Notes for Math 611 Numerical Methods for Computation – Fall 2002

Bruce Bukiet*

Introduction

Purpose: Approximating solutions efficiently for scientific and mathematical problems. Numerical methods are especially useful when problems cannot be solved analytically or if the analytical solution is not meaningful.

The Basics

- students must have a solid understanding of single-variable calculus (Calc I and 2)
- students should know some partial differentiation
- students must be know how to program in a programming language
- students should have a basic knowledge of linear algebra and ordinary differential equations

What you'll learn: I hope

- where approximation methods come from
- why they work and why they sometimes don't work
- when to use a particular method and why

*Department of Mathematics, Department of Biomedical Engineering, Center for Applied Mathematics and Statistics, New Jersey Institute of Technology, Newark, NJ 07102

- how to check your work: for example, for the trapezoidal method for numerical integration, the error should decrease by a factor of 4 when the grid spacing halves. If it does not decrease by a factor of 4, you have an error in the code or the integrand is not sufficiently smooth.

You have to understand what you are approximating well enough to construct a reasonable approximation. It is important to learn the underlying concepts. However, numerical methods is also experimental. You can learn alot from trying a computation out and seeing how it goes.

Some examples of usefulness of numerical methods
Finance

- compound growth and investments (ODE)
- mortgages and annuity issues (root finding)
- options pricing (PDE)
- minimizing costs (Optimization)

Engineering

- Multiple mass springs (ODE)
- Heat transfer (BVP - PDE, Interpolation)
- Chemical concentrations and masses (Integration)

Numerical methods often involve taking a continuous problem and making it discrete;
e.g. root finding: sequence of iterates
e.g. ODE: discretize and iterate
e.g. PDE: linear algebra and solving linear equations

Algorithm: systematic procedure for solving a problem.

For each method, we might discuss:
- what is it used for
- what is the basis of the method
- how efficient is it / storage requirements / computational time
- under what conditions does the method apply
- when might the method fail to converge, or give incorrect results

- what does the error look like, can we bound it – how fast is convergence

Don't just trust any answer a computer spits out. Think about whether the answer makes sense. Understand the problem (and its likely solutions) as well as possible before using the computer. Think of simple examples you can test the method with.

To develop a numerical method means, in most cases, that one applies a small number of general and simple ideas. One combines these ideas in a simple way with one another and uses information from the mathematics or the physics of the problem.

Advantages of numerical methods:

- Numerical answers can often be obtained when no analytical solution exists.
  e.g. length of $\sin x$ is $\int_0^\pi \sqrt{1+\cos^2 x}dx$
- Sometimes analytical solutions can be difficult to understand (messy formulas) – numerical results might be easier to understand.

Disadvantages of numerical methods:

- Solution is approximate but can be made as accurate as desired
- Numerical methods must use specific input. We cannot get solutions with parameters in them from a numerical computation, so numerical results may not be as general as analytical solutions. It often can be easier to analyze behavior and properties if you have an analytical solution.

Some topics we'll cover include:

- Solving for the roots of nonlinear equations
- Solving systems of Linear Equations
- Interpolation and Approximation of functions
- Approximating derivatives and Integrals
- Solving Ordinary Differential Equations

As with many math problems in the real world, we must turn the problem into one or more equations to solve or functions to minimize or maximize.

Example: Two intersecting mine shafts meet at an angle of 123^o. The straight shaft is 7 ft wide and the entrance shaft is 9 ft wide. What is the longest ladder that can negotiate the turn at the intersection of the 2 shafts?

Calculus Review

- Convergence of a sequence of real numbers: $lim_{n\to\infty} x_n = x$ if for all real $\epsilon > 0$, there exists ($\exists$) a real N s.t. $|x - x_n| < \epsilon$ whenever $n > N$

  example: 1, 3/2, 7/4, 15/8, 31/16,... $\to$ 2.

- C^n function has 0th and first n derivatives continuous.

- Intermediate Value Theorem: If $f \in C[a,b]$ and K is any number between $f(a)$ and $f(b)$, $\exists$ a number $c \in (a,b)$, s.t. $f(c) = K$.

- Derivative: $f'(x_0) = lim_{x\to x_0} \frac{f(x)-f(x_0)}{x-x_0}$ when it exists. Derivative = slope.

- Differentiability implies continuity.

- Extreme Value Theorem: If $f \in C^1[a,b]$, then the maximum and minimum are attained either at the endpoints or where $f' = 0$.

- Taylor Series: If $f \in C^n(a,b)$ and f^{n+1} exists on $[a,b]$ then

  $$f(x) = f(c) + (x-c)f'(c) + \frac{1}{2!}(x-c)^2 f''(c) + ... + \frac{1}{n!}(x-c)^n f^n(c) + R_n(x)$$

  where $c \in [a,b]$ and

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

- Taylor Series for functions of two variables:

  $$\begin{aligned} f(x,y) &= f(a,b) + (x-a)f_x(a,b) + (y-b)f_y(a,b) \\ &+ \frac{1}{2!}((x-a)^2 f_{xx}(a,b) + 2(x-a)(y-b)f_{xy}(a,b) + (y-b)^2 f_{yy}(a,b)) + ... \end{aligned}$$

Example: Work out the Taylor series for $f(x) = e^x$. Error term is $\frac{1}{(n+1)!}x^{n+1}e^{\xi}$. Find n such that we are guaranteed that our approximation is within 10^{-6} of the exact value for $-1 \le x \le 1$.

The more terms we take in the Taylor series, the better the approximation ought to be. The error is smallest near the point around which we expand the series and usually grows as we go further away. (See Figs. 1.1 and 1.2, pp. 5-6 Epperson)

- Rolle's Theorem: If $f \in C^1[a,b]$ and if $f(a) = f(b) = 0$, then $\exists$ a number $c \in [a,b]$ s.t. $f'(c) = 0$.

- Mean Value Theorem (MVT): If $f \in C^1[a,b]$, then $\exists$ a number $c \in (a,b)$ s.t. $f'(c) = \frac{f(b)-f(a)}{b-a}$.
- Generalized Rolle's Theorem: Let $f \in C^n(a,b)$. If f vanishes at the $n+1$ distinct numbers $x_0, x_1, ...x_n$ in $[a,b]$ then $\exists$ a number $c \in (a,b)$ s.t. $f^{(n)}(c) = 0$.
- Mean Value Theorem for Integrals: If f is continuous and integrable on $[a,b]$ then $\exists$ a number $c \in (a,b)$ s.t. $\int_a^b f(x)dx = f(c)(b-a)$.
- Weighted Mean Value Theorem for Integrals: If f is continuous on $[a,b]$ and g is integrable on $[a,b]$ and g does not change sign on $[a,b]$, then $\exists$ a number $c \in (a,b)$ s.t. $\int_a^b f(x)g(x)dx = f(c)\int_a^b g(x)dx$. If $g(x) = 1$, this reduces to the Mean Value Theorem for Integrals.
- Discrete Average Theorem: If f is continuous on $[a,b]$, then $f(\eta) = \sum_{k=1}^n a_k f(x_k)$ where all the $a_k \geq 0$ and $\sum_{k=1}^n a_k = 1$ for some value of $\eta \in [a,b]$.

The MVT allows us to replace function differences with simpler differences. For example, $|\cos(x_1) - \cos(x_2)| \leq |x_1 - x_2|$.

Errors and Some basics

Errors:

Errors can arise in a number of ways when solving problems numerically.

- *Truncation error*: errors caused by the method itself e.g. $e^x \simeq 1 + x + x^2/2 + x^3/6$ but terms are left out.
- *Round-off error*: computers don't maintain an infinite number of digits. I.e., computers use floating point arithmetic

  Example: Find the roots of $x^2 + 62.1x + 1 = 0$.
  The correct values are -0.01611 and -62.084.
  Using the quadratic formula with 4 digits (chopped) gives -0.02 and -62.05.
  The error is caused by *subtractive cancellation.*
  Another example is $x^2 + 3000.001x + 3 = 0$.
  Here, the true solutions are $x = -0.001$ and $x = -3000$.
  The computed solutions are $x = 0.0$ and $x = -3000$.
  We discuss how to deal with this situation soon.

If we want to add many identical numbers, eventually adding new ones will have no influence on the computed sum since they will be small with respect to the current sum. If the values vary in size, it is best to add the small ones first to reduce round-off error.
E.g. $\sum_{i=1}^{100,000} i$
E.g. with 3 digit chopping, add 500+500+8+8+...20 times + 8. The results will not equal 8+8+ 20 times +8 + 500 + 500
E.g. Adding a small number to a large number can result in ignoring the small number $10^{10} + 10^{-10}$
E.g. Evaluate e^{-10} using the Taylor series around $x = 0$. (Two problems arise: subtractive cancellation and it takes many terms until the terms get small). It is better to consider $\frac{1}{e^{10}}$ and use the series for e^{10}.

- *Errors in original data*: Coefficients that are imperfectly known or experimental data that is not exact can lead to errors in the solution, especially if the method or model equations are very sensitive to the input.

- *Blunders*: test runs help but are no guarantee that your equations and coding are correct

- *Propagated error*: error in succeeding steps can depend on earlier errors (e.g. in ODE solvers). If errors are magnified with each step (iteration) as the method proceeds, eventually they will overshadow the true value, destroying its validity. We call such a method unstable. For a stable method, applied to a stable problem, errors made at early points die out as the iterations proceed.

  Example: Consider the iteration schemes:

$$p_{n+1} = (1/3)p_n \quad \text{and} \quad p_{n+2} = (10/3)p_{n+1} - p_n$$

  with $p_0 = 1$ and $p_1 = 0.33333....$ Use 5 digits chopping:

Step	First method	Second method	Exact
2	0.11111	0.11110	0.11111
3	0.037036	0.037000	0.037037
4	0.012345	0.012230	0.012346
5	0.0041150	0.003766	0.0041152
6	0.0013716	0.000323	0.0013717
7	0.00045720	−0.0026893	0.00045725

Significant digits - How many digits in the number have meaning? It is not worth computing a solution to many digits of accuracy if the input is not known very accurately.

The numbers $d_1 d_2 ... d_n d_{n+1} ... d_p$, (with $d_1 \neq 0$) and $d_1 d_2 ... d_n e_{n+1} ... e_p$ agree to n significant digits if $0 < |d_{n+1} - e_{n+1}| < 5$, otherwise they agree to $n-1$ significant digits. Another definition of significant digits: p approximates p_* to t significant digits if t is the largest non-negative integer s.t. $\frac{|p-p_*|}{|p_*|} < 5 * 10^{-t}$.

Computer Issues

- *Exponent underflow* - the computer thinks the number is zero. (The number is so close to zero, the computer can't tell difference.)
- *Exponent overflow* - the computer thinks the number is infinity. (The number is larger than any number the computer can represent.)
- *Machine epsilon* - how small a difference between two values a computer can recognize. $1 + \epsilon = 1$ according to the machine where ϵ is the largest such value.

Absolute error = | true value - approximate value |
Relative error = (absolute error)/|true value|

Find the absolute error and relative error if the true value is 10/3 and the approximate value is 3.333. Find number of significant digits to which these two values agree.

Relative errors can be a bigger problem when the correct answer is close to zero, especially after many operations.
Recall the Example: Find the roots of $x^2 + 62.1x + 1 = 0$.
The correct values are -0.01611 and -62.084.
Using the quadratic formula with 4 digits (chopped) gives -0.02 and -62.05
The absolute errors are 0.00389 and 0.034, respectively.
The relative errors are 0.2415 and 0.00055, respectively.
The small value has smaller absolute error but larger relative error.
It is better here to compute the larger solution using the quadratic formula and use $x_1 x_2 = 1$ (product of roots = c) to compute the smaller value.
Another useful trick that could have been applied in this case is rationalizing the numerator.

- *Accuracy* refers to how closely a computed or measured value agrees with the true value.
- *Precision* refers to how closely computed or measured values agree with each other. One might have very good precision but the value may be quite incorrect. Such an occurence is likely due to systematic error or blunders.
- *Inaccuracy* refers to systematic deviation from truth.

- *Imprecision* describes scattered results.

Error analysis usually does not take into account machine errors. We often will assume perfect computational precision when analyzing error. Machine error is usually handled separately (if at all). The primary tool we will use in analyzing errors is Taylor series.

Stability: We want small changes in initial data to give small changes in the results of our algorithm. (Well-posedness is the concept that this is true for the exact solution). Do errors grow using the algorithm? If so, how fast?

Linear growth $E_n \simeq CE_0 n$ is considered stable

Exponential growth $E_n \simeq C^n E_0$ with $C > 1$ is unstable.

Note: A problem can be found for any numerical method such that the method will "barf" on the problem. So it is very important to make sure the solution found numerically makes sense.

Convergence of solutions: Often our analysis will show that a numerical method will yield improved results if we reduce the step size, h (for example in solving ODEs, and integration). If the method is $O(h^2)$ then halving the step size should improve the approximation by a factor of 4. If is it $O(h^3)$, halving the step size should improve the approximation by a factor of 8 and so on. We can see the convergence rate graphically for such cases by plotting error vs. step size on a log-log plot. For example, if $Error \simeq Ch^4$, then $\ln\ Error \simeq \ln C\ +\ 4 \ln h$, which is a line of slope 4 on the log-log plot.

In determining the order of convergence or convergence rates, some terms are more important than others. We usually deal with these leading order terms or the parts of them that have the greatest effect.

For iteration methods, we consider convergence in terms of how a new iterate compares with the current iterate. If $Error_{i+1} \simeq C\ Error_i$ with $|C| < 1$, the method converges linearly. (If $|C| > 1$, the errors grow and the method diverges – $Error_n \simeq C^n Error_0$). If $Error_{i+1} \simeq C\ (Error_i)^p$, the method is converging with pth order. (Called quadratic convergence if $p = 2$).

Consider an example with values $x_0 = 3.5$, $x_1 = 3.1875$ and $x_2 = 3.0264$ converging to $x^* = 3$. Approximate the order of convergence.

You should be prepared to perform numerical experiments to increase your awareness of computational errors and possible ill-conditioned systems. Such experiments may involve repeating the computations with a different step size or method and comparing the results. We may employ sensitivity analysis to see how the computed solution changes when we change model parameters or input values.

Survey of Simple Methods and Tools

Horner's Rule

Operation count: how many add/subtracts and multiply/divides. Operation count is responsible for how much time it takes to solve a problem numerically (on the computer). It also tells us how much the work grows as we refine our computations.

Useful sums:
$1 + 2 + ... + n = (n)(n+1)/2$ and $1^2 + 2^2 + ... + n^2 = \frac{(n)(n+1)(2n+1)}{6}$
usually we care only about the leading terms $O(n^2)$ for example.

Trick to reduce operations: $x^3 - 10x^2 + 100x - 100 = ((x-10)x + 100)x - 1000$ or $-100 + ((100 + (-10 + x)x)x)$; The first way requires 5 multiplications and 3 additions, while the second requires only 2 multplications and 3 additions.

This is called Horner's rule.

Difference Approximations to the Derivative

Example: Consider $f'(x) \simeq \frac{f(x+h)-f(x)}{h} + O(h)$.
Using Taylor series, we have $\frac{f(x+h)-f(x)}{h} = \frac{f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi) - f(x)}{h}$.
So, $f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{1}{2}hf''(\xi)$.

Consider $f(x) = e^x$.
Use 4 digit chopping to approximate $f'(1)$ with $h = 1, 0.1, 0.05, 0.02, 0.01, 0.005, 0.001, 0.0001$.
Error $\simeq \frac{0.0001}{h} + Mh$ (round-off + truncation) where M is related to $f''(x)$ on the interval and the 0.0001 is from noting that subtracting 2 numbers with value approximately 3 and chopping after 4 digits can gives errors from -0.0001 to +0.0001. (The values of $f(1+h)$ are 7.389,3.004,2.857,2.773,2.745,2.731,2.721,2.718 and $f(1) = f'(1) = 2.718$ giving computed values of 4.671,2.860,2.780,2.750,2.700,2.600,3.000,0.00).

Here, the effect of round-off error creeps up. (This is one of the rare times we'll consider round-off error). If the values of f are off by up to ϵ, then $f'(x)$ can have error up to $\frac{\epsilon+\epsilon}{h} + \frac{1}{2}hf''(\xi)$ so as step size shrinks, the second derivative term shrinks but the round-off term grows. So there is a balance and there is a "best" step size to use (Found approximately by minimizing the error term with respect to h).

Notice that:

$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2f''(x) + \frac{1}{6}h^3f'''(\xi_1)$ and
$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2f''(x) - \frac{1}{6}h^3f'''(\xi_2)$
Thus, $f(x+h) - f(x-h) = 2hf'(x) + \frac{1}{6}h^3[f'''(\xi_1) + f'''(\xi_2)]$
and

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{12}h^2[f'''(\xi_1) + f'''(\xi_2)]$$

or

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6}h^2f'''(\xi)$$

by the Discrete Average Theorem.

Ex: Compare results for $f'(1)$ where $f(x) = e^x$ with both approximations with $h = 1/8$ and $h = 1/16$. Look at the error. Compare $O(h)$ to $O(h^2)$.

Euler's Method

This method is easy to understand, apply and code up. It is not used much (because there are other easy to apply methods that have smaller error), but presents the main ideas of alot of numerical methods for ODEs in a simple form.

- Let $t_i = a + ih$, where $h = \frac{b-a}{N}$, dividing the interval from a to b into N pieces.
- We have $\frac{dy}{dt} = f(y,t)$
- So, $y(t_{i+1}) = y(t_i + h) = y(t_i) + hf(y_i, t_i) + \ldots$ by Taylor's Theorem.
- We ignore the higher order terms, giving us the difference equation:
  $w_{i+1} = w_i + hf(w_i, t_i)$
- (We've called the computed values w and will continue to do so later on in the course).
- Since the first term ignored has order h^2, in one step, the error has order h^2.
- Since going from $t = a$ to $t = b$ takes $\frac{b-a}{h}$ steps, the total error is about $h^2\frac{b-a}{h} \simeq h$.
- It can be shown more carefully that the error

  $$|y_i - w_i| \leq \frac{hM}{2L}[e^{hLi} - 1]$$

  where L is the Lipschitz constant and M is an upper bound for $y''(\xi)$ where $\xi \in [a,b]$.

Pictorially, we are approximating the ODE be a tangent line of length h (in the t direction). We then repeat the procedure from the new (computed) point, $w(t_{i+1})$.

Example: Use Euler's method for $\frac{dy}{dt} = -2t - y$ where $y(0) = -1$ using $h = 0.1$ to approximate $y(0.4)$.

Take a look at Fig. 2.4 Epperson and Tables 2.5 and 2.6 pp. 50-51.

Linear Interpolation

Drawing a line connecting 2 points of $f(x)$ at x_0 and x_1 gives $p_1(x) = \frac{x_1-x}{x_1-x_0}f(x_0) + \frac{x-x_0}{x_1-x_0}f(x_1)$. Check this goes through the points. It can be shown using cleverness and Rolle's Theorem (pp. 54-55 of Epperson) that the error is bounded by $\frac{1}{8}(x_1 - x_0)^2 max|f''(\xi)|$.

It makes sense that this is correct for lines but not parabolas so the error should be related to the second derivative.

The Trapezoidal Rule

- Consider the linear polynomial for 2 points x_0 and x_1 as just described.
- Then $\int_{x_0}^{x_1} f(x)\ dx = \int_{x_0}^{x_1} [\frac{x-x_1}{x_0-x_1}f(x_0) + \frac{x-x_0}{x_1-x_0}f(x_1) + \frac{f''(\xi(x))(x-x_0)(x-x_1)}{2}]\ dx.$
- We can pull out the $f''(\xi(x))$ term because the rest doesn't change sign on the interval. Letting $h = x_1 - x_0$ leads to

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}[f(x_0) + f(x_1)] - \frac{h^3}{12}f''(\xi)$$

Trapezoidal's Rule: One step error is $\frac{h^3}{12}f''(\xi)$ If $h = \frac{b-a}{n}$ then

$$\int_a^b f(x)\ dx \simeq \frac{h}{2}[f_0 + 2f_1 + 2f_2 + \ldots + 2f_{n-2} + 2f_{n-1} + f_n]$$

To compute the error:
n pieces with error each $\frac{h^3 f''}{12}$ for a total error of $\frac{nh^3 f''}{12} = \frac{(b-a)f''h^2}{12}$.

Approximate $f(x) = e^x$ on [0,1] with $h = 1/2$, and $1/4$ and look at error in Table 2.9 p. 64 Epperson.

Work Ex. 2.6 Epperson p. 65. How small should the mesh be to approximate $\int_0^1 e^{-x^2} dx$ with error less than 10^{-3}?

A note on stability of the trapezoidal rule: If $f(x)$ is off by at most ϵ at each value, then the integral computed will have an error at most the original error plus $(b-a)\epsilon$ since the method will change each piece by at most $\frac{1}{2}h(\epsilon+\epsilon)$. Multiply by $\frac{b-a}{h}$ pieces to get $(b-a)\epsilon$.

Nonlinear Equations

Examples of Applications:

- When does a nonlinear function equal a certain value? E.g., For a pharmaceutical to be effective, its concentration in the blood must stay above a certain value. One would like to know at what time the concentration will hit that value so another dose can be given.
- In free-all with resistance, can we approximate the coefficient of (linear) drag from the model and data? (Assuming v, m, g and t are known; $v(0) = 0$).

$$\frac{dv}{dt} = g - \frac{cv}{m} \rightarrow v = \frac{gm}{c}(1 - e^{ct/m})$$

- For what value(s) of x are 2 nonlinear functions equal?
- Minimizing the cost to build a road where it costs different amounts through different parcels of land. See story, p. 80 Epperson

For the methods described in this section, we begin mathematically by subtracting the functions or bring all terms to one side to get $f(x) = 0$. A solution to such an equation is called a *root*.

Ex. Find where $x = \tan x$ on $[4, 4.5]$. Use $x - \tan x = 0$.

Bracketing a root - Graphing can give us an estimate of a root, but we can miss when two roots are close to one another or think there is a root when f is close to zero. If f is continuous and we know that $f(a)f(b) < 0$ then there must be at least one root on $[a, b]$, by the IVT.

How can we tell that a continuous function $f(x)$ has exactly one root on an interval $[a, b]$? Show that $f(a)f(b) < 0$. This guarantees at least one root. If, in addition, $f'(x)$ does

not change sign on $[a, b]$, $f(x)$ is monotonic on the interval and so can have at *most* one root on $[a, b]$. Thus, $f(x)$ has exactly one root on $[a, b]$.

Example: Show $f(x) = x^3 - x - 1$ has exactly one root on $[1, 2]$.

Bisection: Work through a few steps to approximate a root of $f(x) = x^3 + x^2 - 3x - 3$ on $[1, 2]$ How do we know there's a root in the interval? IVT ($\sqrt{3}$ is a root);

Start with x_{left} and x_{right}; $x_{new} = \frac{1}{2}(x_{left} + x_{right})$
Keep one previous point such that $f(x_{new}) * f(x_{old}) < 0$, i.e., they have opposite sign.
Continue until $x_{right} - x_{left}$ is smaller than the desired tolerance and/or $f(x_{mid})$ <TOL.

Some issues to note about bisection:

- Bisection always converges (if f is continuous and you have starting values such that $f(a)f(b) < 0$ where a and b are the starting x values at the endpoints). Bisection is a *global* method.
- Length of interval: $L_n = \frac{1}{2^n} L_0$ after n steps (or iterations). So we can compute the number of steps we'll need to reach our error tolerance. Take the final answer to be middle of final interval. This implies: $\frac{1}{2^n} L_0 * \frac{1}{2}$ <TOL $= \epsilon$. $\implies n \geq -1 + \log_2 \frac{L_0}{\epsilon}$. So for the initial interval [1,6], how many steps would it take to be sure our approximation is within 10^{-3} of the exact value? ($n = 12$).
- Early estimates may be better than later ones. Convergence isn't necessarily closer with each step.
- This method will find exactly one solution (it is very robust) on the interval, so beware if there is more than one root on the interval.
- We might prefer a tolerance where $f <$ TOL but f might be very flat.
- Bisection is not as fast as other methods but is sometimes used to get somewhat close to the root and then we let another method take over.
- Roots at large values of x may be found solving using the substitution $y = 1/x$.
- Bisection will not be able to find roots that are even multiple roots since the function does not change sign there.

If we want to estimate $\sqrt[3]{25}$ we can solve $x^3 - 25 = 0$. Often it's nicer to write expressions as polynomials. In theory, you can then do the computations without a "fancy" calculator.

Order of convergence: If $|e_{n+1}|$ approaches $K|e_n|^p$ as $n \to \infty$ we say the method is of order p. Errors usually don't decrease quite this quickly until we are near the root for most methods.

Bisection has $|e_{n+1}| \simeq \frac{1}{2}|e_n|^1 \implies$ Bisection is first order - or of order 1.

Secant Method and Regula Falsi

If $f(x_0) * f(x_1) < 0$, we can draw a line connecting $(x_0, f(x_0))$ and $(x_1, f(x_1))$ and take intersection of this line with the x-axis as the next iterate.

The slope of line through $(x_0, f(x_0))$ and $(x_1, f(x_1))$ is the same as slope of line through $(x_1, f(x_1))$ and any (x, y) on the line, so

$$\frac{y - f(x_1)}{x - x_1} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

where $y = 0$ when $x = x_2$, giving $x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$ or

$$x_{n+2} = x_{n+1} - f(x_{n+1})\frac{x_{n+1} - x_n}{f(x_{n+1}) - f(x_n)}$$

Work a couple of steps for $x^2 - 3 = 0$ on $[1, 2]$. If we keep the old value of x such that we always bracket the root, the method is called Regula Falsi - or method of false position. This is more stable but might have slower convergence.

We hope that our values of x_{n+2} get close to some value x_*, where $f(x_*) = 0$. If so, the method converges. This kind of method is called a *fixed point method* since when we continue iterating, we hope to get to a number that when entered on the right (i.e., if $x_n = x_{n+1} = x_*$) yields the same value on the left.

Some issues to note about Secant and Regula Falsi:

- The closer the function is to linear near the root, the quicker the convergence will be.
- The method can also be written $x_{n+2} = \frac{x_n f(x_{n+1}) - x_{n+1} f(x_n)}{f(x_{n+1}) - f(x_n)}$ but this is more prone to round-off error. (Subtractive cancellation).
- The new value, x_{n+2}, is not guaranteed to be in the interval bounded by x_n and x_{n+1} for the Secant method. On the other hand, x_{n+2}, must be bounded by x_n and x_{n+1} for Regula Falsi.

We'll skip this proof. If we have time, maybe we'll come back to it later. Section 3.10.3 has a very nice analysis

Error analysis (for the Secant method): Notice that this method is of the form

$$x_{n+2} = \phi(x_{n+1}, x_n)$$

This is a two-point fixed point method. The new value depends on two previous values.

- Let $\phi(u, v) = u - \frac{f(u)(u-v)}{f(u)-f(v)}$ if $u \neq v$.
- As $u \to v$ for C^1 functions, this becomes $\phi(u, u) = u - \frac{f(u)}{f'(u)}$ and as long as $f'(x_*) \neq 0$ at the point of interest (where $f(x_*) = 0$), $\phi(x_*, x_*) = x_*$. I.e., we have a fixed point. That is, if we start with $x_1 = x_2 = x_*$, we get the new value $\phi(u, u)$ also $= x_*$.

$$\phi(u, x_*) = u - \frac{f(u)(u - x_*)}{f(u) - 0} = x_* = \phi(x_*, v)$$
$$\text{and} \quad \phi_u(u, x_*) = 0 = \phi_v(x_*, v) = \phi_{uu}(u, x_*) = \phi_{vv}(x_*, v)$$

- So

$$\begin{aligned} \phi(x_* + p, x_* + q) &= \phi(x_*, x_*) + p\phi_u(x_*, x_*) + q\phi_v(x_*, x_*) \\ &\quad + \tfrac{1}{2}[p^2\phi_{uu}(x_*, x_*) + 2pq\phi_{uv}(x_*, x_*) + q^2\phi_{vv}(x_*, x_*)] + O(pq)^3 \\ &= x_* + 0 + 0 + \tfrac{1}{2}[0 + 2pq\phi_{uv}(x_*, x_*) + 0] + O(pq)^3 \end{aligned}$$

- So, if e_n is the error at the nth step $x_* - x_n$, we have

$$\begin{aligned} e_{n+2} &= x_* - x_{n+2} \\ e_{n+2} &= x_* - \phi(x_{n+1}, x_n) \\ &= x_* - \phi(x_* - e_{n+1}, x_* - e_n) \\ &= x_* - (x_* + e_{n+1}e_n\phi_{uv}(x_*, x_*) + O(e_{n+1}e_n)^3) \\ \implies e_{n+2} &\simeq Ce_{n+1}e_n \end{aligned}$$

- Since we have convergence if the error gets smaller (exponentially) with each iteration, we have convergence if $|Ce_{n+1}| < const < 1$, i.e., if $e_{n+2} < e_{n+1}$.
- For $C = 1$ we can easily compute the order of convergence.
  Let $e_{k+1} \simeq e_k^\rho$.
  This gives $e_{k+1} \simeq e_k^\rho$ but we also have $e_{k+1} \simeq e_k e_{k-1} \simeq e_k e_k^{1/\rho}$.
  So, $e_k^\rho \simeq e_k e_k^{1/\rho}$.
  Thus, $\rho = 1 + 1/\rho \implies \rho^2 - \rho - 1 = 0 \implies \rho = \frac{1 \pm \sqrt{5}}{2}$ but only the + makes sense.
  So convergence order is about 1.62.

Regula Falsi This is the same as Secant Method but we always keep the previous x value such that x_{old} and x_{new} bracket the root. I.e. $f(x_{old}) * f(x_{new}) < 0$. Convergence is more difficult to analyze here, but in general Regula Falsi is slower, but more robust than Secant Method. This is a common trade-off in numerical methods.

Problems can be constructed such that bisection converges more quickly than Secant or Regula Falsi. This is true especially when f is very flat near the root. E.g. $f(x) = x^{10} - 1$ on $[0, 1.3]$.

Newton's Method

Idea: Draw a line tangent to $y = f(x)$. Where this tangent line crosses the x axis is our next iterate.

Derivation: $(x_0, f(x_0))$ is our first guess and the slope there is $f'(x_0)$.
The equation of the tangent line is $y - f(x_0) = f'(x_0)(x - x_0)$.
When $y = 0$, we have $x = x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ or, in general,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Work a few steps for $x^2 = 3$ with $x_0 = 1$.

Note the relationship of Newton's method to Secant and Regula Falsi. I.e., how
$x_{n+2} = x_{n+1} - \frac{f(x_{n+1})(x_{n+1} - x_n)}{f(x_{n+1} - f(x_n)} \longrightarrow x_{n+1} - \frac{f(x_{n+1})}{f'(x_{n+1})}$ as $x_{n+1} \to x_n$

Comment: We can also use Newton's Method to find complex roots but we must start with a complex guess.

Another derivation of Newton's method is to write the Taylor series for $f(x)$ around x_n to get $f(x) = f(x_n) + (x - x_n)f'(x_n) + ...$ Truncate (ignore $(x - x_n)^2$ and higher order terms) and let x_{n+1} be where this line crosses the x-axis.

This yields $0 = f(x_n) + (x_{n+1} - x_n)f'(x_n)$ or $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$.

Some issues to note about Newton's Method:

- $f'(x)$ may be difficult or messy to calculate.
- There will be problems if $f'(x) \simeq 0$ near $x = x_*$, since the denominator will be zero. Convergence can be slow for very flat functions. We'll discuss this further later. Consider Newton's method on $f(x) = e^x - x - 1$, which has a double root at $x = 0$.

- There can be problems when there are big changes in $f'(x)$ near the root. Consider $f(x) = x^{10} - 1$ with initial guess $x_0 = 1/2$, then $x_1 = 52$. See e.g., p. 91 Epperson Figs 3.3 and 3.4. These demonstrate the importance of a good initial guess and the robustness of bisection.

- If x_0 is not "close enough to the root", the method may not converge or may converge to a far away root. For example, if there is an inflection point at x_*. E.g. $f(x) = -\frac{x^3}{4} + \frac{5x}{4}$ with $x_0 = 1$ oscillates. Draw examples.

- Newton's method can oscillate near a minimum that is not near a root.

- The crucial assumption is that the quadratic term of the Taylor series (and higher order terms) can be ignored without adverse consequences.

Two fundamental numerical methods ideas:

1. Replacing a general function by a simpler function and doing the computation exactly on the simpler function. (e.g. $f(x)$ by a line).

2. Given an expression with something simple plus remainder, we generate a numerical approximation by dropping the remainder. (e.g., Taylor series derivation of Newton's method).

Stopping Newton's Method: The x_n's get close to one another if the method converges. However, the $f(x_n)$'s might be larger if $|f'(x_*)|$ is large, so sometimes, rather than using $|x_n - x_{n-1}| < \epsilon$, people use $|f(x_n)| + |x_n - x_{n-1}| < \epsilon$
We'll save the error analysis for after fixed point iteration.

Muller's Method We'll only discuss the idea of this method.

- We derive a second degree polynomial (a quadratic) that fits 3 points near the root $(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))$.

- Where this parabola crosses $y = 0$ is the new point.

- Then take the 3 closest points to the root and repeat.

- There are 2 points where the parabola crosses $y = 0$ and we take the one closest to x_0 for the current iteration.

Fixed Point Iteration

Newton's method is a one-point, fixed point iteration method. (Secant method and Bisection were 2-point methods - i.e., you need two old guesses to get a new one with those. Muller's method is a 3-point method). Let's discuss some ideas concerning one point methods. They are the easiest to analyze.

A one-point, fixed point method can be written: $x_{n+1} = g(x_n)$ for $n = 1, 2, 3, ...$ When does it converge to a fixed point x_* such that $x_* = g(x_*)$?

$x_* = g(x_*)$ should be equivalent to $f(x) = 0$.

Consider $f(x) = x^2 - 2x - 3$. Obviously, the roots are $x = -1, 3$.
Derive $g_1(x) = \sqrt{2x+3}$ and let $x_0 = 4$ p. 55 of Gerald and Wheatley
Derive $g_2(x) = \frac{3}{x-2}$ and let $x_0 = 4$ p. 56 of Gerald and Wheatley
Derive $g_3(x) = \frac{x^2-3}{2}$ and let $x_0 = 4$ p. 56 of Gerald and Wheatley
Show pictorially what happens in these cases and what the method looks like in terms of $y = g(x)$ and $y = x$. p. 57 of Gerald and Wheatley.

Theorems:

- If g is continuous in $[a, b]$ and $g(x) \in [a, b]$ then g has a fixed point in $[a, b]$.
  Proof by IVT. Let $h(x) = x - g(x)$ and note $h(a) < 0$ and $h(b) > 0$.

- If, in addition, g' exists on $[a, b]$ and $|g'(x)| < 1$ and $g(x) \in [a, b]$, then g has a unique fixed point in $[a, b]$.
  Proof: If there are 2 fixed points $x = p$ and $x = q$ then $|p - q| = |g(p) - g(q)| = |g'(\xi)||p - q|$ by MVT $< |p - q| \Rightarrow$ contradiction.
  Another proof: Again let $h(x) = x - g(x)$, then $h(a)h(b) < 0$ and $h'(x) = 1 - g'(x) > 0$ on $[a, b]$ so $h(x)$ has exactly one root on $[a, b]$.

  Idea: If the interval is from $x = a$ to $x = b$ then if $g(a)$ is greater than a and $g(b)$ is less than b, there must be root in the interval. If $g(x)$ cannot go up as fast as $y = x$ (since $|f'(x) < 1|$), then once the curves cross (at the root) $g(x)$ cannot catch up to $y = x$ again.

- If $g(x)$ and $g'(x)$ are continuous on an interval around a root of $x = g(x)$ and if $|g'(x)| < 1$ $\forall x$ in the interval, then $x_{n+1} = g(x_n)$ converges to the root if x_0 is in the interval.

  Convergence may occur anyway but this guarantees convergence.

  Proof: $|x_* - x_n| = |g(x_*) - g(x_{n-1})| = |g'(\xi)|\ |x_* - x_{n-1}| \leq k|x_* - x_{n-1}|$ where $k < 1$. This is called first-order or linear convergence with coefficient k indicating how fast, e.g, 1/3, 0.9 so error shrinks and $|x_* - x_n| < k^n|x_* - x_0| \to 0$ so $x_n \to x_*$.

Analyze for the 3 problems on previous page.

An error bound after n steps is $|x_* - x_n| < \frac{k^n}{1-k}|x_1 - x_0|$. From geometric series. We have

$$\begin{aligned} |x_n - x_*| &= \lim_{m\to\infty} |x_n - x_m| \\ &= \lim_{m\to\infty} |(x_n - x_{n+1}) + (x_{n+1} - x_{n+2}) \ + \ \ldots \ + \ (x_{m-1} - x_m)| \\ &\leq k^n|(x_0 - x_1) + k(x_0 - x_1) + k^2(x_0 - x_1) + \ldots + k^{m-n}(x_0 - x_1)| \\ &\leq \frac{k^n}{1-k}|x_0 - x_1| \end{aligned}$$

Notice that if $g'(x_*) > k > 1$, we have

$$|x_* - x_n| = |g(x_*) - g(x_{n-1})| = |g'(\xi)| \ |x_* - x_{n-1}| \geq k|x_* - x_{n-1}|$$

so the error grows and sequence of iterates must diverge.

So, to prove convergence, we need to prove only that there is a root and $|g'(x_*)| < 1$. Ex. $x_{n+1} = \frac{x_n^2 - 1}{3}$ on $[-1, 1]$. $g(-1) = g(1) = 0$ and $g'(x) = \frac{2x}{3}$. Thus, the iteration scheme converges to root of $x^2 - 3x - 1 = 0$.

Note that the closer $g'(x)$ is to zero near the root, the faster the fixed point method converges.

Error analysis of Newton's Method:

- $x_{n+1} \ = \ x_n - \frac{f(x_n)}{f'(x_n)} \ = \ g(x_n)$
- Here $g'(x_*) \ = \ \frac{f(x_*)f''(x_*)}{f'(x_*)^2} = 0$
- The method seems linear but since $k = 0$ it's really better than linear.
- Look at $|x_{n+1} - x_*| = |g(x_n) - g(x_*)|$.
- Since $x_n = x_* + (x_n - x_*)$ we have

$$g(x_n) = g(x_*) + (x_n - x_*)g'(x_*) + \frac{g''(\xi)}{2}(x_n - x_*)^2$$

So

$$\begin{aligned} |x_{n+1} - x_*| &= |g(x_*) + (x_n - x_*)g'(x_*) + \frac{g''(\xi)}{2}(x_n - x_*)^2 - g(x_*)| \\ &= \frac{g''(\xi)}{2}(x_n - x_*)^2 = k(x_n - x_*)^2 \end{aligned}$$

$$\text{This becomes, } = \frac{f''(x_*)}{2f'(x_*)}(x_n - x_*)^2$$

- So there is second order convergence (or quadratic convergence).
- Note that if $g''(x_*) = 0$, we could take one more term of the Taylor series and show that convergence is cubic (third order).

  If $g'(x_*) = g''(x_*) = ... = g^{(p-1)}(x_*) = 0$, then

$$|x_{n+1} - x_*| = |g(x_n) - g(x_*)| = |g(x_* + (x_n - x_*)) - g(x_*)|$$

$$= |g(x_*)+(x_n-x_*)g'(x_*) + ... + \frac{(x_n - x_*)^{p-1}}{(p-1)!}g^{(p-1)}(x_*) + \frac{(x_n - x*)^p}{p!}g^{(p)}(\xi) - g(x_*)|$$

$$= |\frac{(x_n - x*)^p}{p!}g^{(p)}(\xi)|$$

  So convergence is of order p.

Note: For multiple roots, $f'(x_*)^2 = 0$ (in the denominator) so $g'(x_*)$ is not 0.
(Work it out for $f = (x - x_*)^p Q(x)$).
Thus, convergence is reduced to linear with coefficient $1 - \frac{1}{p}$ Look at Table 1.8 p. 77. Gerald and Wheatley

If you know the multiplicity of the root you are interested in, use

$$x_{n+1} = x_n - p\frac{f(x_n)}{f'(x_n)}$$

where p is the multiplicity of the root. To show this, let $f = (x - x_*)^p Q(x)$ where $Q(x_*) \neq 0$ and plug into $g'(x_*) = 1 - p(1 - \frac{ff''}{f'^2})$ and take the limit as $x \to x_*$. This shows that $g'(x_*) = 0$, so the method is quadratically convergent.

If you know only that the root is a multiple root but not its multiplicity, one can perform Newton's method on $\mu(x) = \frac{f(x)}{f'(x)}$. This can be pretty messy.

Once again, consider Newton's method on $f(x) = e^x - x - 1$, which has a double root at $x = 0$.

```
Initial iterate is 1.000000
i     x         f(x)
0   1.000000   0.718282
1   0.5819767  0.2075957
2   0.3190550  0.0567720
```

```
3   0.1679962  0.0149359
4   0.0863489  0.0038377
5   0.0437957  0.0009732
6   0.0220577  0.0002451
7   0.0110694  0.0000615
8   0.0055449  0.0000154
9   0.0027750  0.0000039
10  0.0013881  0.0000010
11  0.0006942  0.0000002
12  0.0003472  0.0000001
13  0.0001736  0.0000000
14  0.0000868  0.0000000
15  0.0000434  0.0000000
16  0.0000217  0.0000000
17  0.0000108  0.0000000
18  0.0000054  0.0000000
19  0.0000027  0.0000000
20  0.0000014  0.0000000
21  0.0000007  0.0000000
22  0.0000003  0.0000000
23  0.0000002  0.0000000
24  0.0000001  0.0000000
25  0.0000000  0.0000000
26  0.0000000  0.0000000
27  0.000000   0.000000
Solution is x = 0.000000 function = 0.000000 iterations = 27
```

Notice that convergence is linear with $C = 1/2$. Newton's method on $\frac{f(x)}{f'(x)}$ is better:

```
Initial iterate is 1.000000
i     x       f(x)
0  1.000000   0.418023
1 -0.2342106 -0.1216724
2 -0.0084583 -0.0042351
3 -0.0000119 -0.0000059
4  0.0000000  0.0000000
5  0.000000   0.000000
Solution is x = 0.000000 function = 0.000000 iterations = 5
```

Two roots that are near one another can get Newton's method into an infinite loop (can happen also with an unlucky starting guess) or have the iterates fly off to infinity. Draw figures. There is no good way to deal with these. If you can find one of the roots (x_{*1}), you can try to use deflation (dividing $f(x)$ by $x - x_{*1}$).

Accelerating Convergence Idea: If we know the order of convergence, say $e_{n+1} = Ce_n^p$, then given 3 consecutive iterates using the method, we can "improve" the estimates, calculating a new set of iterates using
$\frac{x_{n+2} - x_*}{(x_{n+1} - x_*)^p} \simeq Const \simeq \frac{x_{n+1} - x_*}{(x_n - x_*)^p}$ and solve for x_*.
Work it out for $p = 1$ to get $x_* = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}$.

This is called **Aitken's method** when we use x_0, x_1, x_2 to get $\tilde{x}_0$;
x_1, x_2, x_3 to get $\tilde{x}_1$;
x_2, x_3, x_4 to get $\tilde{x}_2$;etc
It is called **Steffenson's method** when using x_0, x_1, x_2 to get $\tilde{x}_0$; then $\tilde{x}_1 = g(\tilde{x}_0)$ and $\tilde{x}_2 = g(\tilde{x}_1)$ to get $\tilde{\tilde{x}}_0$ etc.

Consider Table on p. 143 of Epperson. Good convergence (rates) are obtained for:

- Newton's method with $x_{n+1} = x_n - p\frac{f(x_n)}{f'(x_n)}$
- Newton's method with $u(x) = \frac{f(x)}{f'(x)}$
- Newton's method with Aitken's method
- Secant method with $u(x) = \frac{f(x)}{f'(x)}$

but final result is not as good as previous non-multiple root cases. Why? Geometry of multiple roots - computational issues with floating point arithmetic. Since the curve is flat, $f(x_n)$ appears closer to the root than it is. p. 144 Fig. 3.11 Epperson.

For finding all roots to a polynomial, first find one root, then *deflate* using synthetic division before finding the next root so you don't keep getting same root. Round-off errors can build up and so later roots might not be very accurate.

Another method for dealing with polynomials is called Bairstow's method. It is used for factoring a polynomial into quadratic pieces which can then be solved analytically using the quadratic formula.

Hybrid methods: Use Secant method until an iterate falls outside the bracketed interval. Then use Bisection, then back to Secant until is happens again. This way you can get global convergence but speedier than bisection. (Brent's method).