^2 \left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right] \\
&\quad + h^3\beta_2\beta_3\beta_4 \left[f_t + ff_y\right] f_y^2 + \frac{1}{2}h^3\beta_4\beta_3^2 \left[f_{tt} + 2ff_{ty} + f^2 f_{yy}\right] f_y \\
&\quad + h^3\beta_4^2\beta_3 \left[f_t + ff_y\right]\left[f_{ty} + ff_{yy}\right] + \frac{1}{6}h^3\beta_4^3 \left[f_{ttt} + 3ff_{tty} + 3f^2 f_{tyy} + f^3 f_{yyy}\right] + O(h^4).
\end{aligned}$$

Matching coefficients in

$$\begin{aligned}
y_{i+1} &= y_i + hf + \frac{1}{2}h^2(f_t + f_y f) + \frac{1}{6}h^3(f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f) \\
&\quad + \frac{1}{24}h^4 \left(f_{yyy}f^3 + 3f_{tyy}f^2 + 4f_y f_{yy} f^2 + 3f_{tty}f + 5f_{ty}f_y f + f_y^3 f\right. \\
&\qquad \left. + 3f_t f_{yy} f + f_t f_y^2 + f_{tt} f_y + f_{ttt} + 3f_t f_{ty}\right) + O(h^5).
\end{aligned}$$

with coefficients in

$$y_{i+1} = y_i + h\left[\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \alpha_4 k_4\right]$$

up to order 4 yields the conditions

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1 \tag{6.4.5}$$

$$\alpha_2\beta_2 + \alpha_3\beta_3 + \alpha_4\beta_4 = \frac{1}{2} \tag{6.4.6}$$

$$\alpha_2\beta_2^2 + \alpha_3\beta_3^2 + \alpha_4\beta_4^2 = \frac{1}{3} \tag{6.4.7}$$

$$\alpha_3\beta_2\beta_3 + \alpha_4\beta_3\beta_4 = \frac{1}{6} \tag{6.4.8}$$

$$\alpha_2\beta_2^3 + \alpha_3\beta_3^3 + \alpha_4\beta_4^3 = \frac{1}{4} \tag{6.4.9}$$

$$\alpha_3\beta_3^2\beta_2 + \alpha_4\beta_4^2\beta_3 = \frac{1}{8} \tag{6.4.10}$$

$$2\alpha_3\beta_3^2\beta_2 + 2\alpha_4\beta_4^2\beta_3 + \alpha_3\beta_3\beta_2^2 + \alpha_4\beta_4\beta_3^2 = \frac{1}{3} \tag{6.4.11}$$

$$\alpha_3\beta_3^2\beta_2 + \alpha_4\beta_4^2\beta_3 + \alpha_3\beta_3\beta_2^2 + \alpha_4\beta_4\beta_3^2 = \frac{5}{24} \tag{6.4.12}$$

$$\alpha_3\beta_3\beta_2^2 + \alpha_4\beta_4\beta_3^2 = \frac{1}{12} \tag{6.4.13}$$

$$\alpha_4\beta_2\beta_3\beta_4 = \frac{1}{24}. \tag{6.4.14}$$

Any four-stage ($s = 4$) fourth order Runge-Kutta method of the form (6.4.2) will have to satisfy these 10 equations with only 7 degrees of freedom (7 variables). Either the equations form a dependent set or solutions will be rare. In an attempt to solve the system, we solve (6.4.14) for α_4:

$$\alpha_4 = \frac{1}{24\beta_2\beta_3\beta_4}.$$

Substituting our formula for α_4 into (6.4.8) and solving for α_3:

$$\alpha_3 = \frac{4\beta_2 - 1}{24\beta_2^2\beta_3}.$$

Substituting our formulas for α_3 and α_4 into (6.4.13) and solving for β_3:

$$\beta_3 = -4\beta_2^2 + 3\beta_2.$$

Substituting our formulas for α_3, α_4 and β_3 into (6.4.10) and solving for β_4:

$$\beta_4 = (6 - 16\beta_2 + 16\beta_2^2)\beta_2.$$

Substituting our formulas for α_3, α_4, β_3 and β_4 into (6.4.6) and solving for α_2:

$$\alpha_2 = \frac{2 - 16\beta_2 + 52\beta_2^2 - 48\beta_2^3}{24\beta_2^3\,(3 - 4\beta_2)}.$$

Substituting our formulas for α_2, α_3, α_4, β_3 and β_4 into (6.4.7) and simplifying:

$$16\beta_2^3 - 12\beta_2^2 + 4\beta_2 - 1 = 0.$$

The roots of this last equation are $\beta_2 = \frac{1}{2}, \frac{1 \pm i\sqrt{7}}{8}$, so we conclude that $\beta_2 = \frac{1}{2}$. Back substituting, we find

$$\begin{aligned}
\beta_2 &= \frac{1}{2} \\
\alpha_2 &= \frac{1}{3} \\
\beta_4 &= 1 \\
\beta_3 &= \frac{1}{2} \\
\alpha_3 &= \frac{1}{3} \\
\alpha_4 &= \frac{1}{6}.
\end{aligned}$$

Substituting these values of α_2, α_3, and α_4 into (6.4.5), we find

$$\alpha_1 = \frac{1}{6}.$$

These seven values are the unique simultaneous real solution of the equations (6.4.14), (6.4.8), (6.4.13), (6.4.10), (6.4.6), (6.4.7), and (6.4.5). So the seven parameters are determined by 7 of the ten conditions. It remains to show that these seven values also satisfy (6.4.9), (6.4.11), and (6.4.12), which they do. Finally, note that these are the values of the parameters for the (classic) Runge-Kutta method of order 4.

Key Concepts

Taylor's theorem in two variables: Suppose $f(t, y)$ and all its partial derivatives of order $n + 1$ and lower are continuous on the rectangle $D = \{(t, y) : a \le t \le b, c \le y \le d\}$, and let $(t_0, y_0) \in D$. Then for every $(t, y) \in D$, there exist $\xi \in (a, b)$ and $\mu \in (c, d)$ such that

$$\begin{aligned}
f(t, y) &= f(t_0, y_0) + [(t - t_0) \cdot f_t(t_0, y_0) + (y - y_0) \cdot f_y(t_0, y_0)] \\
&\quad + \frac{1}{2}\left[(t - t_0)^2 f_{tt}(t_0, y_0) + 2(t - t_0)(y - y_0) \cdot f_{ty}(t_0, y_0) + (y - y_0)^2 f_{yy}(t_0, y_0)\right] \\
&\quad + \cdots + \\
&\quad \frac{1}{n!}\left[\sum_{j=0}^{n} \begin{pmatrix} n \\ j \end{pmatrix} (t - t_0)^{n-j}(y - y_0)^j \frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0, y_0)\right] \\
&\quad + \frac{1}{(n+1)!}\left[\sum_{j=0}^{n+1} \begin{pmatrix} n+1 \\ j \end{pmatrix} (t - t_0)^{n+1-j}(y - y_0)^j \frac{\partial^{n+1} f}{\partial t^{n+1-j}\partial y^j}(\xi, \mu)\right].
\end{aligned}$$

Exercises

1. Determine analytically the local truncation error for the o.d.e. solver derived in exercise 1 on page 192. Compare it to the local truncation error of the underlying integration formula. Are they the same? Also compare it to the experimentally determined rate of convergence (see exercise 2 on page 193). Is it one degree higher, as should be expected? [S] [A]

2. Execute one step of Runge-Kutta order four for solving $\dot{y} = ty$ with $y(1) = 0.5$ and $h = 1$, thus approximating $y(2)$. Compare your answer to that of section 6.2 exercise 1c on page 185 in which you used Euler's method with two steps. The exact solution is $y(2) = \frac{e^{3/2}}{2} \approx 2.240844535169032$. [S]

3. Explain geometrically, and in your own words, improved Euler's method.

4. C Write computer code that implements improved Euler's method (same as exercise 4 on page 193 except this time the method has a proper name). [A]

5. C Write computer code that implements Heun's third order method (same as exercise 5 on page 193 except this time the method has a proper name). [A]

6. C Write computer code that implements RK4. [A]

7. C Use your code from exercise 6 to compute $y(2)$ for the o.d.e. in exercise 1 on page 185 using step size $h = 0.05$. [S] [A]

6.5 Adaptive Runge-Kutta Methods

Two of the o.d.e. solvers derived in section 6.3 used the exact same set of calculations for k_1, k_2, and k_3, but combined the results differently to compute y_{i+1}. At the time, these were called open-ode and clopen-ode. In the analysis of section 6.4 it was noted that open-ode was not an efficient method while clopen-ode was, at which point we began referring to clopen-ode by its proper name, Heun's third order method.

Crumpet 34: Heun's third order method

In this article from 1900 [16] Karl Heun puts forth the third order method that bears his name. Even if you can not read the German, his formula VI) is clear!

references/heun1900/00000036.jpg

Due to its inefficiency, open-ode should never be used in practice by itself, but combined with Heun's third order method, it has some potential usefulness.

According to Heun's third order method

$$\begin{aligned}
k_1 &= f(t_i, y_i) \\
k_2 &= f\left(t_i + \frac{h}{3}, y_i + \frac{h}{3}k_1\right) \\
k_3 &= f\left(t_i + \frac{2h}{3}, y_i + \frac{2h}{3}k_2\right) \\
y_{i+1} &= y_i + \frac{h}{4}\left[k_1 + 3k_3\right] + O(h^4).
\end{aligned}$$

Using the same k_1, k_2, and k_3, the open-ode method is calculated as

$$y_{i+1} = y_i + \frac{h}{2}\left[k_2 + k_3\right] + O(h^3).$$

The difference between these estimates is

$$\frac{h}{4}\left[k_1 - 2k_2 + k_3\right] = Mh^3 + O(h^4) \tag{6.5.1}$$

for some constant M, and represents the local truncation error of the lower order method, open-ode. This error estimate can be used to adapt the size of h from one step to the next, decreasing the step size when the local truncation error is bigger than some tolerance and increasing the step size when the local truncation error is smaller than some tolerance.

To illustrate the algrorithm and the benefits of adaptive routines, let's return to o.d.e. 6.2.1, $\dot{y} = -\frac{y}{t} + t^2$, which we have generously leaned upon already. As before we will estimate $y(2)$ given initial condition $y(4) = 20$. This time the number of steps to compute will be determined by the algorithm, not by us, at least after the first step. Unfortunately, there is no standard or fool-proof way to choose the size of the first step. Because we are looking for a computation that can be done by hand, let's try $h = -1$ to begin, $\frac{1}{2}$ of the width of the interval $[2, 4]$, over which we will integrate.

As was needed for adaptive quadrature, a desired level of accuracy, or tolerance, is needed here too. Again because we are looking for a computation that can be done by hand, let's try 0.1, a pretty modest accuracy. Finally, we are ready to compute:

$$\begin{aligned}
k_1 &= f(4, 20) = 11 \\
k_2 &= f\left(4 - \frac{1}{3}, 20 - \frac{1}{3} \cdot 11\right) \approx 8.98989898989899 \\
k_3 &= f\left(4 - \frac{2}{3}, 20 - \frac{2}{3} \cdot 8.9898\ldots\right) \approx 6.90909090909091.
\end{aligned}$$

Before computing y_1 from these values, we need to check that the expected accuracy of the calculation would not violate the 0.1 requirement:

$$\left|\frac{h}{4}\left[k_1 - 2k_2 + k_3\right]\right| \approx 0.017.$$

The approximate error in stepping to $t_1 = 3$ is about 0.02, well below the desired threshhold. We are clear to proceed:

$$\begin{aligned}
y_1 &= y_0 + \frac{h}{4}\left[k_1 + 3k_3\right] \approx 12.06818181818182 \\
t_1 &= t_0 + h = 3.
\end{aligned}$$

Hence we have $y(3) \approx 12.07$. Continuing with $h = 1$,

$$\begin{aligned}
k_1 &= f(3, 12.068\ldots) \approx 4.977272727272728 \\
k_2 &= f\left(3 - \frac{1}{3}, 12.068\ldots - \frac{1}{3} \cdot 4.9773\ldots\right) \approx 3.20770202020202 \\
k_3 &= f\left(3 - \frac{2}{3}, 12.068\ldots - \frac{2}{3} \cdot 3.2077\ldots\right) \approx 1.188852813852814.
\end{aligned}$$

Before computing y_2 from these values, we need to check that the expected accuracy of the calculation would not violate the 0.1 requirement:

$$\left|\frac{h}{4}\left[k_1 - 2k_2 + k_3\right]\right| \approx 0.062.$$

The approximate error in stepping to $t_2 = 2$ is about 0.06, well below the desired threshhold. We are clear to proceed:

$$\begin{aligned} y_2 &= y_1 + \frac{h}{4}\left[k_1 + 3k_3\right] \approx 9.932224025974026 \\ t_1 &= t_0 + h = 2. \end{aligned}$$

Hence we have $y(2) \approx 9.932$. After two steps, the actual error is about $|10 - 9.932| = 0.068$. Of course, we could have simply executed Heun's third order method with step size $h = 1$ (and no error checking) and gotten the same answer. The difference is we would not have had any idea what to expect for an error! With the adaptive method, you can be reasonably sure each step incurs only the error you request. At the risk of belaboring the point, consider redoing the calculation with step size $h = -2$:

$$\begin{aligned} k_1 &= f(4, 20) = 11 \\ k_2 &= f\left(4 - \frac{2}{3}, 20 - \frac{2}{3} \cdot 11\right) \approx 7.311111111111111 \\ k_3 &= f\left(4 - \frac{4}{3}, 20 - \frac{4}{3} \cdot 7.3111\ldots\right) \approx 3.266666666666667. \end{aligned}$$

If we proceed with Heun's third order method (and no error checking), we get

$$\begin{aligned} y_1 &= y_0 + \frac{h}{4}\left[k_1 + 3k_3\right] \approx 9.6 \\ t_1 &= t_0 + h = 2. \end{aligned}$$

However, without the exact answer, which will be the usual when using a numerical method, we have no way to know how accurate this estimate is! In that regard, the value 9.6 is a somewhat useless estimate.

On the other hand, since we know the exact value of $y(2)$ is 10, we know the error is 0.4, larger than the desired 0.1. The adaptive Heun should catch this and arrive at a more accurate estimate:

$$\left|\frac{h}{4}\left[k_1 - 2k_2 + k_3\right]\right| \approx 0.177.$$

The adaptive method would reject this step because the approximate error is greater than the desired accuracy, without calculating y_1! So what should it do instead? The adaptive method will try again with a smaller step size.

Since

$$\left|\frac{h}{4}\left[k_1 - 2k_2 + k_3\right]\right| \approx Mh^3,$$

we have $Mh^3 \approx 0.177$ for any step size close to the one just attempted. If we scale the step size by a factor of q, say, we should expect the new error to be approximately $M(qh)^3$, or $q^3Mh^3 \approx 0.177q^3$. Since we would like that error to be no more than 0.1, we should choose q so that $0.177q^3 < 0.1$ or $q^3 < \frac{0.1}{0.177}$, which implies $q < \sqrt[3]{\frac{0.1}{0.177}} \approx 0.8254$. But it would slow down the algorithm immensely if the step size were too large very often, so instead, we will take a somewhat conservative next step of $0.9qh \approx 0.9(0.8254)(-2) \approx -1.485$. Recalculating with the new step size:

$$\begin{aligned} k_1 &= f(4, 20) = 11 \\ k_2 &= f\left(4 - \frac{1.485}{3}, 20 - \frac{1.485}{3} \cdot 11\right) \approx 8.130924301356263 \\ k_3 &= f\left(4 - \frac{4}{3}, 20 - \frac{4}{3} \cdot 7.3111\ldots\right) \approx 5.087191526760124. \end{aligned}$$

and

$$\left|\frac{h}{4}\left[k_1 - 2k_2 + k_3\right]\right| \approx 0.06487930780869297,$$

so this step is accepted:

$$\begin{aligned} y_2 &= y_1 + \frac{h}{4}\left[k_1 + 3k_3\right] \approx 10.24469652063055 \\ t_1 &= t_0 + h = 2.514132737997418. \end{aligned}$$

Now we keep the new step size until it proves to be inappropriate. In this case, that happens right away. Another step of -1.485 would take the solution to $t_2 \approx 1.028$, well past the desired $t = 2$. So, we shorten the step size to $2 - t_1 = -0.514132737997418$. There is no worry about shortening the step size as that is expected to reduce the error! Finally, with $h = -0.514132737997418$:

$$\begin{aligned} k_1 &= f(2.514\ldots, 10.244\ldots) \approx 2.246020292164824 \\ k_2 &= f\left(2.514\ldots - \frac{0.5141\ldots}{3}, 10.244\ldots - 2\frac{0.5141\ldots}{3}\cdot 2.246\ldots\right) \approx 1.279876276642283 \\ k_3 &= f\left(2.514\ldots - \frac{0.5141\ldots}{3}, 10.244\ldots - 2\frac{0.5141\ldots}{3}\cdot 1.279\ldots\right) \approx 0.1988478127940674. \end{aligned}$$

and

$$\left|\frac{h}{4}\left[k_1 - 2k_2 + k_3\right]\right| \approx 0.01476646399275057,$$

this step is accepted:

$$\begin{aligned} y_2 &= y_1 + \frac{h}{4}\left[k_1 + 3k_3\right] \approx 9.879332752200975 \\ t_1 &= t_0 + h = 2. \end{aligned}$$

We have $y(2) \approx 9.879332752200975$ with some confidence that the error will not be terribly much more than about 0.2, since we took two steps each of which may have incurred an error of about 0.1. There is no guarantee the error will be less than 0.2, but at least we have some confidence that it's not drastically greater. And because we used a conservative estimate for step size, the actual error is probably a bit smaller (as it turns out, the error is about 0.12).

Adaptive Runge-Kutta (pseudo-code)

There are many different adative Runge-Kutta schemes, but the one discussed here uses second and third order methods, so might be called RK2(3). Technically, it is an order 2 method since the error estimate is for the lower order method. In practice, however, it is often the higher order method that is used for the o.d.e. solution. While there is never any guarantee the higher order method is more accurate than the lower order method, it rarely causes any adverse problems. Besides hedging our bets with the 0.9 safety factor when adjusting the step size, we also disallow any scaling of h by any factor less than 0.1 or any factor greater than 5. These extra safeties are not terribly restrictive since they allow for exponential growth or decay of h, but they can help avoid problems when the error estimates are simply bad. Moreover, the estimates are only good for a small range since the constant of proportionality may change dramatically for large changes in h. A more detailed discussion of the algorithm can be found in [26] Section 16.2.

```
Assumptions: $\dot{y} = f(t, y)$, $y(a) = y_0$ has a unique solution over the interval from $a$ to $b$.
Input: Initial value $(a, y_0)$; function $f(t, y)$; interval endpoints, $a$ and $b$; initial step size $h$; desired accuracy $tol$; maximum number of iterations $N$.
Step 1: Set $i = 1$; $t = a$; $y = y_0$; done = $false$;
Step 2: While not done and $i \le N$ do Steps 3-6:
    Step 3: If $((b - (t + h)) \cdot (b - a) \le 0)$ then set $h = b - t$; done = $true$;
    Step 4: Set $k_1 = f(t, y)$; $k_2 = f(t + \frac{h}{3}, y + \frac{h}{3}k_1)$; $k_3 = f(t + \frac{2h}{3}, y + \frac{2h}{3}k_2)$; err = $|\frac{h}{4}(k_1 - 2k_2 + k_3)|$;
    Step 5: If done or err $\le tol$ then set $y = y + \frac{h}{4}(k_1 + 3k_3)$; temp = $t + h$;
    Step 6: If temp = $t$ then do Steps 7-8:
        Step 7: Print "Method failed. Step size reached zero."
        Step 8: Return
```

```
      Step 9: Set $i = i + 1$;
      Step 10: If err $< \frac{\text{tol}}{5}$ or err $>$ tol then do steps 11-14:
          Step 11: Set $q = 0.9\left(\frac{\text{tol}}{\text{err}}\right)^{\frac{1}{3}}$
          Step 12: If $q < \frac{1}{10}$ then set $q = \frac{1}{10}$
          Step 13: If $q > 5$ then set $q = 5$
          Step 14: Set $h = qh$
  Step 15: If not done then Print "Method failed. Maximum iterations exceeded."
  Output: Approximation $y(b)$ or message of failure.
```

The formulas for k_i and err will need to be changed for different adaptive Runge-Kutta schemes, as will the recalculation of h in Steps 11-14, but the basic algorithm does not require modification for other embedded methods.

General Runge-Kutta Schemes

Up to now, we have considered Runge-Kutta methods of the form (6.4.2), copied here for convenience:

$$\begin{aligned}
k_1 &= f(t_i, y_i) \\
k_2 &= f\left(t_i + \beta_2 h, y_i + \beta_2 h k_1\right) \\
k_3 &= f(t_i + \beta_3 h, y_i + \beta_3 h k_2) \\
&\vdots \\
k_s &= f(t_i + \beta_s h, y_i + \beta_s h k_{s-1}) \\
y_{i+1} &= y_i + h\left[\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \cdots + \alpha_s k_s\right].
\end{aligned}$$

In methods of this type, k_1 is used in the computation of k_2; k_2 is used in the computation of k_3; k_3 is used in the computation of k_4; and so on. However, there is nothing preventing one from deriving a method where both k_1 and k_2 are used in the computation of k_3; all of k_1, k_2, and k_3 are used in the computation of k_4; and in general allowing all of $k_1, k_2, \ldots, k_{j-1}$ to be used in computing k_j. Doing so gives more degrees of freedom for satisfying the error analysis equations, lending hope that there are many more Runge-Kutta methods possible. Any method of this more general form is called an explicit Runge-Kutta method and can be formulated as

$$\begin{aligned}
k_1 &= f(t_i, y_i) \\
k_2 &= f\left(t_i + \delta_2 h, y_i + \beta_{21} h k_1\right) \\
k_3 &= f(t_i + \delta_3 h, y_i + \beta_{31} h k_1 + \beta_{32} h k_2) \\
&\vdots \\
k_s &= f(t_i + \delta_s h, y_i + \sum_{j=1}^{s-1} \beta_{sj} h k_j) \\
y_{i+1} &= y_i + h\left[\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \cdots + \alpha_s k_s\right].
\end{aligned} \tag{6.5.2}$$

Methods of this form are often summarized in a Butcher tableau,

$$\begin{array}{c|ccccc}
0 & & & & & \\
\delta_2 & \beta_{21} & & & & \\
\delta_3 & \beta_{31} & \beta_{32} & & & \\
\vdots & \vdots & & \ddots & & \\
\delta_s & \beta_{s1} & \beta_{s2} & \cdots & \beta_{s(s-1)} & \\
\hline
 & \alpha_1 & \alpha_2 & \cdots & \alpha_{s-1} & \alpha_s
\end{array}$$

much like the coefficients of a system of linear equations might be summarized in a matrix. The Butcher tableau for any of the Runge-Kutta methods we have considered so far will take the form

$$\begin{array}{c|cccccc}
0 & & & & & & \\
\delta_2 & \beta_{21} & & & & & \\
\delta_3 & 0 & \beta_{32} & & & & \\
\delta_4 & 0 & 0 & \beta_{43} & & & \\
\vdots & \vdots & \vdots & \ddots & \ddots & & \\
\delta_s & 0 & 0 & \cdots & 0 & \beta_{s(s-1)} & \\
\hline
 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{s-1} & \alpha_s
\end{array}$$

For example, Heun's third order method would be summarized in a Butcher tableau as

$$\begin{array}{c|ccc}
0 & & & \\
\frac{1}{3} & \frac{1}{3} & & \\
\frac{2}{3} & 0 & \frac{2}{3} & \\
\hline
 & \frac{1}{4} & 0 & \frac{3}{4}
\end{array}$$

For our purposes, adaptive Runge-Kutta schemes, also called embedded methods, will be coded in a Butcher tableau by adding one more line for the coefficients α_j of the lower order method. For example the Butcher tableau for RK2(3) as presented above would be

$$\begin{array}{c|ccc}
0 & & & \\
\frac{1}{3} & \frac{1}{3} & & \\
\frac{2}{3} & 0 & \frac{2}{3} & \\
\hline
 & \frac{1}{4} & 0 & \frac{3}{4} \\
\hline
 & 0 & \frac{1}{2} & \frac{1}{2}
\end{array}$$

The most general Butcher tableaux for non-embedded methods take the form

$$\begin{array}{c|cccc}
0 & \beta_{11} & \beta_{12} & \cdots & \beta_{1s} \\
\delta_2 & \beta_{21} & \beta_{22} & \cdots & \beta_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\delta_s & \beta_{s1} & \beta_{s2} & \cdots & \beta_{ss} \\
\hline
 & \alpha_1 & \alpha_2 & \cdots & \alpha_s
\end{array}$$

If any of the β_{ij} with $j > i$ are nonzero, the associated Runge-Kutta scheme is an implicit method. Each step of the method will require solving a system of equations. Implicit Runge-Kutta methods can be considered for approximating the solutions of stiff o.d.e. since explicit methods are often exceedingly bad at it.

Crumpet 35: A Stiff Ordinary Differential Equation

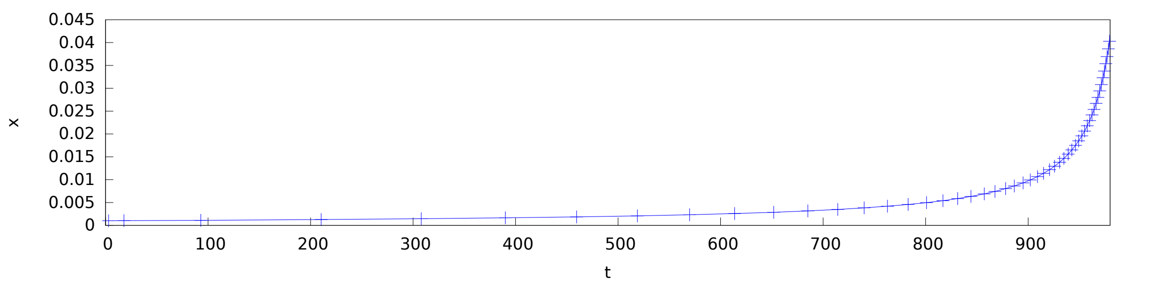
The ordinary differential equation

$$\begin{aligned} \dot{x} &= x^2 - x^3 \\ x(0) &= \delta \end{aligned} \tag{6.5.3}$$

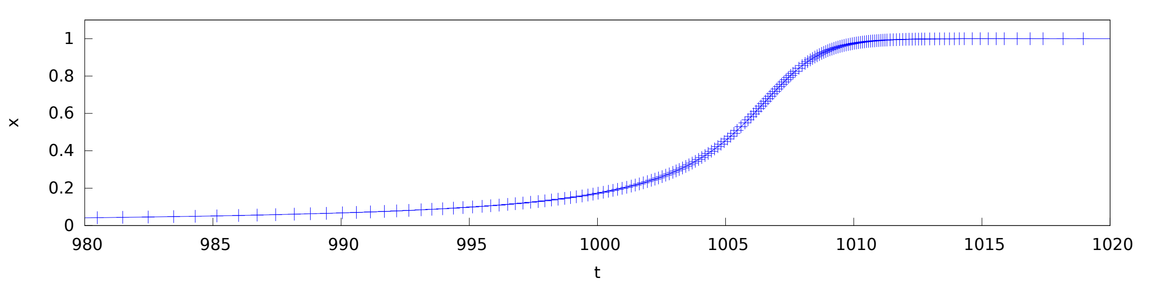
has no closed form solution. The best one can do is derive an implicit solution, so a numerical solution is necessary to approximate values of the function. Some basic analysis can give an idea what the solution is like, however. It has an equilibrium at $x = 0$, which means if $x(t_0) = 0$ for some t_0, then $x(t) = 0$ for all t. The function remains constant for all time. It is in equilibrium. It does not change. This follows from the fact that when $x = 0$,

$\dot{x} = 0^2 - 0^3 = 0$. Similarly, the o.d.e. has an equilibrium at $x = 1$ (because 1 is another root of the polynomial $x^2 - x^3$), and it has no others. However, the two equilibria are very different from one another. The equilibrium at $x = 0$ is unstable while the equilibrium at $x = 1$ is stable. If $x(t_0)$ is near enough to 1 ($|x(t_0) - 1| < 1$ will do), then x will tend toward 1 as $t \to \infty$. However, there is no such condition near $x = 0$. No matter how close $x(t_0)$ is to zero, if it is positive, x will still tend to the other equilibrium, 1, as $t \to \infty$. More to the point, though, is how the values of x approach 1 as $t \to \infty$.

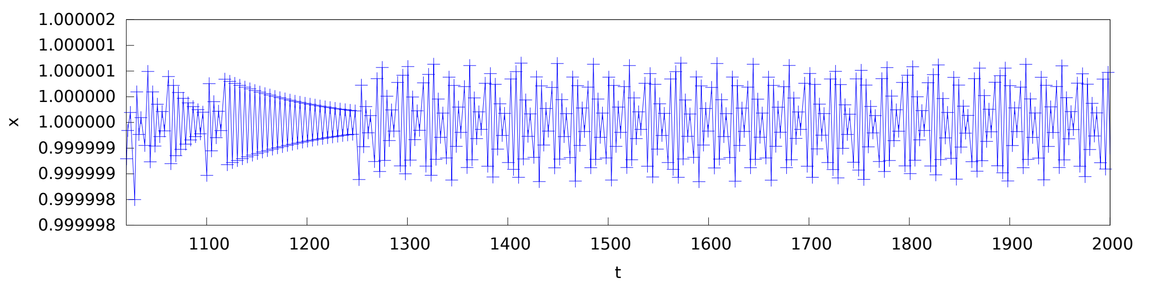
The hope for an adaptive o.d.e. solver is that it will take large steps where the function is not varying quickly (has a small first derivative) and will be more careful by taking small steps where the function is varying quickly (has a large first derivative). More often than not, this is exactly what happens. Stiff o.d.e.s are an exception to the rule where an adaptive method takes many small steps even in a region where the function has a small first derivative. The following figures show the solution of (6.5.3) using RK2(3) with tolerance 10^{-6}, $\delta = 10^{-3}$, and initial step size 3 over the interval $[0, \frac{2}{\delta}]$. First, the solution over $[0, 980]$ acts as we would hope. The solver takes large steps, including one step from $t \approx 93$ to $t \approx 210$, a step size $h > 117$ at the beginning where the function changes very slowly.



In the middle, the solution over $[980, 1020]$ continues to act as we would hope. The solution begins to vary more quickly here and, consequently, the solver takes a number of smaller steps.



Toward the end, the solution over $[1020, 2000]$ demonstrates the consequence of stiffness. The exact solution is very nearly constant over this region, gradually approaching 1 from below. A good solver would again take large steps across this region, but adaptive explicit Runge-Kutta schemes do not. The numerical solution oscillates within tolerance about 1, so it does what it is supposed to do, but it takes many short steps to do so.



Key Concepts

Embedded Runge-Kutta method: A Runge-Kutta method in which there are two schemes of different orders derived from the same set of function evaluations.

Adaptive Runge-Kutta method: A Runge-Kutta method that takes advantage of an embedded Runge-Kutta scheme to automatically adapt the step size as it estimates the solution of an o.d.e.

Butcher tableau: A tabular representation of a Runge-Kutta method.

RK$m(n)$: Shorthand for an embedded Runge-Kutta method containing schemes with rates of convergence (commonly called orders) m and n.

Exercises

1. Write computer code that implements RK2(3) as presented in pseudo-code. [A]

2. Which are the Butcher tableaux of implicit methods? [A]

(a)
$$\begin{array}{c|ccccc} 0 & & & & & \\ \frac{1}{4} & \frac{1}{8} & \frac{1}{8} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & \\ \frac{3}{4} & \frac{3}{16} & 0 & \frac{9}{16} & & \\ 1 & -\frac{3}{7} & 2 & -\frac{12}{7} & \frac{8}{7} & \\ \hline & \frac{7}{90} & \frac{32}{90} & \frac{12}{90} & \frac{32}{90} & \frac{7}{90} \end{array}$$

(b)
$$\begin{array}{c|ccccc} 0 & & & & & \\ \frac{1}{4} & \frac{1}{4} & & & & \\ \frac{3}{4} & -\frac{9}{4} & 3 & & & \\ \frac{1}{2} & \frac{1}{18} & \frac{5}{12} & \frac{1}{36} & & \\ 1 & \frac{7}{9} & -\frac{5}{3} & -\frac{1}{9} & 2 & \\ \hline & \frac{1}{6} & 0 & 0 & \frac{2}{3} & \frac{1}{6} \end{array}$$

(c)
$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

(d)
$$\begin{array}{c|cccc} 0 & \frac{1}{12} & -\frac{\sqrt{5}}{12} & \frac{\sqrt{5}}{12} & \frac{1}{12} \\ \frac{5-\sqrt{5}}{10} & \frac{1}{12} & \frac{1}{4} & \frac{10-7\sqrt{5}}{60} & \frac{\sqrt{5}}{60} \\ \frac{5+\sqrt{5}}{10} & \frac{1}{12} & \frac{10+7\sqrt{5}}{60} & \frac{1}{4} & -\frac{\sqrt{5}}{60} \\ 1 & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \\ \hline & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \end{array}$$

3. Show that this is the Butcher tableau for Euler's method.

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

4. Show that this is the Butcher tableau for the improved Euler method. [S]

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

5. Show that the method given by the Butcher tableau has order 2 for any $\delta \in [\frac{1}{2}, 1]$.

$$\begin{array}{c|cc} 0 & & \\ \delta & \delta & \\ \hline & 1-\frac{1}{2\delta} & \frac{1}{2\delta} \end{array}$$

6. Demonstrate numerically that the method suggested by the Butcher tableau has rate of convergence $O(h^3)$.

(a)
$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & 0 & \frac{2}{3} & & \\ 1 & 0 & 0 & 1 & \\ \hline & 0 & \frac{3}{4} & 0 & \frac{1}{4} \end{array}$$

(b)
$$\begin{array}{c|cccc} 0 & & & & \\ \frac{2}{7} & \frac{2}{7} & & & \\ \frac{4}{7} & -\frac{8}{35} & \frac{4}{5} & & \\ \frac{6}{7} & \frac{29}{42} & -\frac{2}{3} & \frac{5}{6} & \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{5}{12} & \frac{1}{4} \end{array}$$ [S]

(c)
$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{3}{4} & 0 & \frac{3}{4} & \\ \hline & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} \end{array}$$

7. Euler's method and the improved Euler method use the same function evaluations. Thus, they can be combined into an embedded, and therefore adaptive, method. Write the Butcher tableau for the Euler/improved Euler embedded method.

8. Write computer code that implements the adaptive method suggested in exercise 7.

9. **$\frac{3}{8}$-rule Runge-Kutta method**. Demonstrate numerically that the $\frac{3}{8}$-rule method, given by the Butcher tableau, has rate of convergence $O(h^4)$.

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{3} & \frac{1}{3} & & & \\
\frac{2}{3} & -\frac{1}{3} & 1 & & \\
1 & 1 & -1 & 1 & \\
\hline
 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8}
\end{array}$$

10. Write computer code that implements the RK3(4) adaptive method ([6] page 301) given by the Butcher tableau. [S]

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & \\
\frac{3}{4} & -\frac{9}{4} & 3 & & & \\
\frac{1}{2} & \frac{1}{18} & \frac{5}{12} & \frac{1}{36} & & \\
1 & \frac{7}{9} & -\frac{5}{3} & -\frac{1}{9} & 2 & \\
\hline
 & \frac{1}{6} & 0 & 0 & \frac{2}{3} & \frac{1}{6} \\
\hline
 & \frac{7}{9} & -\frac{5}{3} & -\frac{1}{9} & 2 & 0
\end{array}$$

11. **Cash-Karp RK4(5).** Write computer code that implements the Cash-Karp adaptive method given by the Butcher tableau. [A]

$$\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & \\
\frac{3}{5} & \frac{3}{10} & -\frac{9}{10} & \frac{6}{5} & & & \\
1 & -\frac{11}{54} & \frac{5}{2} & -\frac{70}{27} & \frac{35}{27} & & \\
\frac{7}{8} & \frac{1631}{55296} & \frac{175}{512} & \frac{575}{13824} & \frac{44275}{110592} & \frac{253}{4096} & \\
\hline
 & \frac{37}{378} & 0 & \frac{250}{621} & \frac{125}{594} & 0 & \frac{512}{1771} \\
\hline
 & \frac{2825}{27648} & 0 & \frac{18575}{48384} & \frac{13525}{55296} & \frac{277}{14336} & \frac{1}{4}
\end{array}$$

12. The following pairs of Runge-Kutta methods use the same function evaluations, but have different rates of convergence. They can each therefore be paired to form an embedded Runge-Kutta scheme. Write the Butcher tableau for the embedded method.

(a) The method of exercise 6a and open-ode.

(b) The $\frac{3}{8}$-rule (exercise 9) and the following. [A]

$$\begin{array}{c|ccc}
0 & & & \\
\frac{1}{3} & \frac{1}{3} & & \\
\frac{2}{3} & -\frac{1}{3} & 1 & \\
\hline
 & 0 & \frac{1}{2} & \frac{1}{2}
\end{array}$$

(c) The $\frac{3}{8}$-rule (exercise 9) and the following.

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{3} & \frac{1}{3} & & & \\
\frac{2}{3} & -\frac{1}{3} & 1 & & \\
1 & 1 & -1 & 1 & \\
\hline
 & \frac{3}{2} & -\frac{3}{2} & 0 & 1
\end{array}$$

(a) The method of exercise 6b and the following.

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{2}{7} & \frac{2}{7} & & & & \\
\frac{4}{7} & -\frac{8}{35} & \frac{4}{5} & & & \\
\frac{6}{7} & \frac{29}{42} & -\frac{2}{3} & \frac{5}{6} & & \\
1 & \frac{1}{6} & \frac{1}{6} & \frac{5}{12} & \frac{1}{4} & \\
\hline
 & \frac{11}{96} & \frac{7}{24} & \frac{35}{96} & \frac{7}{48} & \frac{1}{12}
\end{array}$$

(b) **Bogacki–Shampine rk2(3).** The method of exercise 6c and the following. [S]

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{3}{4} & 0 & \frac{3}{4} & & \\
1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
\hline
 & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8}
\end{array}$$

13. Butcher [6] credits Merson (1957) with the earliest proposed embedded Runge-Kutta method, given by the Butcher tableau. What are the orders of the two methods?

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{1}{3} & \frac{1}{3} & & & & \\
\frac{1}{3} & \frac{1}{6} & \frac{1}{6} & & & \\
\frac{1}{2} & \frac{1}{8} & 0 & \frac{3}{8} & & \\
1 & \frac{1}{2} & 0 & -\frac{3}{2} & 2 & \\
\hline
 & \frac{1}{6} & 0 & 0 & \frac{2}{3} & \frac{1}{6} \\
\hline
 & \frac{1}{10} & 0 & \frac{3}{10} & \frac{2}{5} & \frac{1}{5}
\end{array}$$

14. **Merson (1957).** Write computer code that implements the adaptive method of exercise 13. [A]

15. The initial value problem

$$\begin{aligned}
y' &= \frac{x + 2e^y \cos(e^x)}{1 + e^y} \\
y(0) &= 2
\end{aligned} \tag{6.5.4}$$

can not be solved analytically. The solution must be approximated. Use your code from the given exercise to approximate $y(4)$ with an error of no more than 10^{-4}.

(a) 1 [S]

(b) 8

(c) 10

(d) 11 [A]

(e) 12a

(f) 12b [A]

(g) 12c

(h) 12a

(i) 12b

(j) 13

(k) 14

16. The initial value problem

$$\begin{aligned} y' &= \frac{x^2+y}{x-y^2} \\ y(0) &= 5 \end{aligned} \tag{6.5.5}$$

can not be solved analytically. The solution must be approximated. Use your code from the given exercise to approximate $y(3)$ with an error of no more than 10^{-4}.

(a) 1 [S]

(b) 8

(c) 10

(d) 11 [A]

(e) 12a

(f) 12b [A]

(g) 12c

(h) 12a

(i) 12b

(j) 13

(k) 14

17. Consider the initial value problem

$$\begin{aligned} y' &= -\frac{\frac{2}{x}+y^2}{2xy} \\ y(1) &= 1. \end{aligned}$$

(a) Use your code from exercise 5 on page 206 (Heun's third order method) to estimate $y(2)$ with step size 0.01.

(b) Use your code from exercise 6 on page 206 (RK4) to estimate $y(2)$ with step size 0.01.

(c) Compare the results of parts (a) and (b). You should notice that they are rather different. The rest of this exercise explores the reason for the discrepancy.

(d) Use your code from exercise 1 (rk2(3)) to estimate $y(2)$ with tolerance 0.001 and maximum number of steps 1000.

(e) Use your code from any of the parts of exercise 12 to estimate $y(2)$ with tolerance 0.001 and maximum number of steps 1000.

(f) You should have found that the method fails in both parts (d) and (e). However, if you look at the last calculated values of x and y anyway (`x(1001)` and `y(1001)`), you should find that in both cases, $x \approx 1.648$ and $y \approx 0$. The failure to approximate $y(2)$ is not a shortcoming of the numerical method. The solution of the initial value problem only exists over the interval $[1, \sqrt{e}) \approx [1, 1.648)$. For dependable results, care must be taken that the solution of the o.d.e. exists and is unique over the entire interval from a to b. That said, the basic (non-adaptive) solvers plow right along and give an approximation for $y(2)$ that is entirely incorrect. Without some further analysis, you may not notice that the basic solvers are producing bogus information. On the other hand, the adaptive solvers give some clue as to what is going on due to their failure to proceed beyond $x = \sqrt{e}$. They get "stuck" taking tinier and tinier steps near $x = \sqrt{e}$, as they should since the solution does not exist beyond that point.

18. Attempt to approximate $y(4)$ for the initial value problem in exercise 16. Use a variety of adaptive and non-adaptive methods with a variety of tolerances. You should find that you can not obtain dependable results. Can you explain why not? HINT: You may wish to plot the approximate solutions. If your solvers are written so as to store the points in arrays, it is a simple matter to plot the solutions, as demonstrated for RK2(3), using the code from the solution of exercise 1.

```
[y,x]=rk23(f,0,5,4,.0001,1000);
plot(x,y)
```

19. The initial value problem

$$\begin{aligned} y' &= \ln(x+y) \\ y(0) &= \frac{1}{2} \end{aligned}$$

can not be solved analytically. The solution must be approximated. Apply the indicated method to compute $y(5)$ using tolerance 10^{-4} and an initial step size $\frac{1}{10}$. Is the global error (the error in approximating $y(5)$) around 10^{-4}? significantly smaller? significantly larger? Accurate to 10 significant digits, $y(5) = 6.409445034$. [A]

(a) Cash-Karp (exercise 11)

(b) Bogacki-Shampine (exercise 12b)

(c) Merson (exercise 14)

(d) RK2(3) (exercise 1)

20. Modify the code you used in exercise 19 to count the number of function evaluations performed. Which method was most efficient? The method with the fewest evaluations was the most efficient. [A]

21. There are many embedded methods not mentioned in this text, mostly of high order. Look some of them up, write code to implement them, and test your code. In particular, you may look for the methods of Fehlberg, Verner, or Dormand & Prince.

22. The Cash-Karp RK4(5) method [8] was designed to contain embedded methods of all orders from 1 through 5, not just orders 4 and 5. Show that the three embedded methods given in the Butcher tableau have the indicated orders.

$$\begin{array}{c|ccccl}
0 & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & \\
\frac{3}{5} & \frac{3}{10} & -\frac{9}{10} & \frac{6}{5} & & \\
\hline
 & \frac{19}{54} & 0 & -\frac{10}{27} & \frac{55}{54} & \text{Order 3} \\
\hline
 & -\frac{3}{2} & \frac{5}{2} & 0 & 0 & \text{Order 2} \\
\hline
 & 1 & 0 & 0 & 0 & \text{Order 1}
\end{array}$$

Bibliography

[1] Robert E. Barnhill and Richard F. Riesenfeld, editors. *Computer Aided Geometric Design : Proceedings of a conference held at the University of Utah, Salt Lake City, Utah, March 18-21, 1974*. Academic Press, New York, 1974.

[2] Michael F. Barnsley. *Fractals Everywhere*. Academic Press, Boston, 1988.

[3] R. P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 14(4):422–425, 1971.

[4] John Briggs and F. David Peat. *Turbulent Mirror*, page 69. Harper & Row Publishers, New York, 1989.

[5] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Thomson Brooks/Cole, 8th edition, 2005.

[6] J.C. Butcher. *The Numerical Analysis of Ordinary Differential Equations : Runge-Kutta and General Linear Methods*. John Wiley & Sons, 1987.

[7] J.C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 20:247–260, 1996.

[8] J.R. Cash and Alan H. Karp. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software*, 16(3):201–222, September 1990.

[9] Bill Casselman. From Bèzier to Bernstein. http://www.ams.org/samplings/feature-column/fcarc-bezier, June 2014.

[10] Paul de Faget de Casteljau. De Casteljau's autobiography : My time at Citroën. *Computer Aided Geometric Design*, 16(7):583–586, August 1999.

[11] David Goldberg. What every computer scientist should know about floating-point arithmetic. http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html, Accessed June 2014.

[12] S. W. Golomb. Checker boards and polyominoes. *Amer. Math. Monthly*, 61:675–682, 1954.

[13] Richard Guichard. Calculus : Early transcendentals. http://www.whitman.edu/mathematics/multivariable/, January 2014.

[14] Denny Gulick. *Encounters with Chaos*, page 2. McGraw-Hill, New York, 1992.

[15] Bryce Harrington and Johan Engelen. Inkscape. Software available at http://www.inkscape.org/.

[16] K. Heun. Neue methode zur approximativen integration der differentialgleichungen einer unabhängigen veränderlichen. *Zeitschrift für Mathematik und Physik*, 45:23–38, 1900.

[17] Jeffery J. Leader. *Numerical Analysis and Scientific Computing*. Pearson, 2004.

[18] Eugene Loh and G. William Walster. Rump's example revisited. *Reliable Computing*, 8(3):245–248, 2002.

[19] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, March 1963.

[20] Michael R. Matthews. *Time for science education : how teaching the history and philosophy of pendulum motion can contribute to science literacy*. Kluwer Academic/Plenum Publishers, New York, 2000.

[21] Michael R. Matthews, Michael P. Clough, and Craig Ogilvie. Pendulum motion: The value of idealization in science. `http://www.storybehindthescience.org/pdf/pendulum.pdf`.

[22] Cleve Moler. *Numerical Computing with MATLAB*, chapter 4. The MathWorks, Natick, MA, 2004. `https://www.mathworks.com/moler/index_ncm.html`.

[23] David E. Müller. A method for solving algebraic equations using an automatic computer. *Mathematical Tables and Other Aids to Computation*, 10(56):208–215, October 1956.

[24] L. Mumford. *Technics and Civilization*. Harcourt Brace Jovanovich, New York, 1934.

[25] Ron Naylor. Galileo, copernicanism and the origins of the new science of motion. *The British Journal for the History of Science*, 36(2):151–181, June 2003.

[26] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, New York, 2nd edition, 1999.

[27] The GNOME Project. Dia. Software available at `http://live.gnome.org/Dia`.

[28] Siegfried M. Rump. Algorithms for verified inclusions: Theory and practice. In R. E. Moore, editor, *Reliability in Computing: The Role of Interval Methods in Scientific Computing*, pages 109–126, Boston, 1988. Academic Press.

[29] J. R. Sharma. A family of methods for solving nonlinear equations using quadratic interpolation. *Computers and Mathematics with Applications*, 48(5-6):709–714, September 2004.

[30] Avram Sidi. Generalization of the secant method for nonlinear equations. *Applied Mathematics E-Notes*, 8:115–123, 1999. Available free at mirror sites of `http://www.math.nthu.edu.tw/~amen/`.

[31] Gilbert Strang. Calculus. `http://ocw.mit.edu/ans7870/resources/Strang/Edited/Calculus/Calculus.pdf`. Accessed June 2014.

[32] Ruedeger Timm et al. Libreoffice. Software available at `http://www.libreoffice.org/`.

[33] Unknown. Huygens' clocks. `http://www.sciencemuseum.org.uk/onlinestuff/stories/huygens_clocks.aspx`.

[34] Charles F. Van Loan. *Introduction to Scientific Computing : A Matrix Vector Approach Using MATLAB*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition, 2000.

[35] Christopher Vickery. IEEE-754 analysis. `http://babbage.cs.qc.cuny.edu/IEEE-754/`. Accessed June 2013.