Numerical Analysis

Contents

1. Error Analysis

1.1. **Exact and approximate numbers, rounding.** In calculations and measurements in the sciences, we often refer to real numbers as **exact** or **approximate**. In pure mathematics, an exact number does not differ from a real or complex number. Sometimes it is sufficient, desireable, or easier to use a different number, called an **approximate number**, in place of an exact number. If x is an exact number, and if $\overline{x}$ is an approximation to x, then the error is $e_x = x - \overline{x}$. That is the error is the difference between the exact and approximate values. Sometimes an exact number is called an **exact value** or a **true value**.

Example 1. *If we regard* $\sqrt{2}$ *as an exact value, a calculator may use* $1.414\,213\,6$ *in its place. Then the error in this approximation is*

$$\sqrt{2} - 1.414\,213\,6.$$

Note that an approximate value for this error is $-3.762\,690\,5 \times 10^{-8}$.

Example 2. *When a carpenter measures a block of oak that is exactly* 10.4563892 *inches long, she uses a tape measure that has 32nds of an inch marked but no finer marks. She estimates that the board is* $10\frac{14.6}{32}$ *inches long, and so the error in her measurement is* $10.4563892 - (10 + 14.6/32) = 0.000\,139\,2$. *Note that in this case, the value of the error can be expressed exactly.*

Sometimes it is useful to think of an approximate number as a variable real number, consisting of the approximate number $\overline{x}$ and a known maximum error E_x, where one knows that the true value x of the number lies within the maximum error of $\overline{x}$. That is,

$$\overline{x} - E_x \le x \le \overline{x} + E_x \ ,$$

or alternately

$$\begin{aligned} |x - \overline{x}| &\le E_x \ , \text{ or} \\ |e_x| &\le E_x \ . \end{aligned}$$

That is the case when one uses calculators or computers to perform mathematical calculations or doing experiments in the sciences. Usually one does not know the true value of most quantities, only an approximate value. Such an **error bound** is very important.

In a calculator or computer, it is often the case that real numbers produced by computations are given as decimal approximations. For example, a computer may calculate a quantity such as $2^{1.7}$ accurate to a certain number of decimal places. Often the computer does the calculation accurate to more decimal places and then performs either a **rounding** or a **truncating**. To see what this means, suppose that the computer initially calculates $x = 2^{1.7} \approx 3.249\,009\,585\,424\,942\,09$ and then will display the answer to 8 significant digits (in this case, 7 digits past the decimal point). If the computer truncates the result, it merely displays the first 8 significant digits: $\overline{x} = 3.249\,009\,5$. If it instead rounds the result, the answer is $\overline{x} = 3.249\,009\,6$, because the next digit 8 is ≥ 5. In these cases, the errors are:

$$\begin{aligned} \text{truncation} \ &: \ e_x = x - \overline{x} \approx 0.000\,000\,085\,424\,942\,09 \\ \text{rounding} \ &: \ e_x = x - \overline{x} \approx -0.000\,000\,014\,575\,057\,91 \end{aligned}$$

In general, for truncation, one can always assume that $0 \le e_x < 10^{-n}$ if $\overline{x}$ is given as a decimal with n digits to the right of the decimal point. For rounding, one can always assume that $-5 \times 10^{-n-1} \le e_x < 5 \times 10^{-n-1}$ if $\overline{x}$ is given as a decimal with n digits to the right of the

decimal point. These inequalities imply that $|e_x| < 10^{-n}$ for truncation and $|e_x| \leq 5 \times 10^{-n-1}$ for rounding.

1.2. **Floating Point Numbers.** How do computers actually store numbers? Typically computers use a binary (base 2) or hexadecimal system (base 16) for integers, but of typically these numbers are converted to base 10 for display purposes. For example, the base 2 representation for the decimal number 3286 is 110011010110_2, because $2^{11} + 2^{10} + 2^7 + 2^6 + 2^4 + 2^2 + 2^1 = 3286$. Note that by this observation, integers between 0 and 3286 can certainly be stored in a computer with binary number system using 12 bits, that is by 12 choices of 1 or zero. In fact every integer from 0 to $111111111111_2 = 2^{12} - 1 = 4095$ can be represented using 12 bits. After a little bit of practice, you can develop a method for converting a base 10 number into a base 2 number. Examples include:

$$\begin{aligned} 11 &= 2^3 + 2^1 + 2^0 = 1011_2 \\ 3474 &= 2^{11} + 2^{10} + 2^8 + 2^7 + 2^4 + 2^1 = 110110010010_2 \\ 3.7 &= 2^1 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-11} + 2^{-12} + ... \\ &= 11.10\overline{1100}_2 \end{aligned}$$

In each case of finding the binary expansion of a number x, you should first find the highest power of 2, say 2^r, that is $\leq x$, and that is the first digit. Subtract that and start again with $x' = x - 2^r$. For numbers with fractional parts, you can also compute the binary expansion using long division: $3.7 = \frac{37}{10} = \frac{100101_2}{1010_2}$, and we have

$$\begin{array}{r} 11.101100 \\ 1010\ \overline{)\,100101.000000} \\ \underline{10100} \\ 10001 \\ \underline{1010} \\ 1110 \\ \underline{1010} \\ 10000 \\ \underline{1010} \\ 1100 \\ \underline{1010} \\ 10000 \end{array}$$

For general real numbers, we must allow for positive and negative parts as well as fractional parts of the number. The **floating point representation** is a method of using a certain number of bits to store a general (approximation to a) real number. This representation is very similar to scientific notation.

Recall that in scientific notation, a number such as -0.0034667932 is rewritten as $-3.4667932 \times 10^{-3}$. The format of a number in scientific notation is a nonzero digit, followed by a decimal point, followed by a sequence of digits (number determined by the number of digits of accuracy), all multiplied by a power of 10. The same idea allows computers to store a wide range of real numbers in the floating point representation.

The different ingredients to floating point representation are:

(1) the **base** of the number system (given, b)
(2) the **sign** of the number (determined by $s = 1$ or 0, with the sign given by $(-1)^s$
(3) the **exponent** (e)
(4) the **fraction** (also called significand or mantissa, f), consisting of a certain maximum number of digits
(5) the **bias** (given, B)

All of these items can be given as a nonnegative integer. The real number that is described by the nonnegative integers (f, e, s) is

$$(-1)^s \, f b^{-t} b^{e-B}.$$

The number t determines the placement of the decimal point (or really **radix point** for general bases) in the significand. The bias subtracts a certain amount from the exponent in order to allow for both positive and negative exponents. For example, if the base is 10, and the numbers have a maximum of 5 digits, all to be placed after the decimal point, and the bias is 4, we have

$$\begin{aligned} (f, e, s) &= (34201, 12, 1) \\ &\leftrightarrow -0.34201 \times 10^8. \end{aligned}$$

The standard floating point system used in most modern computers is denoted IEEE754. This system was developed in 2000 and approved in June, 2008, and it is used in almost every modern computing device. It is a tad more complicated than the general system described above. Here are the details. First of all, the base is $b = 2$, and each digit of the integers f, e, s corresponds to a bit in computer memory. The radix point in this system is always placed between the first and second digits in the significand. Because the base is 2, observe that the first digit is **always** 1, except in the case where the number is zero. For that reason, in the IEEE754 system, the first digit is implied to be 1, and no bit is wasted on this; instead only the digits after the radix point are stored. The number of different bits used in various versions of this system are described in the table below:

precision type	f	e	s	bias
half-precision (16 bit)	10	5	1	15
single precision (32 bit)	23	8	1	127
double precision (64 bit)	52	11	1	1023
quadruple precision (128 bit)	112	15	1	16383

So, for example, in half-precision, the binary 16 bit number 1001011101101001 means (in base 2)

$$-1.1001011101 \times 2^{10100-1111},$$

which in base ten means

$$\begin{aligned} & -\left(1 + 2^{-1} + 2^{-4} + 2^{-6} + 2^{-7} + 2^{-8} + 2^{-10}\right) \times 2^{2^4+2^2-15} \\ = \; & -\frac{1629}{1024} \times 2^5 = -50.\,906\,25. \end{aligned}$$

Note that each nonzero half-precision number carries a total of 11 (10 bits for f and 1 extra for the implicit leading digit) significant binary digits. Since $2^{11} = 2048$, this means that all decimal numbers from 000 to 999 are covered, so that this translates to a decimal representation with 3 significant digits. The smallest possible exponent is -15, and since

$2^{15} = 32\,768$, this translates to a smallest decimal exponent of -4. Similarly, since the largest possible exponent is 16 and since $2^{16} = 65\,536$, the largest possible decimal exponent is $+4$. The range of numbers able to be represented by half-precision IEEE754 is from $-(2 - 2^{-10}) \times 2^{15} = -65\,504$ to $(2 - 2^{-10}) \times 2^{15} = 65504$.

Now, there are a few additional features added to IEEE754. They are:

(1) An exponent $e = 0$ (ie 0 for each bit) does not assume that the first digit is 1; so in this case the f is the whole significand. The point of this is to allow the possibility of the number being zero. Note that this means that such numbers only have 10 significant binary digits (still leading to 3 significant decimal digits).
(2) The exponent $e = 11...1$ (all ones) along with a mantissa $f = 000...0$ (all zeros) is the special value ∞. (Could be $\pm\infty$ depending on s).
(3) There are also values reserved for NaN (Not a Number), and in fact two categories of those: SNaN (Signaling NaN), and QNaN (Quiet NaN). These values are $e = 11...1$ (all ones) along with a mantissa $f = 00...01$ through $f = 011...1$ (SNaN) or $f = 100...0$ through $11...1$ (QNaN).

Similar analysis to the above may be applied to single, double, and quadruple precision floating point numbers.

Another interesting topic coming from the IEEE754 standards are the classification of the five most common rounding algorithms for these floating point numbers in calculations. The point is that calculations with floating point numbers involve algorithms that can produce a new floating point number with arbitrary accuracy (for a given input, eg. division); however, the result of the calculation must be stored as a new floating point number. So how should that number with fewer significant figures be chosen? The algorithms to make that choice are:

(1) **Round to nearest, ties away from zero (standard)**
This is the standard method of using the next decimal (or binary) place; if that digit is 5 or higher (or 1), the absolute value of the number is rounded up.
(2) **Round to nearest, ties to even**
If the calculation produces a number that is exactly half-way between two possible floating point numbers (so that if it is a decimal number, the numbers following would be 50000...; in the binary case 10000...), then the number is rounded to the nearest even number. For example, for a decimal number, 5687435861.5000 rounds to 5687435862, but 5687435862.5000 rounds to 5687435862. For a binary number, 010100100101001.100000 rounds to 010100100101010, but 010100100101000.100000 rounds to 010100100101000.
(3) **Round toward zero (truncation)**
Here we just cut off the expansion and stop.
(4) **Round toward +infinity (ceiling, rounding up)**
like it says....
(5) **Round toward -infinity (floor, rounding down)**
like it says...

1.3. **Relative Error and Propagation of Error.** Suppose that x is an exact number, and let $\overline{x}$ be the corresponding approximate number. Then $e_x = x - \overline{x}$ be the error in x. We define the **relative error in** x to be

$$\varepsilon_x = \frac{e_x}{\overline{x}}.$$

From this equation and a little algebra, note that

$$\overline{x} = \frac{x}{1+\varepsilon_x}.$$

If $\overline{x}$ is a digital floating point value of the exact number x such that f has m digits with nonzero leading digit, note that

$$\overline{x} = (-1)^s f b^{-t} b^{e-B},$$

and

$$|e_x| \leq b^{-t} b^{e-B} = b^{e-B-t},$$

depending on the method of rounding. Then the relative error ε_x satisfies

$$|\varepsilon_x| \leq \frac{1}{f b^{-t} b^{e-B}} b^{-t} b^{e-B} = \frac{1}{f} \leq b^{-m+1},$$

where that last inequality is achieved when $f = 100...0$. Thus the best bound the relative error is dependent only on the base b and the number m of significant digits in the expression for $\overline{x}$.

The simplest example is the consideration of what happens when two approximate numbers $\overline{x}$ and $\overline{y}$ are added. In other words, we declare that $\overline{x+y} = \overline{x} + \overline{y}$ is the approximation to $x + y$. The error e_{x+y} in the sum is

$$\begin{aligned} e_{x+y} &= (x+y) - \overline{(x+y)} \\ &= (x - \overline{x}) + (y - \overline{y}) \\ &= e_x + e_y, \end{aligned}$$

and thus the relative error satisfies

$$\begin{aligned} \varepsilon_{x+y} &= \frac{e_x + e_y}{\overline{(x+y)}} \\ &= \frac{\overline{x}\varepsilon_x + \overline{y}\varepsilon_y}{\overline{x} + \overline{y}} \\ &= \frac{\overline{x}}{\overline{x} + \overline{y}}\varepsilon_x + \frac{\overline{y}}{\overline{x} + \overline{y}}\varepsilon_y. \end{aligned} \tag{1}$$

We have a nice bound for the error, using the triangle inequality of real numbers $|a+b| \leq |a| + |b|$:

$$|e_{x+y}| = |e_x + e_y| \leq |e_x| + |e_y|.$$

Also, we have a similar result for the relative error if $\overline{x}$ and $\overline{y}$ are positive:

$$\begin{aligned} |\varepsilon_{x+y}| &= \left| \frac{\overline{x}}{\overline{x} + \overline{y}}\varepsilon_x + \frac{\overline{y}}{\overline{x} + \overline{y}}\varepsilon_y \right| \\ &\leq \left| \frac{\overline{x}}{\overline{x} + \overline{y}}\varepsilon_x \right| + \left| \frac{\overline{y}}{\overline{x} + \overline{y}}\varepsilon_y \right| \\ &= \frac{\overline{x}}{\overline{x} + \overline{y}} |\varepsilon_x| + \frac{\overline{y}}{\overline{x} + \overline{y}} |\varepsilon_y| \leq |\varepsilon_x| + |\varepsilon_y|. \end{aligned}$$

If $\overline{x}$ and $\overline{y}$ are positive, we have that $\frac{\overline{x}}{\overline{x}+\overline{y}}\left|\varepsilon_x\right| + \frac{\overline{y}}{\overline{x}+\overline{y}}\left|\varepsilon_y\right|$ is a weighted average of the two numbers, so in fact

$$\left|\varepsilon_{x+y}\right| \leq \max\left\{\left|\varepsilon_x\right|, \left|\varepsilon_y\right|\right\}.$$

Next, suppose that the two approximate numbers are multiplied together, and we declare that the approximation to xy is $\overline{xy} = \overline{x}\,\overline{y}$. Then

$$\begin{aligned} e_{xy} &= xy - \overline{x}\,\overline{y} \\ &= (\overline{x} + e_x)(\overline{y} + e_y) - \overline{x}\,\overline{y} \\ &= \overline{y}e_x + \overline{x}e_y + e_x e_y. \end{aligned}$$

Then

$$\begin{aligned} \varepsilon_{xy} &= \frac{e_{xy}}{\overline{x}\,\overline{y}} = \frac{e_x}{\overline{x}} + \frac{e_y}{\overline{y}} + \frac{e_x e_y}{\overline{x}\,\overline{y}} \\ &= \varepsilon_x + \varepsilon_y + \varepsilon_x\varepsilon_y, \end{aligned}$$

and thus

$$\left|\varepsilon_{xy}\right| \leq \left|\varepsilon_x\right| + \left|\varepsilon_y\right| + \left|\varepsilon_x\right|\left|\varepsilon_y\right|.$$

If ε_x and ε_y are very small, we have the approximation

$$\varepsilon_{xy} \approx \varepsilon_x + \varepsilon_y.$$

Similar computations can be done for other arithmetic operations and other functions of x and y.

1.4. **General formula for propagation of error.** In general, let $\overline{x}$ be an approximation to x, and let f be a C^2 function (a twice continuously differentiable function — ie a function such that its second derivative is continuous) that can be calculated accurately. We declare that $\overline{f(x)} = f(\overline{x})$ is the approximation to $f(x)$. By Taylor's formula with remainder,

$$f(x) = f(\overline{x}) + f'(\overline{x})(x - \overline{x}) + \frac{f''(c)}{2}(x - \overline{x})^2,$$

where c is a number between $\overline{x}$ and x. Then,

$$e_{f(x)} = f(x) - f(\overline{x}) = f'(\overline{x})\,e_x + \frac{f''(c)}{2}(e_x)^2.$$

From this formula, we have

$$\begin{aligned} \varepsilon_{f(x)} &= \frac{e_{f(x)}}{f(\overline{x})} = \frac{f'(\overline{x})}{f(\overline{x})}e_x + \frac{f''(c)}{2f(\overline{x})}(e_x)^2 \\ &= \frac{\overline{x}f'(\overline{x})}{f(\overline{x})}\frac{e_x}{\overline{x}} + \frac{\overline{x}^2 f''(c)}{2f(\overline{x})}\left(\frac{e_x}{\overline{x}}\right)^2 \\ &= \frac{\overline{x}f'(\overline{x})}{f(\overline{x})}\varepsilon_x + \frac{\overline{x}^2 f''(c)}{2f(\overline{x})}\varepsilon_x^2. \end{aligned}$$

Thus,

$$\begin{aligned} \left|e_{f(x)}\right| &\leq \left|f'(\overline{x})\right|\,\left|e_x\right| + \frac{\left|f''(c)\right|}{2}(\left|e_x\right|)^2 \\ &\leq M\left|e_x\right| + \frac{N}{2}(\left|e_x\right|)^2, \end{aligned}$$

where M and N are positive real number bounds on $|f'|$ and $|f''|$, respectively: $|f'(t)| \leq M$ and $|f''(t)| \leq N$ for t in the range from x to $\overline{x}$. Similarly,

$$\left|\varepsilon_{f(x)}\right| \leq A\left|\varepsilon_x\right| + B\left|\varepsilon_x\right|^2 \approx A\left|\varepsilon_x\right|,$$

where A and B are bounds:

$$\left|\frac{\overline{x}f'(\overline{x})}{f(\overline{x})}\right| \leq A, \ \left|\frac{\overline{x}^2 f''(c)}{2f(\overline{x})}\right| \leq B.$$

For example, we could use

$$A = \sup_x \left|\frac{xf'(x)}{f(x)}\right|, \ B = \sup_x \left|\frac{x^2}{2f(x)}\right| \sup_c \left|f''(c)\right|,$$

where the sup 's are taken over the range of the variables under consideration. The formulas above give general error formulas for the propagation of error through single variable functions.

The formula above for $\varepsilon_{f(x)}$ is a second order estimation of the propagation of error, but often a first order approximation is sufficient. Instead we may start with the zeroth order Taylor's formula with remainder (i.e. Mean Value Theorem):

$$f(x) = f(\overline{x}) + f'(c)(x - \overline{x}),$$

where c is a number between x and $\overline{x}$. Then

$$e_{f(x)} = f(x) - f(\overline{x}) = f'(c)e_x,$$

so that

$$\left|e_{f(x)}\right| \leq |f'(c)|\,|e_x|.$$

Thus, if $|f'(c)|$ can be estimated or bounded independent of c, the formula above can be used to bound the maximum error in the approximation $f(\overline{x})$ to $f(x)$.

We may then compute the first order approximation to $\varepsilon_{f(x)}$. From the formula above,

$$\varepsilon_{f(x)} = \frac{f'(c)\overline{x}}{f(\overline{x})}\varepsilon_x,$$

so that

$$\left|\varepsilon_{f(x)}\right| = \left|\frac{f'(c)\overline{x}}{f(\overline{x})}\right| \left|\varepsilon_x\right|.$$

If $|f'(c)|$ and $|\overline{x}|$ can be bounded from above and $|f(\overline{x})|$ can be bounded from below, the formula above could be used to bound $\left|\varepsilon_{f(x)}\right|$.

It is often the case that multivariable functions are needed. For example, multiplication is simply the application of the function $g(x, y) = xy$. In general we let g be a real-valued C^2 function of n variables, and let $\overline{x} = (\overline{x_1}, ..., \overline{x_n})$ be approximations for $x = (x_1, ..., x_n)$. Note that the function being C^2 means that all second partial derivatives are continuous. We declare $g(\overline{x_1}, ..., \overline{x_n})$ to be an approximation to $g(x_1, ..., x_n)$, and we wish to compute the error $e_{g(x_1,...,x_n)}$. In order to use the Taylor remainder formula, we let $F(t)$ be the function

$$F(t) = g\left(\overline{x_1} + t\left(x_1 - \overline{x_1}\right), \overline{x_2} + t\left(x_2 - \overline{x_2}\right), ..., \overline{x_n} + t\left(x_n - \overline{x_n}\right)\right),$$

which is C^2 (since g is). Observe that $F(0) = g(\overline{x_1}, ..., \overline{x_n})$ and $F(1) = g(x_1, ..., x_n)$, so we may use the Taylor remainder formula to see that

$$\begin{aligned} F(t) &= F(0) + \frac{dF}{dt}(c)\, t; \\ F(1) &= F(0) + \frac{dF}{dt}(c), \end{aligned}$$

where c is a number between 0 and 1. Then

$$\begin{aligned} e_{g(x_1,...,x_n)} &= g(x_1, ..., x_n) - g(\overline{x_1}, ..., \overline{x_n}) = F(1) - F(0) \\ &= \frac{dF}{dt}(c) \\ &= \left(\frac{\partial g}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial g}{\partial x_2}\frac{\partial x_2}{\partial t} + ... + \frac{\partial g}{\partial x_n}\frac{\partial x_n}{\partial t} \right)(c) \end{aligned}$$

using the chain rule. Continuing to use the product and chain rule (and the fact that $\frac{\partial x_j}{\partial t} = (x_j - \overline{x_j})$):

$$\begin{aligned} e_{g(x_1,...,x_n)} &= \sum_{j=1}^{n} \frac{\partial g(\overline{c})}{\partial x_j}(x_j - \overline{x_j}) \\ &= \sum_{j=1}^{n} \frac{\partial g(\overline{c})}{\partial x_j} e_{x_j} \\ &= \nabla g(\overline{c}) \cdot (e_{x_1}, ..., e_{x_n}), \end{aligned}$$

where $\overline{c} = (\overline{x_1} + c(x_1 - \overline{x_1}), ..., \overline{x_n} + c(x_n - \overline{x_n}))$ is a specific point on the straight line connecting $\overline{x}$ to x, and ∇g is the gradient of g. This is the general error formula, and we get the corresponding bound on the relative error to be:

$$\begin{aligned} \varepsilon_{g(x_1,...,x_n)} &= \frac{e_{g(x_1,...,x_n)}}{g(\overline{x_1}, ..., \overline{x_n})} \\ &= \sum_{j=1}^{n} \frac{\overline{x_j}}{g(\overline{x})}\frac{\partial g(\overline{c})}{\partial x_j}\frac{e_{x_j}}{\overline{x_j}} = \sum_{j=1}^{n} \frac{\overline{x_j}}{g(\overline{x})}\frac{\partial g(\overline{c})}{\partial x_j}\varepsilon_{x_j}. \end{aligned}$$

Using bounds on the function g and its derivatives and the approximate values $\overline{x} = (\overline{x_1}, ..., \overline{x_n})$, as well as the Cauchy-Schwarz inequality $|v \cdot w| \leq |v|\,|w|$, we may get general error bounds:

$$\left| e_{g(x_1,...,x_n)} \right| \leq |\nabla g(\overline{c})|\, |e_x|,$$

where $|e_x| = \sqrt{\sum \left| e_{x_j} \right|^2}$and $|\nabla g(\overline{c})|$ is the length of the gradient vector of g at $\overline{c}$.

Similarly, we may bound the relative error as follows:

$$\begin{aligned} \left| \varepsilon_{g(x_1,...,x_n)} \right| &= \frac{\left| e_{g(x_1,...,x_n)} \right|}{|g(\overline{x})|} \leq |\nabla g(\overline{c})| \frac{|e_x|}{|g(\overline{x})|} \\ &= \frac{|\nabla g(\overline{c})|\ |\overline{x}|}{|g(\overline{x})|}\ |\varepsilon_x|, \end{aligned}$$

where again $|\varepsilon_x| = \sqrt{\sum \left| \varepsilon_{x_j} \right|^2}$.

2. Interpolation, extrapolation, finite differences

2.1. **Polynomial interpolation.** We wish to find a polynomial function that fits the data points $(x_0, y_0), ..., (x_n, y_n)$, with $x_0 < x_1 < ... < x_n$. It turns out that we can always find a polynomial

$$f(x) = a_0 + a_1 x + ... + a_n x^n,$$

which is of degree n or less, that exactly fits the given data.

Theorem 3. ***(Lagrange interpolation)*** *The polynomial $P_n(x)$ of degree at most n given by the formula below satisfies $P_n(x_j) = y_j$ for $0 \leq j \leq n$, and it is the unique polynomial of degree n or less that satisfies all of those equations.*

$$P_n(x) = \sum_{j=0}^{n} y_j \prod_{k \neq j} \frac{(x - x_k)}{(x_j - x_k)}.$$

2.2. **Finite Differences.** We will now discuss another method of finding the interpolating polynomial in the particular cases where the x-values of the data points are evenly spaced. This means the data points satisfy $x_{j+1} - x_j = h$ (a fixed number) for each j, we may make a finite difference table of data. In this table we will use the notation

$$\begin{aligned} \Delta y_k &= \Delta f(x_k) = f(x_k + h) - f(x_k) \\ &= f(x_{k+1}) - f(x_k) \\ &= y_{k+1} - y_k. \end{aligned}$$

Then, for example.

$$\begin{aligned} \Delta^2 f(x_k) &= \Delta f(x_{k+1}) - \Delta f(x_k) \\ &= f(x_{k+2}) - 2f(x_{k+1}) + f(x_k). \end{aligned}$$

The table of data looks like

x_j	y_j	Δy_j	$\Delta^2 y_j$	$\Delta^3 y_j$	...
x_0	y_0	$y_1 - y_0$	$y_2 - 2y_1 + y_0$	$y_3 - 3y_2 + 3y_1 - y_0$	...
x_1	y_1	$y_2 - y_1$	$y_3 - 2y_2 + y_1$	$y_4 - 3y_3 + 3y_2 - y_1$	...
x_2	y_2	$y_3 - y_2$	$y_4 - 2y_3 + y_2$	$y_5 - 3y_4 + 3y_3 - y_2$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Again, we may use the data in the first line of the chart to determine the interpolating polynomial.

Theorem 4. ***(Newton Finite Difference Formula)*** *The unique polynomial of degree n or less fitting the first $n+1$ points in the finite difference table above with $x_{j+1} - x_j = h$ for each j satisfies*

$$\begin{aligned} P_n(x) &= y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!h^2}(x - x_0)(x - x_1) \\ &\quad + ... + \frac{\Delta^n y_0}{n!h^n}(x - x_0)...(x - x_{n-1}). \end{aligned}$$

We wish to determine the error in polynomial interpolation (given either by the Lagrange or Newton formulas). We are interested in an unknown function $f(x)$ with known values at

$x_0, ..., x_n$ with $x_0 < x_1 < ... < x_n$ that is approximated by $P_n(x)$, which is the interpolating polynomial constructed from the given data. We are interested in the error (or remainder)

$$R(x) = f(x) - P_n(x)$$

and whether or not we can control it. Here is one such result.

Theorem 5. ***(Error in Polynomial Interpolation)*** *We use the notation as above. Suppose that $f(x)$ is C^{n+1} on the interval (x_0, x_n) and continuous on $[x_0, x_n]$. Then for each $x \in (x_0, x_n)$ there exists $\xi \in (x_0, x_n)$ such that*

$$R(x) = \frac{f^{n+1}(\xi)}{(n+1)!}(x - x_0)(x - x_1)...(x - x_n).$$

Proof. Note that if $x = x_j$ for some j, then there is nothing to prove. Otherwise, fix x, and let $F(x) = (x - x_0)(x - x_1)...(x - x_n)$, and define the one-variable function $\phi(s)$ by

$$\phi(s) = f(s) - P_n(s) - \frac{f(x) - P_n(x)}{F(x)} F(s).$$

Observe that $\phi(x_j) = 0$ for every j, since $f(x_j) = P_n(x_j)$ and $F(x_j) = 0$. Also, $\phi(x) = 0$. Thus, $\phi(s)$ has at least $n+2$ zeros in the interval (x_0, x_n). By Rolle's Theorem from Calculus, between each pair of zeros must be a point where $\phi'(s) = 0$. Thus, ϕ' has at least $n+1$ zeros in the interval. As we keep going, we must have that $\phi^{(n+1)}(s)$ has at least one zero in the interval; call one of those zeros ξ. Thus, since the $(n+1)^{st}$ derivative of $F(s)$ is $(n+1)!$,

$$\begin{aligned} 0 &= \phi^{(n+1)}(\xi) \\ &= f^{(n+1)}(\xi) - 0 - \frac{f(x) - P_n(x)}{F(x)}(n+1)! \end{aligned}$$

Then

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

□

One result of this theorem is that we see that unless we know a lot of information about the unknown function f, we don't know how to bound the error $R(x)$. In order to have a good bound, we would need to know an upper bound on the quantity $\left|f^{(n+1)}(\xi)\right|$ for $\xi \in (x_0, x_n)$. The famous Runge Example (see the problems from this section) shows that if $f(x) = \frac{1}{1+25x^2}$ on the interval $[-1, 1]$, for x near ± 1, the error in the polynomial approximation goes to infinity as the number of equal subintervals n increases. We now see why this happens, because the $(n+1)^{st}$ derivatives of this function get very large around $x = \pm 1$, as $n \to \infty$.

2.3. **Cubic Splines.** Another technique of approximating an unknown function from a set of data points is by using **piecewise polynomial approximation**; this means that we use subsets of points to interpolate with polynomials and connect these together to form one function. This technique is used heavily in numerical integration (quadrature). One example of this is to use quadratic polynomials only. We find the quadratic polynomial fitting the data through x_0, x_1, x_2, and then we find the quadratic polynomial fitting the data through

x_2, x_3, x_4, and so on. Our approximate function is then given by the piecewise degree two polynomial through all of the data points.

Another popular technique is to use **splines**. Splines are piecewise-defined functions (usually polynomials) that are determined between each consecutive pair of data points, designed to piece together to a certain degree of smoothness in the spline function $g(x)$. For example, if the first derivatives of the piecewise spline functions match at a data point, then there will not be a corner at that data point. In general, we may require that several orders of derivatives must match at each data point. Furthermore, a spline is designed so that the derivatives of the spline function do not grow too large (as in the Runge example) on average. This property of splines is controlled by requiring that the quantity

$$\int_{x_0}^{x_n} \left(g^{(k)}(x)\right)^2 dx$$

is minimized among all possible curves. The positive integer k depends on the amount of smoothness required and the degrees of the polynomials used between data points. The data points $x_0 < x_1 < ... < x_n$ are called **nodes** or **knots** for historical reasons; typically a draftsman or engineer uses flexible material between fixed locations, and the flexible material assumes the shape where curvature is minimized. In a sense, the integral above measures some amount of total curvature.

The **natural cubic spline** is the simplest and most commonly used example of a spline. In this case cubic polynomials are used, and it is required that the first and second derivatives of the polynomials match at the data points. In addition, we impose the conditions $g''(x_0) = g''(x_n) = 0$. It can be shown that such a spline is uniquely determined by the data and conditions, and the quantity

$$\int_{x_0}^{x_n} \left(g''(x)\right)^2 dx$$

is the smallest possible among all curves connecting the data points. In the case where the first derivative is not too large, the integral above is close to the total curvature as measured in differential geometry.

To see that it is reasonable that the cubic spline is uniquely determined, consider the numbers of equations and unknowns. With $n+1$ data points, we will have n cubic functions to determine, which have 4 parameters each ($a_0 + a_1x + a_2x^2 + a_3x^3$), to give a total of $4n$ constants to determine. Since each cubic curve must match two data points (one on each side), that gives $2n$ linear constraints. Since the first and second derivatives must match in the interior points ($n-1$ of those), there are $2(n-1)$ more linear constraints. Further, we require g'' is zero at the two endpoints of the whole interval $[x_0, x_n]$, and so that gives two additional constraints. This means that there are a total of

$$2n + 2(n-1) + 2 = 4n$$

equations and $4n$ unknown functions, which can be solved using a matrix equation. By examining carefully the kind of matrix that is obtained, one may show that the equation is of the form $Mx = b$, with M invertible, and thus the cubic spline is uniquely determined.

Example: Find the cubic spline connecting the points $(0,0), (1,3), (4,7)$.

Solution: Let $a_0 + a_1x + a_2x^2 + a_3x^3$ be the cubic polynomial between $x = 0$ and $x = 1$, and let $b_0 + b_1x + b_2x^2 + b_3x^3$ be the cubic polynomial between $x = 1$ and $x = 4$. The

equations we get are:

$$\begin{aligned} a_0 &= 0 \\ a_0+a_1+a_2+a_3 &= 3 \\ b_0+b_1+b_2+b_3 &= 3 \\ b_0+4b_1+16b_2+64b_3 &= 7 \\ a_1+2a_2+3a_3 &= b_1+2b_2+3b_3 \\ 2a_2+6a_3 &= 2b_2+6b_3 \\ 2a_2 &= 0 \\ 2b_2+24b_3 &= 0 \end{aligned}$$

The first four equations come from matching the functions with the data points, the next one is the requirement that the first derivatives match at $x=1$, the next is the requirement that the second derivatives match at $x=1$, and the last two equations come from the requirement that the second derivatives are zero at $x=0$ and $x=4$. Putting these equations into a matrix equation, we get

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 16 & 64 \\ 0 & 1 & 2 & 3 & 0 & -1 & -2 & -3 \\ 0 & 0 & 2 & 6 & 0 & 0 & -2 & -6 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 24 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \\ 3 \\ 7 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

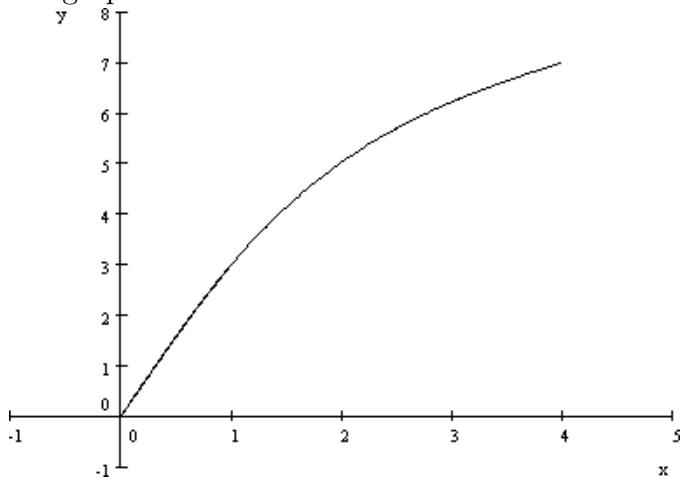
So that

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 16 & 64 \\ 0 & 1 & 2 & 3 & 0 & -1 & -2 & -3 \\ 0 & 0 & 2 & 6 & 0 & 0 & -2 & -6 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 24 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 3 \\ 3 \\ 7 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{77}{24} \\ 0 \\ -\frac{5}{24} \\ -\frac{5}{18} \\ \frac{97}{24} \\ -\frac{5}{6} \\ \frac{5}{72} \end{pmatrix}$$

Thus, the natural cubic spline $g(x)$ fitting the data $(0,0),(1,3),(4,7)$ is

$$g(x) = \begin{cases} \frac{77}{24}x - \frac{5}{24}x^3 & 0 \le x \le 1 \\ -\frac{5}{18} + \frac{97}{24}x - \frac{5}{6}x^2 + \frac{5}{72}x^3 & 1 \le x \le 4 \end{cases}$$

We graph to see:



3. Numerical Integration

In this section, we will learn to approximate definite integrals $\int_a^b f(x)\ dx$; such approximation using numerical methods is called **quadrature**. We will split up the interval $[a, b]$ into subintervals, will approximate the function $f(x)$ by a simple function (such as a polynomial) on each subinterval, and will then evaluate the definite integral of each approximate function. We will need to keep track of the number of function evaluations necessary (related to how fast a computer could calculate such things) and to have some control over the error in these approximations.

3.1. **Rectangle and trapezoidal approximations.** The simplest approximations come from the **Riemann sum approximation** used in the definition of the Riemann integral. If f is a continuous function, then

$$\begin{aligned}\int_a^b f(x)\ dx &= \lim_{n\to\infty} LEFT(n) \\ &= \lim_{n\to\infty} \frac{b-a}{n} \sum_{j=0}^{n-1} f\left(a + j\frac{b-a}{n}\right) \\ &= \lim_{n\to\infty} h\left(f(x_0) + f(x_1) + ... + f(x_{n-1})\right),\end{aligned}$$

where here the subinterval size is $h = \frac{b-a}{n}$, and the points of evaluation are $x_j = a + jh$. The sum shown above is the **left Riemann sum approximation with equal subdivisions**. Similarly,

$$\begin{aligned}RIGHT(n) &= \frac{b-a}{n} \sum_{j=1}^{n} f\left(a + j\frac{b-a}{n}\right) \\ &= h\left(f(x_1) + ... + f(x_n)\right)\end{aligned}$$

is the **right Riemann sum approximation with equal subdivisions**. To determine the error in the $LEFT(n)$ approximation, observe that by the **mean value theorem**,

$\frac{f(x)-f(x_j)}{x-x_j} = f'(c)$ for some $c \in (x_j, x)$, and so

$$\begin{aligned}\int_{x_j}^{x_{j+1}} f(x)\ dx - hf(x_j) &= \int_{x_j}^{x_{j+1}} (f(x_j) + (x-x_j) f'(c))\ dx - hf(x_j) \\ &= h\ f(x_j) + \int_{x_j}^{x_{j+1}} (x-x_j) f'(c)\ dx - hf(x_j) \\ &= \int_{x_j}^{x_{j+1}} (x-x_j) f'(c)\ dx\end{aligned}$$

(note that c depends on x). We now use a standard trick in this subject, the **mean value theorem for integrals:**

Theorem 6. ***(Mean Value Theorem for Integrals - MVTI)*** *Given a function $G(t)$ that is continuous on $[a,b]$ and an integrable function $\phi(t)$ that does not change sign on (a,b), then there exists a $\xi \in (a,b)$ such that*

$$\int_a^b G(t)\,\phi(t)\ dt = G(\xi)\ \int_a^b \phi(t)\ dt.$$

Also important is the **intermediate value theorem**:

Theorem 7. ***(Intermediate Value Theorem - IVT)*** *For any continuous function $F : [s,t] \to \mathbb{R}$, if y is any number between $F(s)$ and $F(t)$, then there exists an $x \in (s,t)$ such that $F(x) = y$.*

Thus, with MVTI in mind, and with $(x - x_j)$ being the function that does not change sign in (x_j, x_{j+1}), if f' is continuous, then for some $\xi \in (x_j, x_{j+1})$

$$\int_{x_j}^{x_{j+1}} (x-x_j) f'(c)\ dx = f'(\xi) \int_{x_j}^{x_{j+1}} (x-x_j)\ dx = f'(\xi)\,\frac{(x_{j+1}-x_j)^2}{2} = \frac{h^2}{2} f'(\xi).$$

Here we have used the slightly tricky fact that if f' is continuous, then the c in the mean value theorem above can be chosen to be continuous in x. Now, adding up the contributions from all the intervals, we have

$$\int_a^b f(x)\ dx - LEFT(n) = \frac{h^2}{2}\left(f'(\xi_1) + ... + f'(\xi_n)\right),$$

where each $\xi_j \in (x_{j-1}, x_j)$. Next, since the average of the $f'(\xi_j)$ must be a number between the smallest and largest of the $f'(\xi_k)$s, we have by the intermediate value theorem that

$$\begin{aligned}\int_a^b f(x)\ dx - LEFT(n) &= \frac{h^2}{2} n\,(\text{average of the } f'(\xi_j)) \\ &= \frac{h^2}{2} n\ f'(\tau)\end{aligned}$$

for some $\tau \in (a,b)$. Thus, since $h = \frac{b-a}{n}$, the error in $LEFT(n)$ satisfies

$$\int_a^b f(x)\ dx - LEFT(n) = \frac{(b-a)^2}{2n} f'(\tau)$$

for some $\tau \in (a,b)$. Similarly, we have

$$\int_a^b f(x)\ dx - RIGHT(n) = -\frac{(b-a)^2}{2n} f'(\alpha)$$

for some $\alpha \in (a,b)$. Thus, these approximations are good when the first derivative of the function is small in absolute value (and when n is large). Observe that each of these approximations uses n function evaluations.

The **trapezoid rule** may be defined simply by the formula

$$TRAP(n) = \frac{1}{2}(LEFT(n) + RIGHT(n))$$

as the average of the left and right Riemann sum approximations. However, geometrically there is another way to think about it. The approximation to the function $f(x)$ in the subinterval is the linear interpolation — that is, the function obtained by using a straight line connecting $(x_{j-1}, f(x_{j-1}))$ and $(x_j, f(x_j))$. Note then that theapproximation to the integral is

$$\frac{h}{2}(f(x_{j-1}) + f(x_j)),$$

the area of the trapezoid. Therefore,

$$\begin{aligned} TRAP(n) &= \frac{h}{2}\sum_{j=1}^{n}(f(x_{j-1}) + f(x_j)) \\ &= \frac{h}{2}(y_0 + 2y_1 + 2y_2 + ... + 2y_{n-1} + y_n), \end{aligned}$$

where $y_j = f(x_j)$ for each j. Note that the trapezoid rule uses $n+1$ function evaluations for n subdivisions of $[a,b]$.

We now need to determine the error in the trapezoid approximation. First, let's consider the interval from $a = x_0$ to $a + h = x_1$. We use the finite difference formula:

$$f(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{1}{2} f''(c)(x - x_0)(x - x_1),$$

where c depends on x and is a number between x_0 and x_1. We can assume c is chosen continuously if necessary, by assuming f is C^2 (second derivatives continuous). If we substitute $u = \frac{x - x_0}{h}$, we get

$$\begin{aligned} \int_{x_0}^{x_1} f(x)\ dx &= \int_{x_0}^{x_1} \left(y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{1}{2} f''(c)(x - x_0)(x - x_1) \right) dx \\ &= h\int_0^1 \left(y_0 + \Delta y_0 u + \frac{h^2}{2} f''(c)\, u(u-1) \right) du \\ &= y_0 h + \Delta y_0 \frac{h}{2} + \frac{h^3}{2}\int_0^1 f''(c)\, u(u-1)\ du. \end{aligned}$$

By the mean value theorem for integrals (since $u(u-1)$ is negative on the interval of integration) and $\int_0^1 u(u-1)\ du = -\frac{1}{6}$, we have

$$\begin{aligned}\int_{x_0}^{x_1} f(x)\ dx &= y_0 h + \Delta y_0 \frac{h}{2} + \frac{h^3}{2} f''(\xi_1) \int_0^1 u(u-1)\ du \\ &= y_0 h + \Delta y_0 \frac{h}{2} + \frac{-h^3}{12} f''(\xi_1) \\ &= h\left(\frac{y_0}{2} + \frac{y_1}{2} - \frac{h^2}{12} f''(\xi_1)\right)\end{aligned}$$

for some $\xi_1 \in (x_0, x_1)$. Then the error in the interval $[x_0, x_1]$ is $-\frac{h^3}{12} f''(\xi_1)$. Summing over all n of the subintervals, we get

$$\int_a^b f(x)\ dx - TRAP(n) = -\frac{h^3}{12}\left(f''(\xi_1) + f''(\xi_2) + ... + f''(\xi_n)\right),$$

where each ξ_j is an element of (x_{j-1}, x_j). Since the average $\frac{1}{n}\left(f''(\xi_1) + f''(\xi_2) + ... + f''(\xi_n)\right)$ is between the largest and smallest $f''(\xi_j)$ values, the intermediate value theorem tells us that there is a $\tau \in (a,b)$ such that $f''(\tau) = \frac{1}{n}\left(f''(\xi_1) + f''(\xi_2) + ... + f''(\xi_n)\right)$. Then we have that the **error in** $TRAP(n)$ is

$$\begin{aligned}\int_a^b f(x)\ dx - TRAP(n) &= -\frac{h^3 n}{12} f''(\tau) \\ &= -\frac{(b-a)^3}{12n^2} f''(\tau),\end{aligned}$$

for some $\tau \in (a,b)$. Note that as $n \to \infty$, $Trap(n)$ rapidly approaches $\int_a^b f(x)\ dx$, and the approximation is best when $|f''(x)|$ is small. Note that this approximation is valid only when f is C^2 (i.e. has continuous second derivatives).

3.2. **Higher degree approximations.** We now derive the m^{th} degree polynomial interpolation approximation to $\int_a^b f(x)\ dx$. We start with the data points $(x_0, y_0), (x_1, y_1), ..., (x_m, y_m)$ and find the unique polynomial curve through those points, which are spaced equally with $x_j - x_{j-1} = h$ for all j. We then integrate that polynomial expression to yield an approximation to $\int_{x_0}^{x_m} f(x)\ dx$. We then sum up a series of such approximations to get the m^{th} degree polynomial approximation. Here goes!

$$\begin{aligned}\int_{x_0}^{x_m} f(x)\ dx &\approx \int_{x_0}^{x_m} \left(y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!h^2}(x-x_0)(x-x_1) + ... \right. \\ &\quad \left. + \frac{\Delta^m y_0}{m!h^m}(x-x_0)(x-x_1)...(x-x_{m-1})\right) dx \\ &= h\int_0^m \left(y_0 + (\Delta y_0)\, u + \frac{\Delta^2 y_0}{2!} u(u-1) + ... \right. \\ &\quad \left. + \frac{\Delta^m y_0}{m!} u(u-1)...(u-m+1)\right) du,\end{aligned}$$

using the substitution $u = \frac{x-x_0}{h}$. Then the polynomial approximation to $\int_{x_0}^{x_m} f(x)\ dx$ is

$$hy_0 m + h(\Delta y_0)\frac{m^2}{2} + h\frac{\Delta^2 y_0}{2}\left(\frac{m^3}{3} - \frac{m^2}{2}\right) + h\frac{\Delta^3 y_0}{6}\left(\frac{m^4}{4} - m^3 - m^2\right)... \\ + h\frac{\Delta^m y_0}{m!}\int_0^1 u(u-1)...(u-m+1)\,du$$

For examples, we choose a particular m, calculate the approximation above for each set of m intervals, and then add up the contributions.

$$\begin{aligned} m &= 1 \text{ (Trapezoid Rule):} \\ TRAP &= hy_0 + h(y_1 - y_0)\frac{1}{2} = \frac{h}{2}(y_0 + y_1) \\ \int_a^b f(x)\ dx &\approx TRAP(n) = \frac{h}{2}(y_0 + 2y_1 + 2y_2 + ... + 2y_{n-1} + y_n). \end{aligned}$$

$$\begin{aligned} m &= 2 \text{ (Simpson's Rule)} \\ SIMP &= hy_0(2) + h(y_1 - y_0)\left(\frac{2^2}{2}\right) + h\frac{(y_2 - 2y_1 + y_0)}{2}\left(\frac{2^3}{3} - \frac{2^2}{2}\right) \\ &= h\left(2y_0 + 2y_1 - 2y_0 + (y_2 - 2y_1 + y_0)\left(\frac{4}{3} - 1\right)\right) \\ &= \frac{h}{3}(y_0 + 4y_1 + y_2) \\ \int_a^b f(x)\ dx &\approx SIMP(n) = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + ... + 4y_{n-1} + y_n). \end{aligned}$$

Note that n must be even in order to apply Simpson's rule.

$$\begin{aligned} m &= 3 \text{ (Simpson's } \frac{3}{8} \text{ Rule)} \\ SIMP3/8 &= hy_0(3) + h(y_1 - y_0)\left(\frac{3^2}{2}\right) + h\frac{(y_2 - 2y_1 + y_0)}{2}\left(\frac{3^3}{3} - \frac{3^2}{2}\right) \\ &\quad + h\frac{(y_3 - 3y_2 + 3y_1 - y_0)}{6}\left(\frac{3^4}{4} - 3^3 + 3^2\right) \\ &= ... \\ &= \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3) \\ \int_a^b f(x)\ dx &\approx \\ SIMP3/8(n) &= \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + 2y_3 + 3y_4 + 3y_5 + 2y_6 + ... + 3y_{n-1} + y_n). \end{aligned}$$

In the case of Simpson's $\frac{3}{8}$ Rule, n must be a multiple of three.

To illustrate, we derive the error in the Simpson's Rule approximation.

Recall the formula for the Newton finite difference formula with error term:

$$\begin{aligned}\int_{x_0}^{x_m} f(x)\ dx &= \int_{x_0}^{x_m} \left(y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2!h^2}(x-x_0)(x-x_1) + ... \right. \\ &\quad \left. + \frac{\Delta^m y_0}{m!h^m}(x-x_0)(x-x_1)...(x-x_{m-1}) \right) dx \\ &\quad + \int_{x_0}^{x_m} \frac{f^{(m+1)}(\xi)}{(m+1)!}(x-x_0)(x-x_1)...(x-x_m)\,dx\end{aligned}$$

Therefore, the last term gives the formula for the error in the m^{th} order polynomial approximation. Note that when m is even and positive,

$$\int_{x_0}^{x_m} \frac{\Delta^{m+1} y_0}{(m+1)!h^{m+1}}(x-x_0)(x-x_1)...(x-x_m)\ dx$$

$$\frac{\Delta^{m+1} y_0}{(m+1)!h^{m+1}} \int_{x_0}^{x_m} (x-x_0) \qquad (x-x_1)...(x-x_m)\ dx$$

$$= \frac{\Delta^{m+1} y_0}{(m+1)!} \int_0^m (u)(u-1)...(u-m)\ du = 0$$

using the substitution $u = \frac{x-x_0}{h}$. So to get the error when m is even, we may construct an interpolating polynomial of one higher degree that has the same integral. For example, the $m = 2$ case is Simpson's Rule, and we let

$$\begin{aligned}p_2(x) &= y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2!h^2}(x-x_0)(x-x_1), \\ \widetilde{p_2}(x) &= p_2(x) + \frac{1}{h^2}\left(p_2'(x_1) - f'(x_1)\right)(x-x_0)(x-x_1)(x-x_2).\end{aligned}$$

Observe that

$$\int_{x_0}^{x_2} p_2(x)\,dx = \int_{x_0}^{x_2} \widetilde{p_2}(x)\,dx,$$

and observe that $(f-\widetilde{p_2})(x) = 0$ when $x = x_0, x_1, x_2$ and also $(f-\widetilde{p_2})'(x_1) = 0$, so that x_1 is a double root. Let x be a fixed element of (x_0, x_2) that is not x_1. Now, we define the auxillary function

$$g(t) = f(t) - \widetilde{p_2}(t) - \frac{f(x) - \widetilde{p_2}(x)}{(x-x_0)(x-x_1)^2(x-x_2)}(t-x_0)(t-x_1)^2(t-x_2).$$

Then, observe that $g(t) = 0$ at $t = x_0, x, x_2$, and g has a double root at $t = x_1$. Then, by Rolle's Theorem, there exists t_1, t_2, t_3 in (x_0, x_2) (but not equal to x_0, x_1, x_2, x) such that $g'(t_j) = 0$, and also $g'(x_1) = 0$. Since there are four points where g' is zero, by Rolle's Theorem there are three points where g'' is zero, and thus two points where g''' is zero, and hence one $\xi \in (x_0, x_2)$ where $g^{(4)}(\xi) = 0$. Taking the 4^{th} derivative with respect to t of the equation above and evaluating at $t = \xi$, we get

$$0 = f^{(4)}(\xi) - \frac{f(x) - \widetilde{p_2}(x)}{(x-x_0)(x-x_1)^2(x-x_2)}(24).$$

Thus,

$$f(x) - \widetilde{p_2}(x) = \frac{1}{24} f^{(4)}(\xi)(x-x_0)(x-x_1)^2(x-x_2).$$

As we have seen in other examples, ξ depends on the choice of $x \in [x_0, x_n]$, and $f^{(4)}(\xi)$ is actually continuous in x. Then the error in the Simpson's Rule approximation for $\int_{x_0}^{x_2} f(x)\ dx$ is

$$\begin{aligned}
\int_{x_0}^{x_2} f(x)\ dx - SIMP &= \int_{x_0}^{x_2} (f(x) - p_2(x))\ dx \\
&= \int_{x_0}^{x_2} (f(x) - \widetilde{p_2}(x))\ dx \\
&= \int_{x_0}^{x_2} \frac{1}{24} f^{(4)}(\xi)(x - x_0)(x - x_1)^2 (x - x_2)\, dx \\
&= \frac{1}{24} f^{(4)}(\kappa) \int_{x_0}^{x_2} (x - x_0)(x - x_1)^2 (x - x_2)\, dx
\end{aligned}$$

for some $\kappa \in (x_0, x_2)$, by the mean value theorem for integrals. Making the substitution $u = \frac{x - x_0}{h}$, we have

$$\begin{aligned}
\int_{x_0}^{x_2} f(x)\ dx - SIMP &= \frac{h^5}{24} f^{(4)}(\kappa) \int_0^2 u(u-1)^2 (u-2)\, du \\
&= \frac{h^5}{24} f^{(4)}(\kappa) \left(-\frac{4}{15}\right) \\
&= -\frac{h^5}{90} f^{(4)}(\kappa).
\end{aligned}$$

Next, we add this result among all subintervals from $a = x_0$ to $b = x_n$ to get

$$\int_a^b f(x)\ dx - SIMP(n) = \sum_{j=1}^{n/2} -\frac{h^5}{90} f^{(4)}(\kappa_j)$$

(one κ_j for every two subintervals)

$$= -\frac{h^5}{90} \frac{n}{2} f^{(4)}(\tau)$$

where $\tau \in (a, b)$ is chosen by the intermediate value theorem so that $f^{(4)}(\tau)$ is the average of the $f^{(4)}(\kappa_j)$'s. Thus,

$$\begin{aligned}
\int_a^b f(x)\ dx - SIMP(n) &= -\frac{(b-a)^5}{90n^5} \frac{n}{2} f^{(4)}(\tau) \\
&= -\frac{(b-a)^5}{180n^4} f^{(4)}(\tau)
\end{aligned}$$

Notice that Simpson's Rule is a good approximation when the fourth derivative can be bounded, and the approximation rapidly becomes better as the number of subdivisions increases.

3.3. **Adaptive Quadrature.** Computer programs typically use adaptive quadrature to evaluate definite integrals. The idea is that equal length subintervals do not necessarily provide the most efficient means of approximating a definite integral within a desired level of accuracy. This process utilizes a quadrature technique on each subinterval and an estimate of the error. If the error is not sufficiently low, the subinterval is divided again.

We illustrate the technique with an adaptive trapezoid method to compute $\int_a^b g(x)\ dx$. First, suppose that the trapezoid rule is used to get the estimate $TRAP(1)$ for the integral over the subinterval $[x_0, x_1]$ with one subdivision. According to the remainder estimate, the error $E(1)$ in this approximation is

$$E(1) = \int_{x_0}^{x_1} g(x)\ dx - TRAP(1) = -\frac{(x_1 - x_0)^3}{12} f''(\tau_1)$$

for some $\tau_1 \in (x_0, x_1)$. If this interval is divided into two, the error estimate becomes

$$E(2) = \int_{x_0}^{x_1} g(x)\ dx - TRAP(2) = -\frac{(x_1 - x_0)^3}{12(4)} f''(\tau_2)$$

for some $\tau_2 \in (x_0, x_1)$. Making the assumption that $f''(\tau_1) \approx f''(\tau_2)$, we have $E(1) \approx 4E(2)$. Subtracting $E(1) - E(2)$, we see that

$$\begin{aligned} TRAP(2) - TRAP(1) &\approx 3E(2), \text{ or} \\ E(2) &\approx \frac{TRAP(2) - TRAP(1)}{3}. \end{aligned}$$

We can use this formula to estimate the accuracy of the trapezoid estimate in each subinterval.

We could now estimate the value of the integral $\int_a^b g(x)\ dx$, accurate approximately to within a total error E, in the following **adaptive trapezoid rule algorithm**.

(1) Let $E_0 = E$, $SUM = 0$. Let the "unfinished intervals/errors list" be the empty set. Set $[a, b]$ to be the interval under consideration.
(2) Subdivide the interval into two equal subdivisions. Calculate $TRAP(2)$ and $TRAP(1)$ for the interval and the subdivided interval.
(3) Calculate $E(2) = \frac{1}{3}(TRAP(2) - TRAP(1))$.
(4) If $|E(2)| < E_0$, then
   (a) Let $SUM = SUM + TRAP(2) + E(2)$.
   (b) If there are any other intervals left in the "unfinished intervals/errors list", go back to Step (2) for the next interval on that list (setting E_0 to the paired error bound), and also delete that interval from the "unfinished intervals/errors" list.
   (c) Otherwise, go to Step (5).

   Otherwise, if $|E(2)| \geq E_0$, subdivide the interval into two equal subdivisions, and
   (a) Let $E_0 = \frac{E_0}{2}$.
   (b) Add the second subinterval to the "unfinished intervals/errors list", paired with E_0 as the error bound with that subinterval.
   (c) Go back to Step (2), using the first subinterval as the new interval under consideration.
(5) Stop; SUM is the approximation to the integral.

Note that in this algorithm, we do the additional step of adding the predicted error to the estimate so that the estimate is even more accurate. This means that in effect we are gaining an extra level of accuracy, comparable to Simpson's rule on the original interval. This idea is called **Romberg integration**.

The algorithm above is good, however there is a serious drawback. This approximation gives the given integral within the desired error bound, but not a relative error bound. Thus if $\int_a^b g(x)\ dx$ is large, the approximation will be very good, but if $\int_a^b g(x)\ dx$ is small, then

the approximation will not be accurate to very many significant digits. We will need to change this procedure so that it applies to relative errors.

First, recall from (1), if I_1 and I_2 are two approximate values of the integral over two subintervals, then the relative error in the sum is

$$\varepsilon_{I_1+I_2} = \frac{I_1}{I_1+I_2}\varepsilon_{I_1} + \frac{I_2}{I_1+I_2}\varepsilon_{I_2} \ .$$

Using induction and a similar technique to the derivation of (1), the relative error in the formula $S = \sum_{j=1}^{k} I_k$ is

$$\varepsilon_S = \sum_{j=1}^{k} \frac{I_j}{S}\varepsilon_{I_j}.$$

We see that if $\varepsilon > 0$ is given, if $\left|\varepsilon_{I_j}\right| < \varepsilon$ and each $I_j \geq 0$, then

$$\begin{aligned} |\varepsilon_S| &= \sum_{j=1}^{k} \frac{I_j}{S}\left|\varepsilon_{I_j}\right| < \sum_{j=1}^{k} \frac{I_j}{S}\varepsilon \\ &= \varepsilon. \end{aligned}$$

A similar result would be true if $I_j \leq 0$ for each j. However, we run into trouble if $I_j = 0$ or if some I_j's are positive and others are negative.

Notice that if the integral is zero, it does not make sense to insist on the relative error being small, since the relative error does not exist. However, it does make sense that we would like for the relative error to be small over small subintervals where the integral is positive or negative. The trouble spots are where I_j is zero; a recommended approach for these subintervals is that they be subdivided anyway (without calculating the relative error), and that the intervals be subdivided only to a maximum number of times.

Another interesting situation, discovered by Kevin Little, is that sometimes the relative error does not decrease in particular subintervals in the subdivision. For example, if $g(x) = x^2$, the trapezoid approximation with one subdivision to $\int_0^a x^2 dx$ has relative error

$$\frac{\int_0^a x^2 dx - \frac{1}{2}a\left(a^2\right)}{\int_0^a x^2 dx} = \frac{\frac{1}{3}a^3 - \frac{1}{2}a^3}{\frac{1}{3}a^3} = -\frac{1}{2},$$

independent of a ! So no amount of subdividing would reduce the relative error in that first subdivision. A similar situation will always occur when the second derivative of the function is zero at one endpoint. Again, to avoid creating an infinite loop in a quadrature program, we should insist that there are a maximum number of subdivisions of a given interval.

Thus, a **new algorithm for adaptive trapezoid quadrature that utilizes relative error** would look something like this:

(1) Let RE_0 =maximum relative error tolerance, let N be the maximum number of subdivision levels allowed (eg 25). $SUM = 0$. Let the "(unfinished interval, subdivision level) list" be the empty set. Set $[a, b]$ to be the interval under consideration. Set initial value of $SL = 1$, meaning the subdivision level of the first interval under consideration to be 1.

(2) Subdivide the interval into two equal subdivisions. Calculate $TRAP\,(2)$ and $TRAP\,(1)$ for the interval and the subdivided interval.

(3) If $TRAP\,(2) \neq 0$:

Calculate $RE\,(2) = \frac{E(2)}{TRAP(2)} = \frac{1}{3}\left(1 - \frac{TRAP(1)}{TRAP(2)}\right)$. *** note that $RE\,(2)$ is the estimated relative error in $TRAP\,(2)$.

Otherwise, if $TRAP\,(2) = 0$, skip to Step (5)

(4) If $|RE\,(2)| < RE_0$, then

(a) Let $SUM = SUM + TRAP\,(2) + E\,(2)$. *** Recall $E\,(2) = \frac{1}{3}\,(TRAP\,(2) - TRAP\,(1))$, so this can be simplified.

(b) If there are any other intervals left in the "(unfinished interval, subdivision level) list" , go back to Step (2) for the next interval on that list, setting SL to the subdivision level that is paired with it, and also delete that pair from the "(unfinished interval, subdivision level) list".

(c) Otherwise, go to Step (6).

Otherwise, if $|RE\,(2)| \geq RE_0$, go to Step (5)

(5) Let $SL = SL + 1$.

If $SL > N$, then

(a) Let $SUM = SUM + TRAP\,(2) + E\,(2)$.

(b) If there are any other intervals left in the "(unfinished interval, subdivision level) list" , go back to Step (2) for the next interval on that list, setting SL to the subdivision level that is paired with it, and also delete that pair from the "(unfinished interval, subdivision level) list".

Otherwise, if the "(unfinished interval, subdivision level) list" is empty, go to Step (6).

Otherwise, if $SL \leq N$,

(a) Subdivide the interval into two subintervals.

(b) Add the second subinterval to the "(unfinished interval, subdivision level) list", paired with SL as subdivision level for that subinterval.

(c) Go back to Step (2), using the first subinterval as the new interval under consideration.

(6) Stop; SUM is the approximation to the integral.

Note that adaptive quadrature is more efficient than standard equal subdivision quadrature methods, but it is not perfect. Counterexamples can be constructed that thwart any numerical quadrature method.

3.4. **Multivariable integrals and Monte Carlo integration.** So far we have only considered integrals of functions of one variable. Any of the techniques used can be applied to functions of N variables over regions in N-dimensional space, by using an iterated integral. For example, to integrate the function x^2z over the upper half of the unit ball in $\mathbb{R}^3$, one could do the following iterated integral:

$$I = \int_{x=-1}^{1} \int_{y=-\sqrt{1-x^2}}^{\sqrt{1-x^2}} \int_{z=0}^{\sqrt{1-x^2-y^2}} x^2 z \; dz \; dy \; dx.$$

This function could be integrated by doing the single variable integral

$$I = \int_{-1}^{1} F(x) \; dx,$$

where

$$F(x) = \int_{y=-\sqrt{1-x^2}}^{\sqrt{1-x^2}} \int_{z=0}^{\sqrt{1-x^2-y^2}} x^2 z \; dz \; dy.$$

In order to employ the quadrature methods, the function $F(x)$ would need to be evaluated at several points. To find each single function evaluation, one would need to employ the quadrature methods to the single variable integral

$$\begin{aligned} F(x) &= \int_{y=-\sqrt{1-x^2}}^{\sqrt{1-x^2}} G_x(y) \; dy, \text{ where} \\ G_x(y) &= \int_{z=0}^{\sqrt{1-x^2-y^2}} x^2 z \; dz. \end{aligned}$$

In order to employ the quadrature method, the function $G_x(y)$ would have to be evaluated at several points, which would involve using a numerical quadrature method on the integral used to define $G_x(y)$.

As we can see from this example, numerical quadrature can be used to evaluate iterated multidimensional integrals, but many more function evaluations are needed to get the integral to a desired level of accuracy. Roughly, if a desired level of accuracy is achieved for a single variable integral with N function evaluations, the same level of accuracy could be achieved for a k-dimensional integral with approximately N^k function evaluations. As a result, the use of ordinary quadrature methods for multidimensional integrals is not efficient if the number of dimensions is large (say, above 7).

An alternate method of approximating a definite integral in any number of dimensions utilizes the theory of probability and is called **Monte Carlo Integration**. We illustrate this method with a **one-variable Monte Carlo algorithm** to calculate $\int_a^b g(x) \; dx$. We will need to know in advance that the graph of $y = g(x)$ over $[a, b]$ is contained in the range $c \leq y \leq d$. The integral is the same as

$$\int_a^b g(x) \; dx = (b-a)(d-c) P + c(b-a),$$

where P is the probability that a random point in the bounding box $a \leq x \leq b$, $c \leq y \leq d$ is below the graph $y = g(x)$. Here is the algorithm:

(1) Given: the function $g(x)$, the interval $[a,b]$, and a bounding interval $[c,d]$ designed so that the range satisfies $g([a,b]) \subset [c,d]$. Initial variables are $count = 0$, $n =$ total number of iterations.
(2) Do this a total of n times:
   (a) Choose a random point (x,y) in the box $a \leq x \leq b$, $c \leq y \leq d$. To do this, use a random number generator $RAND()$ to produce random numbers in $(0,1)$, and then convert by the formulas
$$\begin{aligned} x &= (b-a)\, RAND() + a \\ y &= (d-c)\, RAND() + c. \end{aligned}$$
   (b) If $y < g(x)$, set $count = count + 1$. Back to (a).
(3) Approximation to the integral is $Ans = (b-a)(d-c)\frac{count}{n} + c(b-a)$.

By the laws of probability and statistics, as n increases, the standard deviation of the answers obtained by this method will decrease as $\sigma_n = \frac{\sigma}{\sqrt{n}}$, where σ is a fixed constant. This means that roughly, the error in this method is asymptotic to $\frac{\text{constant}}{\sqrt{n}}$, where n is the number of function evaluations. Clearly, this is very slow convergence, as the number of function evaluations must increase by a factor of 4 in order to obtain twice the accuracy. The good news about this method is that this rate of convergence does not change as the number of dimensions increases. Also, no assumptions on the differentiability of the integrand $g(x)$ are required to obtain an estimate of the accuracy.

We now show the algorithm for calculating the multidimensional integral $\int_{\text{box}} g(x_1, x_2, ..., x_k)\ dx_1\, dx_2\, ...\, dx_k$. Note that any multidimensional integral over a bounded region can be changed to an integral over a box by replacing the function with 0 if it is outside of its region of integration.

(1) Given: the function $g(x_1, x_2, ..., x_k)$, the region of integration $a_j \leq x_j \leq b_j$ for $1 \leq j \leq k$, and a bounding interval $[b,c]$ designed so that the range satisfies $\text{Im}(g) \subset [c,d]$. Initial variables are $count = 0$, $n =$ total number of iterations.
(2) Do this a total of n times:
   (a) Choose a random point $(x_1, x_2, ..., x_k, y)$ in the box $a_j \leq x_j \leq b_j$, $c \leq y \leq d$. To do this, use a random number generator $RAND()$ to produce random numbers in $(0,1)$, and then convert by the formulas
$$\begin{aligned} x_j &= (b_j - a_j)\, RAND() + a_j \\ y &= (d-c)\, RAND() + c. \end{aligned}$$
   (b) If $y < g(x_1, x_2, ..., x_k)$, set $count = count + 1$. Back to (a).
(3) Approximation to the integral is $Ans = \prod_{j=1}^{k} (b_j - a_j)(d-c)\frac{count}{n} + c\prod_{j=1}^{k}(b_j - a_j)$.

As before, the error in this method is asymptotic to $\frac{\text{constant}}{\sqrt{n}}$, where n is the number of function evaluations. By comparison, suppose that Simpson's Rule is used in an iterated integral. Then to produce an error that is roughly $\frac{\text{constant}}{m^4}$, for a one-dimensional integral, approximately m function evaluations are needed. Thus, approximately m^k function evaluations would be needed in a k-dimensional integral. If $n = m^k$, then $\frac{\text{constant}}{m^4} = \frac{\text{constant}}{n^{4/k}}$, so that for Simpson's Rule, the error in the approximation to a k-dimensional integral is asymptotic to $\frac{\text{constant}}{n^{4/k}}$ for n function evaluations, so that the Monte Carlo method should be faster for dimensions 9 and above. For example, in a physics integral over the phase space with M

particles, there may be a region in $6M$-dimensional space, so already with only two particles, the Monte Carlo method is the more efficient method.

4. Solutions to algebraic and transcendental equations

Often one must solve algebraic and transcendental equations that do not have an analytic solution, and numerical methods abound to obtain these. Algebraic equations are those such as $x^5 + 5x = 5$ or $\sqrt{x} + \sqrt[3]{x} = 7$ whose solutions are those of polynomial equations of the form $p(x) = 0$, where p is a polynomial. Transcendental equations such as $e^x = x^3$ are those whose solutions are not roots of any polynomial equations. In either case, we are trying to solve equations of the type $f(x) = 0$, where f is a known function. All equations in one variable can be expressed this way; $F(x) = G(x)$ if and only if $f(x) = 0$ with $f = F - G$. We will begin by discussing numerical solutions to equations of one real variable and then will also briefly discuss complex roots of functions of complex variables.

4.1. **Method of bisection.** The simplest means of finding a solution to $f(x) = 0$ is the method of bisection. Here the function $f(x)$ is assumed to be continuous on the interval $[x_1, x_2]$, and we assume that $f(x_1)$ and $f(x_2)$ have opposite signs. The method works as follows:

(1) Initialization: $a = x_1, b = x_2, n = 3, N =$ maximum number of iterations/subdivisions.
(2) Start with an interval $[a, b]$ such that $f(a)$ and $f(b)$ have opposite signs. By the intermediate value theorem, there is an $x \in (a, b)$ such that $f(x) = 0$.
(3) Set $x_n = \frac{a+b}{2}$. Evaluate $f(x_n)$.
(4) If $f(x_n) = 0$ or $n > N+2$, STOP; we have found a root or an approximation to a root.

Otherwise, if $f(x_n) \neq 0$, it has the opposite sign to one of $f(a)$ or $f(b)$ (call it $f(z)$). Then, let $a = \min\{z, x_n\}$, $b = \max\{z, x_n\}$, $n = n + 1$, and go to Step (2).

In this process, notice that $[a, b]$ has length $b - a$, and the distance between x_{n-1} and $x_n$is by construction $\frac{x_2-x_1}{2^{n-2}}$. Thus a zero should be within $\frac{x_2-x_1}{2^N}$ of the last x-value found (x_{N+2}). So we see that if x_* is the zero of f, then $|x_n - x_*| \leq \frac{x_2-x_1}{2^{n-2}}$, so the sequence $\{x_j\}$ converges to the zero x_* (if it were allowed to go on forever). Let $e_N = \frac{x_2-x_1}{2^N}$ be the maximal error at the N^{th} iteration. Observe that $\frac{e_{N+1}}{e_N} = \frac{1}{2}$. In general, **an iteration method converges with order** d if the maximal error E_N at the N^{th} stage satisfies

$$\limsup_{k\to\infty} \frac{|E_{k+1}|}{|E_k|^d} = C \neq 0.$$

We now try to solve for d in the expression above:

$$|E_{k+1}| \leq C\,|E_k|^d$$

implies (since log is an increasing function)

$$\log|E_{k+1}| \leq \log\left(C\,|E_k|^d\right) = d\log(|E_k|) + \log(C),$$

and if $|E_k|$ is sufficiently small, $\log(|E_k|) < 0$, so we may divide by this to get

$$\frac{\log|E_{k+1}|}{\log|E_k|} \geq d + \frac{\log(C)}{\log|E_k|}.$$

Thus, we have shown the formula in the proposition below.

Proposition 8. *If an iteration method has error E_k at the k^{th} iteration, and if $E_k \to \infty$ as $k \to \infty$, then the order of convergence d is given by the formula*

$$d = \lim \inf_{k\to\infty} \frac{\log(|E_{k+1}|)}{\log(|E_k|)}.$$

In every case, we assume the error E_k is never zero (because otherwise, the method would stop, and the convergence would be much quicker than the general case!).

In the method of bisection, the maximal error e_{k+1} satisfies

$$\frac{e_{k+1}}{e_k} = \frac{1}{2},$$

so that the method of bisection has convergence of order 1, or **linear convergence**.

Some important things to notice about the method of bisection are:

(1) We must find two values of x such that f has different signs at those values in order to start the method.
(2) With the last comment in mind, we will be unable to find roots that such that $f(x) \geq 0$ near the root or $f(x) \leq 0$ near the root (for example, $f(x) = (x-c)^2$).
(3) We will not easily be able to find multiple roots within an interval.
(4) If the function has a jump discontinuity across the value 0, this method will detect that jump discontinuity.
(5) Very little information about the function is needed in this method.

4.2. **Regula Falsi: the method of false position.** Another method of finding a root of the equation $f(x) = 0$ is called the **method of false position**, or **regula falsi**. The idea here is that one may guess that if f has opposite signs at x_1 and x_2 , $|f(x_1)|$ is much larger than $|f(x_2)|$, one would expect a root between them to be closer to x_2. The method works by finding the intersection point $(x_3, 0)$ of the line connecting $(x_1, f(x_1))$ with $(x_2, f(x_2))$ with the x-axis. Note that

$$(0 - f(x_1)) = \left(\frac{f(x_2) - f(x_1)}{x_2 - x_1}\right)(x_3 - x_1),$$

so that

$$\begin{aligned} x_3 &= x_1 - \frac{f(x_1)(x_2 - x_1)}{f(x_2) - f(x_1)} \\ &= \frac{x_1(f(x_2) - f(x_1)) - f(x_1)(x_2 - x_1)}{f(x_2) - f(x_1)} \\ &= \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)}. \end{aligned}$$

The method works as follows:

(1) Initialization: $a = x_1$, $b = x_2$, $n = 3$, $N =$ maximum number of iterations/subdivisions.
(2) Start with an interval $[a, b]$ such that $f(a)$ and $f(b)$ have opposite signs. By the intermediate value theorem, there is an $x \in (a, b)$ such that $f(x) = 0$.
(3) Set

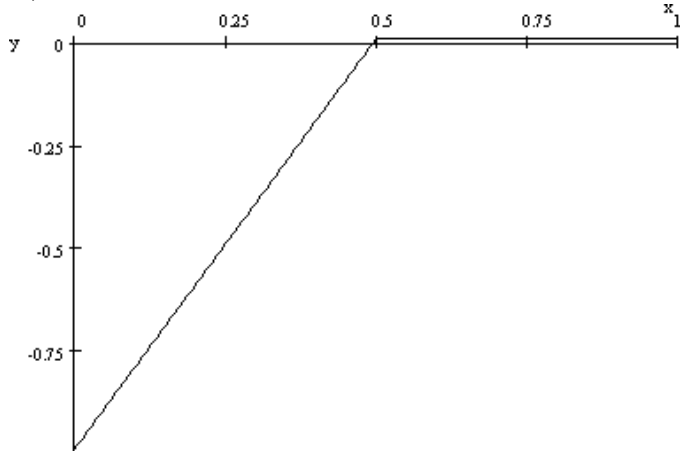
$$x_n = \frac{x_{n-2} f(x_{n-1}) - x_{n-1} f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}.$$

Evaluate $f(x_n)$.

(4) If $f(x_n) = 0$ or $n > N+2$, STOP; we have found a root or an approximation to a root.

Otherwise, if $f(x_n) \neq 0$, it has the opposite sign to one of $f(a)$ or $f(b)$ (call it $f(z)$). Then, let $a = \min\{z, x_n\}$, $b = \max\{z, x_n\}$, $n = n+1$, and go to Step (2).

Geometrically, it appears that the method of false position could be really fast — much faster than the method of bisection. However, certain examples yield extremely slow convergence. For example, if the function looks like this, the convergence will be very slow (at first):



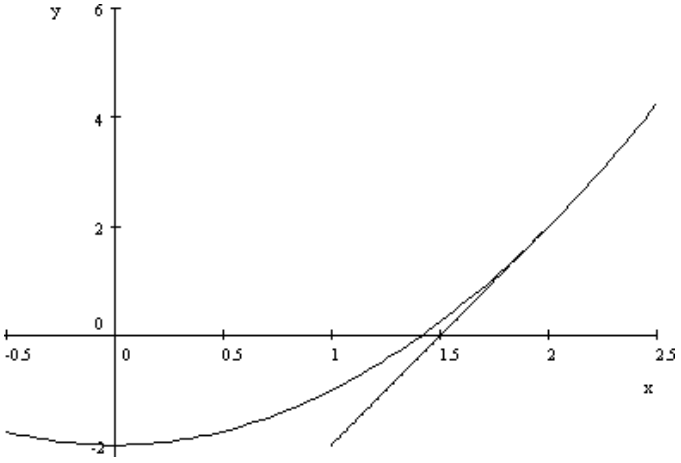
Some important things to notice about the method of false position are:

(1) We must find two values of x such that f has different signs at those values in order to start the method.
(2) We will be unable to find roots that such that $f(x) \geq 0$ near the root or $f(x) \leq 0$ near the root (for example, $f(x) = (x-c)^2$).
(3) We will not easily be able to find multiple roots within an interval.
(4) If the function has a jump discontinuity across the value 0, this method will detect that jump discontinuity.
(5) More information about the function is needed in this method than in the method of bisection.
(6) With the example above in mind, note that in certain cases we will not be able to control how fast this method converges.

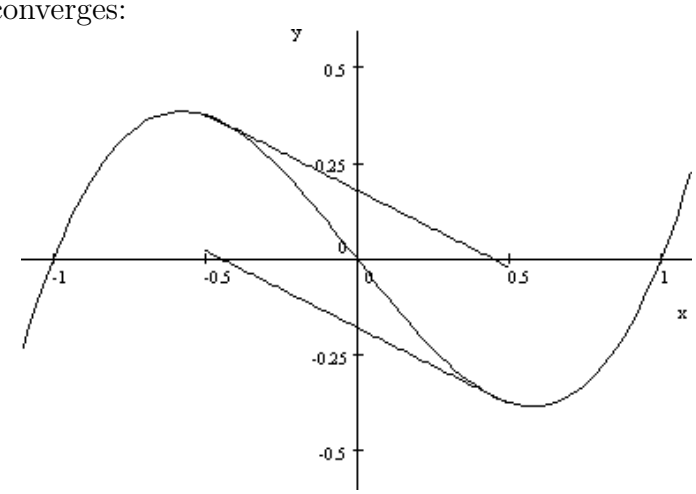
4.3. **Newton-Raphson Method.** Another method of finding a solution to an equation of the form $g(x) = 0$ is the **Newton-Raphson Method**. For this method to work, the function must be differentiable, and we must have a way of calculating the derivative function $g'(x)$. The idea is that we start with an initial guess x_0 for a root, and then in general for $k \geq 0$, x_{k+1} is obtained from x_k by setting $(x_{k+1}, 0)$ to be the point of intersection of the x-axis with the tangent line to $y = g(x)$ at the point $(x_k, g(x_k))$. That is

$$\begin{aligned} 0 - g(x_k) &= g'(x_k)(x_{k+1} - x_k), \text{ or} \\ x_{k+1} &= x_k - \frac{g(x_k)}{g'(x_k)}. \end{aligned}$$

In order for this method to be defined and to converge, the value of $g'(x_k)$ must never be zero, and the initial guess should be close to the root. An illustration of this technique is shown in the picture below, where $x_k = 2$, and $x_{k+1} = 1.5$ for the given function.



One would guess that the Newton-Raphson method would converge rapidly, but there are some potential problems with convergence. First, if the initial point is not chosen sufficiently close to the root, we could have this problem, where the sequence (x_k) oscillates but never converges:



Other examples can show even more wild behavior.

We do know that the convergence of the Newton-Raphson method is very fast, if we do know in advance that it converges. The following theorem shows the order of the Newton-Raphson method.

Theorem 9. *Suppose that g is C^2 (second derivative is continuous), and suppose that x_* is a root of g such that $\lim_{n\to\infty} x_n = x_*$, where x_n is determined from an initial guess x_0 and from the Newton-Raphson iteration. Let $e_n = x_* - x_n$, the error in the approximation x_n. Then:*

(1) *If $g'(x_*) \neq 0$, then*

$$\lim_{n\to\infty} \frac{e_{n+1}}{(e_n)^2} = -\frac{g''(x_*)}{2g'(x_*)} .$$

(2) *If $g'(x_*) = 0$ and $g''(x_*) \neq 0$,*

$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n} = \frac{1}{2} .$$

(3) *In general, if* $g'(x_*) = 0, g''(x_*) = 0, ..., g^{(k-1)}(x_*) = 0, g^{(k)}(x_*) \neq 0$, *and if* g *is* C^k, *then*

$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n} = \frac{k-1}{k} .$$

Proof. If we calculate the first degree Taylor polynomial with remainder for g with center at x_n and evaluated at x_* , we get

$$0 = g(x_*) = g(x_n) + g'(x_n)(x_* - x_n) + \frac{g''(\xi)}{2}(x_* - x_n)^2 ,$$

where ξ is between x and x_n. Since the Newton-Raphson iteration is known to converge, $g'(x_n)$ is nonzero for each n, so we may divide by $g'(x_n)$:

$$\begin{aligned} 0 &= \frac{g(x_n)}{g'(x_n)} + (x_* - x_n) + \frac{g''(\xi)}{2g'(x_n)}(x_* - x_n)^2 , \text{ or} \\ x_* - \left(x_n - \frac{g(x_n)}{g'(x_n)}\right) &= -\frac{g''(\xi)}{2g'(x_n)}(x_* - x_n)^2 , \text{ or} \\ e_{n+1} &= x_* - x_{n+1} = -\frac{g''(\xi)}{2g'(x_n)}(e_n)^2 . \end{aligned}$$

Thus, if $g'(x_*) \neq 0$, since g' and g'' are continuous, we have

$$\lim_{n\to\infty} \frac{e_{n+1}}{(e_n)^2} = \lim_{n\to\infty} -\frac{g''(\xi)}{2g'(x_n)} = -\frac{g''(x_*)}{2g'(x_*)}.$$

On the other hand, if g has a root of order k at x_* , then we have $g(x) = (x - x_*)^k p(x)$, where $p(x)$ is C^2 and $p(x_*) \neq 0$, then

$$g'(x) = k(x - x_*)^{k-1} p(x) + (x - x_*)^k p'(x) = k(x - x_*)^{k-1} p(x) + \mathcal{O}\left(|x - x_*|^k\right),$$

and

$$\begin{aligned} g''(x) &= k(k-1)(x - x_*)^{k-2} p(x) + 2k(x - x_*)^{k-1} p'(x) + (x - x_*)^k p''(x) \\ &= k(k-1)(x - x_*)^{k-2} p(x) + \mathcal{O}\left(|x - x_*|^{k-1}\right). \end{aligned}$$

In general, for $1 \leq m \leq k$

$$g^{(m)}(x) = \frac{k!}{(k-m)!}(x - x_*)^{k-m} p(x) + \mathcal{O}\left(|x - x_*|^{k-m+1}\right).$$

Above, the "big $\mathcal{O}$" notation means that if $f(x) = \mathcal{O}\left(|x - x_*|^j\right)$, then there exists a constant C such that for x in the interval under consideration, $|f(x)| \leq C|x - x_*|^j$. The Taylor polynomial with remainder to the $k-1$ degree is

$$\begin{aligned} 0 = g(x_*) = g(x_n) + g'(x_n)(x_* - x_n) + \frac{g''(x_n)}{2}(x_* - x_n)^2 + ... \\ + \frac{g^{(k-1)}(x_n)}{(k-1)!}(x_* - x_n)^{k-1} + \frac{g^{(k)}(\tau)}{k!}(x_* - x_n)^k \end{aligned}$$

for some τ between x_* and x_n . Then, dividing by $g'(x_n)$ and solving for $x_* - \left(x_n - \frac{g(x_n)}{g'(x_n)}\right)$, we get

Thus,

$$\begin{aligned}
x_* - \left(x_n - \frac{g(x_n)}{g'(x_n)}\right) &= -\frac{g''(x_n)}{2g'(x_n)}(x_* - x_n)^2 - ... - \frac{g^{(k-1)}(x_n)}{(k-1)!g'(x_n)}(x_* - x_n)^{k-1} \\
&\quad -\frac{g^{(k)}(\tau)}{k!g'(x_n)}(x_* - x_n)^k \\
&= -\frac{k(k-1)(x_n - x_*)^{k-2} p(x_n) + \mathcal{O}\left(|x_n - x_*|^{k-1}\right)}{2k(x_n - x_*)^{k-1} p(x_n) + \mathcal{O}\left(|x_n - x_*|^k\right)}(x_* - x_n)^2 \\
&\quad -\frac{k(k-1)(k-2)(x_n - x_*)^{k-3} p(x_n) + \mathcal{O}\left(|x_n - x_*|^{k-2}\right)}{3!k(x_n - x_*)^{k-1} p(x_n) + \mathcal{O}\left(|x_n - x_*|^k\right)}(x_* - x_n)^3 \\
&\quad ... \\
&\quad ... - \frac{k!p(\tau) + \mathcal{O}(|\tau - x_*|)}{k!\left(k(x_n - x_*)^{k-1} p(x_n) + \mathcal{O}\left(|x_n - x_*|^k\right)\right)}(x_* - x_n)^k.
\end{aligned}$$

Lumping together all higher order terms, and simplifying, we have

$$\begin{aligned}
e_{n+1} = x_* - x_{n+1} = \frac{(k-1)}{2}e_n &\quad -\frac{(k-1)(k-2)}{3!}e_n \\
&\quad ... + (-1)^k \frac{p(\tau)}{kp(x_n)}e_n + \mathcal{O}\left(|e_n|^2\right)
\end{aligned}$$

Thus,

$$\begin{aligned}
\lim_{n\to\infty} \frac{e_{n+1}}{e_n} &= \lim_{n\to\infty}\left(\frac{k-1}{2} - \frac{(k-1)(k-2)}{3!} + ... + (-1)^k \frac{p(\tau)}{kp(x_n)}\right) \\
&= \left(\frac{k-1}{2} - \frac{(k-1)(k-2)}{3!} + ... + (-1)^k \frac{1}{k}\right) \\
&= \frac{k-1}{k}.
\end{aligned}$$

□

Remark 10. *This theorem implies that if $g'(x_*) \neq 0$ (ie the root is a simple root), the order of convergence of the Newton-Raphson method is at least two, and if the root is a multiple root, the convergence has order one.*

Some comments about the Newton-Raphson method:

(1) Unlike the method of false position and the method of bisection, we do not need to find two values of x such that f has different signs at those values in order to start the method. We need only an initial point (that is sufficiently close to a root).
(2) We are able to find roots that such that $f(x) \geq 0$ near the root or $f(x) \leq 0$ near the root (for example, $f(x) = (x-c)^2$).
(3) By choosing different initial points x_0, we may be able to find multiple roots within an interval. But there is no guarantee that we will find all of them.
(4) The function must be differentiable to use this method.
(5) In order to use this method, we need to compute values of the derivative of the function and of the function.

(6) With the examples above in mind, we will often not get any convergence from this method.
(7) This has highest order convergence (2) among the methods we have seen, if the root is a simple root. If the root is a multiple root, the convergence has order 1.

4.4. **Modified Newton-Raphson method.** In order to achieve faster convergence for multiple roots, the Newton-Raphson method can be modified. First consider the case where a function $g(x)$ has a root of order $k \geq 1$ at $x = \alpha$. This means that

$$g(x) = (x-\alpha)^k p(x),$$

where $p(\alpha) \neq 0$. Suppose also that p is C^{k+1}, which means the $(k+1)^{\text{th}}$ derivative is continuous. Observe that

$$\begin{aligned} \frac{g(x)}{g'(x)} &= \frac{(x-\alpha)^k p(x)}{k(x-\alpha)^{k-1} p(x) + (x-\alpha)^k p'(x)} \\ &= \frac{(x-\alpha)}{k + (x-\alpha)\frac{p'(x)}{p(x)}}. \end{aligned}$$

Then

$$\begin{aligned} x - \alpha &= \frac{g(x)}{g'(x)} \left(k + (x-\alpha)\frac{p'(x)}{p(x)} \right) \\ &= k\frac{g(x)}{g'(x)} + \frac{g(x)p'(x)}{g'(x)p(x)}(x-\alpha), \end{aligned}$$

so that

$$\alpha - \left(x - k\frac{g(x)}{g'(x)} \right) = -\frac{g(x)p'(x)}{g'(x)p(x)}(x-\alpha).$$

Now, we have

$$\frac{g(x)p'(x)}{g'(x)p(x)} = \frac{(x-\alpha)p'(x)}{\left(k + (x-\alpha)\frac{p'(x)}{p(x)}\right)p(x)} = \frac{p'(x)}{(kp(x) + (x-\alpha)p'(x))}(x-\alpha).$$

Thus,

$$\alpha - \left(x - k\frac{g(x)}{g'(x)} \right) = -\frac{p'(x)}{(kp(x) + (x-\alpha)p'(x))}(x-\alpha)^2.$$

So if we modify the Newton-Raphson method by letting

$$x_{n+1} = x_n - k\frac{g(x_n)}{g'(x_n)},$$

Then, by the equation above, the error $e_n = \alpha - x_n$ satisfies

$$e_{n+1} = -\frac{p'(x_n)}{(kp(x_n) - p'(x_n)e_n)}(e_n)^2.$$

If this new method gives a convergent sequence so that $x_n \to \alpha$, $e_n \to 0$, then

$$\lim_{n\to\infty} \frac{e_{n+1}}{(e_n)^2} = \lim_{n\to\infty} -\frac{p'(x_n)}{(kp(x_n) - p'(x_n)e_n)} = -\frac{p'(\alpha)}{kp(\alpha)}.$$

We have shown the following result.

Proposition 11. *If α is a root of order k of the function $g(x) = (x-\alpha)^k p(x)$, and if we fix x_0 and let*

$$x_{n+1} = x_n - k\frac{g(x_n)}{g'(x_n)},$$

Then if $x_n \to \alpha$, we have that the error $e_n = x_n - \alpha$ satisfies

$$\lim_{n\to\infty} \frac{e_{n+1}}{(e_n)^2} = -\frac{p'(\alpha)}{kp(\alpha)},$$

so that this convergence has order two (at least).

Now, you may ask: what happens if we do not know the order of the root α of the function $g(x)$, which is generally the case? Another trick is used to overcome this difficulty. Define $k(x)$ for x near α by

$$k(x) = \frac{\log|g(x)|}{\log|g(x)| - \log|g'(x)|}.$$

The following result comes in handy.

Lemma 12. *Suppose that a root α of $g(x)$ has multiplicity m, so that $g(x) = (x-\alpha)^m p(x)$ with $p(\alpha) \neq 0$. Then the function $k(x)$ defined above satisfies*

$$\lim_{x\to\alpha} k(x) = m.$$

Proof. We have

$$k(x) = \frac{\log|(x-\alpha)^m p(x)|}{\log|(x-\alpha)^m p(x)| - \log\left|m(x-\alpha)^{m-1}p(x) + (x-\alpha)^m p'(x)\right|},$$

and the denominator is

$$\begin{aligned}\log\left|\frac{(x-\alpha)^m p(x)}{m(x-\alpha)^{m-1}p(x) + (x-\alpha)^m p'(x)}\right| &= \log\left|\frac{(x-\alpha)}{m + (x-\alpha)\frac{p'(x)}{p(x)}}\right| \\ &= \log|x-\alpha| - \log\left|m + (x-\alpha)\frac{p'(x)}{p(x)}\right|,\end{aligned}$$

so that

$$\begin{aligned}k(x) &= \frac{\log|(x-\alpha)^m p(x)|}{\log|x-\alpha| - \log\left|m + (x-\alpha)\frac{p'(x)}{p(x)}\right|} \\ &= \frac{m\log|x-\alpha| + \log|p(x)|}{\log|x-\alpha| - \log\left|m + (x-\alpha)\frac{p'(x)}{p(x)}\right|}\end{aligned}$$

As $x \to \alpha$, $\log|x-\alpha| \to -\infty$, and the other terms are bounded, so

$$\lim_{x\to\alpha} k(x) = \lim_{x\to\alpha} \frac{m\log|x-\alpha|}{\log|x-\alpha|} = m.$$

□

The **modified Newton-Raphson method** for finding a root of unknown multiplicity for the function $g(x)$ is defined by an initial guess x_0 and the equation

$$x_{n+1} = x_n - \frac{g(x_n)\log|g(x_n)|}{g'(x_n)(\log|g(x_n)| - \log|g'(x_n)|)}.$$

Then, if this iteration converges, then it achieves second order convergence, by the lemma and proposition above. In summary:

Proposition 13. *If α is a root of the function $g(x)$, and if we fix x_0 near α and let*

$$x_{n+1} = x_n - \frac{g(x_n)\log|g(x_n)|}{g'(x_n)(\log|g(x_n)| - \log|g'(x_n)|)},$$

Then if $x_n \to \alpha$, we have that the error $e_n = x_n - \alpha$ satisfies

$$\lim_{n\to\infty} \frac{e_{n+1}}{(e_n)^2} = -\frac{p'(\alpha)}{kp(\alpha)},$$

so that this convergence has order two (at least), where k is the multiplicity of the zero at α and $g(x) = (x-\alpha)^k p(x)$ with $p(\alpha) \neq 0$.

We note that if one is programming a computer to find roots using this method, it may be worthwhile to calculate $k(x) = \frac{\log|g(x)|}{\log|g(x)| - \log|g'(x)|}$ for each $x = x_n$, and if this number approaches a limit that is a positive integer, the program could switch to the simpler formula $x_{n+1} = x_n - \frac{kg(x_n)}{g'(x_n)}$.

4.5. **Secant method.** The secant method of finding roots to an equation of the form $f(x) = 0$ is similar to Newton's method but does not require knowledge of the derivative function f' . The idea is that one starts with two initial points x_0 and x_1 , and then uses the values of the function to find the intersection x_2 of the secant line (line connecting the two points) and the x-axis. In general, given x_{j-1} and x_j , we define x_{j+1} to satisfy the formula

$$\begin{aligned} -f(x_j) &= \frac{f(x_j) - f(x_{j-1})}{x_j - x_{j-1}}(x_{j+1} - x_j), \text{ or} \\ x_{j+1} &= x_j - f(x_j)\frac{x_j - x_{j-1}}{f(x_j) - f(x_{j-1})} \\ x_{j+1} &= \frac{x_{j-1}f(x_j) - x_j f(x_{j-1})}{f(x_j) - f(x_{j-1})}. \end{aligned}$$

This iteration is called the **secant method** of iteration. Notice that this is the same formula as regula falsi, but in this case we do not worry about whether $f(x_j)$ or $f(x_{j-1})$ are opposite signs or whether they are the same sign.

This method has a very interesting order of convergence. To find this, observe that if x is the root to be found,

$$\begin{aligned} e_{j+1} &= x - x_{j+1} = x - \frac{x_{j-1}f(x_j) - x_j f(x_{j-1})}{f(x_j) - f(x_{j-1})} \\ &= x\left(\frac{f(x_j)}{f(x_j) - f(x_{j-1})} + \frac{-f(x_{j-1})}{f(x_j) - f(x_{j-1})}\right) - \frac{x_{j-1}f(x_j) - x_j f(x_{j-1})}{f(x_j) - f(x_{j-1})} \\ &= \frac{e_{j-1}f(x_j) - e_j f(x_{j-1})}{f(x_j) - f(x_{j-1})}. \end{aligned}$$

Expanding using a Taylor polynomial, we have

$$\begin{aligned} f(x_j) &= f(x-e_j) = f(x) - f'(x)(e_j) + \frac{1}{2}f''(x)(e_j)^2 + \mathcal{O}\left(|e_j|^3\right) \\ &= -f'(x)(e_j) + \frac{1}{2}f''(x)(e_j)^2 + \mathcal{O}\left(|e_j|^3\right) \end{aligned}$$

and similarly with $f(x_{j-1})$. Then

$$\begin{aligned} e_{j+1} &= \frac{e_{j-1}f(x-e_j) - e_j f(x-e_{j-1})}{f(x-e_j) - f(x-e_{j-1})} \\ &= \frac{e_{j-1}\left(-f'(x)e_j + \frac{1}{2}f''(x)e_j^2 + \mathcal{O}\left(|e_j|^3\right)\right) - e_j\left(-f'(x)e_{j-1} + \frac{1}{2}f''(x)e_{j-1}^2 + \mathcal{O}\left(|e_{j-1}|^3\right)\right)}{\left(-f'(x)e_j + \frac{1}{2}f''(x)e_j^2 + \mathcal{O}\left(|e_j|^3\right)\right) - \left(-f'(x)e_{j-1} + \frac{1}{2}f''(x)e_{j-1}^2 + \mathcal{O}\left(|e_{j-1}|^3\right)\right)} \\ &= \frac{e_{j-1}\frac{1}{2}f''(x)e_j^2 - e_j\frac{1}{2}f''(x)e_{j-1}^2 + \mathcal{O}\left(|e_{j-1}|^3\right)}{-f'(x)e_j + f'(x)e_{j-1} + \mathcal{O}\left(|e_{j-1}|^2\right)} \\ &= \frac{1}{2}\frac{e_{j-1}e_j f''(x)(e_j - e_{j-1}) + \mathcal{O}\left(|e_{j-1}|^3\right)}{-f'(x)(e_j - e_{j-1}) + \mathcal{O}\left(|e_{j-1}|^2\right)} \\ &= -e_{j-1}e_j\frac{f''(x)}{2f'(x)} + \mathcal{O}\left(|e_{j-1}|^3\right) = e_{j-1}e_j\left(-\frac{f''(x)}{2f'(x)} + \mathcal{O}\left(|e_{j-1}|\right)\right) \end{aligned}$$

Thus, we have

$$e_{j+1} = e_{j-1}e_j\left(a + \mathcal{O}\left(|e_{j-1}|\right)\right), \tag{2}$$

where

$$a = -\frac{f''(x)}{2f'(x)},$$

if it is defined (ie $f'(x) \neq 0$). In many texts, the following heuristic argument gives evidence that the order of convergence of the secant method is $\frac{1+\sqrt{5}}{2}$. If we assume that $|e_{j+1}| \approx C|e_j|^p$ for some $C > 0$ and some $p > 0$, then at the same time $|e_{j-1}| \approx C^{-1/p}|e_j|^{1/p}$, and the equation (2) implies that

$$C|e_j|^p \approx C^{-1/p}|e_j|^{1+1/p}$$

for all j, which implies $p = 1 + \frac{1}{p}$, or $p^2 - p - 1 = 0$, or $p = \frac{1+\sqrt{5}}{2}$ (the other solution to the quadratic is negative).

Here is a more rigorous argument. First, observe that the order of convergence of the secant method is at least 1. To see this, divide both sides of the equation (2) by e_j. (We may do this since otherwise some $e_j = 0$, and this would mean the method would terminate.) Then

$$\frac{e_{j+1}}{e_j} = e_{j-1}\left(a + \mathcal{O}\left(|e_{j-1}|\right)\right),$$

so since $e_{j-1} \to 0$ as $j \to \infty$, the order or convergence is at most 1 (and probably larger).

Next, we look more carefully at equation (2), letting $A_k = \frac{|e_{k+1}|}{|e_k|^p}$, $C_k = |a + \mathcal{O}(|e_k|)|$, $p = \frac{1+\sqrt{5}}{2} = \frac{1}{p} + 1$:

$$\begin{aligned} A_j &= \frac{|e_{j-1}|}{|e_j|^{p-1}} C_{j-1} \\ &= \frac{|e_{j-1}|}{|e_j|^{1/p}} C_{j-1} = C_{j-1} (A_{j-1})^{-1/p}. \end{aligned}$$

Letting $a_j = \log(A_j)$, $c_j = \log(C_j)$, we apply $\log(\bullet)$ to both sides of the equation above to get

$$\begin{aligned} a_j &= c_{j-1} - \frac{1}{p} a_{j-1} \\ &= c_{j-1} - \frac{1}{p}\left(c_{j-2} - \frac{1}{p} a_{j-2}\right) = c_{j-1} - \frac{1}{p} c_{j-2} + \frac{1}{p^2} a_{j-2} \\ &= c_{j-1} - \frac{1}{p} c_{j-2} + ... + \frac{(-1)^{j-1}}{p^{j-1}} c_0 + \frac{(-1)^j}{p^j} a_0. \end{aligned}$$

Lemma 14. *If $f''(x)$ and $f'(x)$ are nonzero, the sequence (a_j) converges.*

Proof. First, since $c_j = \log\left|-\frac{f''(x)}{2f'(x)} + \mathcal{O}(|e_k|)\right|$, (c_j) converges and is bounded, say $|c_j| < Q$ for all j. Since (c_j) converges, for any $\varepsilon > 0$, we may choose $N_1 > 0$ such that for any $n, m \geq N_1$, $|c_n - c_m| < \frac{\left(1-p^{-1}\right)\varepsilon}{3}$. Also, since $\frac{1}{p^{j+1}}\left(\frac{1}{1-p^{-1}}\right) 2Q \to 0$ as $j \to \infty$, we may choose $N_2 > 0$ such $\frac{1}{p^{j+1}}\left(\frac{1}{1-p^{-1}}\right) 2Q < \frac{\varepsilon}{3}$ for $j \geq N_2$. Similarly, since $\frac{1}{p^j}|a_0| \to 0$ as $j \to \infty$, we may choose N_3 such that $\frac{1}{p^j}|a_0| < \frac{\varepsilon}{3}$ for $j \geq N_3$. Letting $N = \max\{N_1, N_2, N_3\}$, then if $k \geq \ell \geq 2N + 1$,

$$\begin{aligned} |a_k - a_\ell| &= \Bigg| c_{k-1} - \frac{1}{p} c_{k-2} + ... + \frac{(-1)^{k-1}}{p^{k-1}} c_0 + \frac{(-1)^k}{p^k} a_0 \\ &\quad - c_{\ell-1} + \frac{1}{p} c_{\ell-2} + ... + \frac{(-1)^\ell}{p^{\ell-1}} c_0 + \frac{(-1)^{\ell-1}}{p^\ell} a_0 \Bigg| \\ &\leq |c_{k-1} - c_{\ell-1}| + \frac{1}{p}|c_{k-2} - c_{\ell-2}| + ... + \frac{1}{p^N}|c_{k-N-1} - c_{\ell-N-1}| \\ &\quad + \frac{1}{p^{N+1}} 2Q + \frac{1}{p^{N+2}} 2Q + ... + \frac{1}{p^{k-1}} 2Q + \frac{1}{p^k}|a_0| \\ s &< \frac{\left(1-p^{-1}\right)\varepsilon}{3}\left(\frac{1}{1-p^{-1}}\right) + \frac{1}{p^{N+1}}\left(\frac{1}{1-p^{-1}}\right) 2Q + \frac{1}{p^k}|a_0| \\ &< \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon. \end{aligned}$$

Hence, (a_j) is a Cauchy sequence and thus converges. □

By the lemma, $\lim_{j\to\infty} a_j = B$ for some limit B with $a_j = \log A_j$, and so for $p = \frac{1+\sqrt{5}}{2}$,

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = \lim_{k\to\infty} A_k = \lim_{k\to\infty} e^{a_k} = e^B.$$

This implies that the order of convergence is $p = \frac{1+\sqrt{5}}{2}$, the golden ratio. In our proof, we have relied on the fact that $f''(x)$ is nonzero. In summary:

Theorem 15. *The order of convergence of the secant method to a root x of f is $\frac{1+\sqrt{5}}{2}$ if $f''(x)$ and $f'(x)$ are nonzero and the iteration converges.*

4.6. **Iterations to fixed points.** We have seen that several methods used to find roots of the equation $f(x) = 0$ result in an iteration of the form

$$x_{j+1} = \phi(x_j, x_{j-1}, ..., x_{j-k})$$

along with an initial guesses $x_0, x_1, ..., x_k$ and where $\phi : \mathbb{R}^{k+1} \to \mathbb{R}$ is a real-valued function of $k+1$ variables. In many cases, we are able to reduce the iteration to the simple case, where $k = 0$, so that

$$x_{j+1} = \phi(x_j),$$

which ϕ a real-valued function on $\mathbb{R}$. In this case, if the iteration converges to x (i.e. $\lim_{j\to\infty} x_j = x$), then $x = \phi(x)$, and we say that we are looking for a **fixed point** of the function ϕ. Such an iteration is also called a **discrete dynamical system**.

For example, suppose that we are trying to find a solution to the equation $f(x) = x^2 - x - 1 = 0$ in the interval $[1, 2]$. We may rewrite the equation as $x^2 = x + 1$ or $x = 1 + \frac{1}{x}$ (Hmmm — looks familiar) In this case we may try an initial guess $x_0 = 1.5$ and then let $\phi(x) = 1 + \frac{1}{x}$. Then this iteration has the form

$$\begin{aligned} x_{j+1} &= \phi(x_j), \text{ or} \\ x_{j+1} &= 1 + \frac{1}{x_j} \end{aligned}$$

We obtain

$$\begin{aligned} x_1 &= \phi(1.5) = 1.666\,666\,666\,666\,666\,67 \\ x_2 &= \phi(1.666\,666\,666\,666\,666\,67) = 1.600\,000\,000\,000\,000\,00 \\ x_3 &= \phi(1.600\,000\,000\,000\,000\,00) = 1.625 \\ x_4 &= \phi(1.625) = 1.615\,384\,615\,384\,615\,38, \end{aligned}$$

and eventually the iteration converges to $\frac{1+\sqrt{5}}{2} \approx 1.618\,033\,988\,749\,894\,85$, one root of the equation $x^2 - x - 1 = 0$, or one root of $x = 1 + \frac{1}{x}$.

Sometimes such calculations do not lead to accurate or convergent results, due to the situations that we have already encountered in Newton's method. One interesting thing to note is that if we try an initial guess $x_0 = -.6$ in an attempt to find the other root

$\frac{1-\sqrt{5}}{2} \approx -0.618\,033\,988\,749\,894\,848$, we get

$$\begin{aligned} x_1 &= \phi(-.6) = -0.666\,666\,666\,666\,666\,667 \\ x_2 &= \phi(-0.666\,666\,666\,666\,666\,667) = -0.499\,999\,999\,999\,999\,999 \\ x_3 &= \phi(-0.499\,999\,999\,999\,999\,999) = -1.0 \\ x_4 &= \phi(-1) = 0.0 \\ x_5 &= \phi(0) = \infty!! \end{aligned}$$

Why did the sequence not converge the second time?

A useful result follows.

Proposition 16. *(Iteration convergence theorem) Suppose that the function ϕ is continuously differentiable and that $|\phi'(x)| \leq M < 1$ for $a \leq x \leq b$, and suppose that it is known that ϕ has a fixed point α in the interval (a,b), If $x_0 \in [a,b]$, $|\alpha - x_0| \leq |\alpha - a|$, and $|\alpha - x_0| \leq |\alpha - b|$, then the sequence defined by $x_{k+1} = \phi(x_k)$ for $k \geq 0$ converges to α, and α is the only fixed point.*

Proof. Since $|\phi'(x)| \leq M$, note that $|\phi(p) - \phi(q)| = \left|\int_q^p \phi'(x)\ dx\right| \leq \int_q^p |\phi'(x)|\, dx \leq M|p-q|$ for $p, q \in [a,b]$. Then

$$\begin{aligned} |\alpha - x_j| &= |\phi(\alpha) - \phi(x_{j-1})| \leq M|\alpha - x_{j-1}| \\ &\leq ... \leq M^j |\alpha - x_0|. \end{aligned}$$

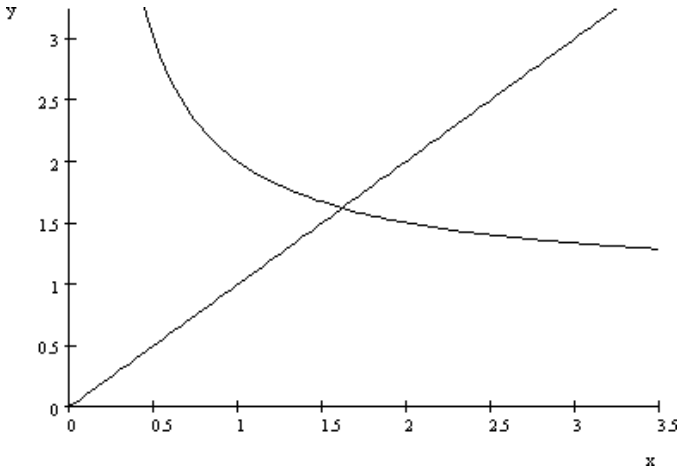
Thus, since $M < 1$, $|\alpha - x_j| \to 0$ as $j \to \infty$. Note that each $x_j \in [a,b]$, since $|\alpha - x_j| \leq |\alpha - x_0|$. □

The opposite case is considered in the following proposition.

Proposition 17. *(Iteration divergence theorem) Suppose that the function ϕ is continuously differentiable and that $|\phi'(x)| \geq N > 1$ for $x \in [a,b]$, and suppose that it is known that ϕ has a fixed point x_* in the interval $[a,b]$. Then, if $x_0 \in [a,b]$ and $x_0 \neq x_*$, then the sequence defined by $x_{k+1} = \phi(x_k)$ for $k \geq 0$ will leave $[a,b]$ eventually.*

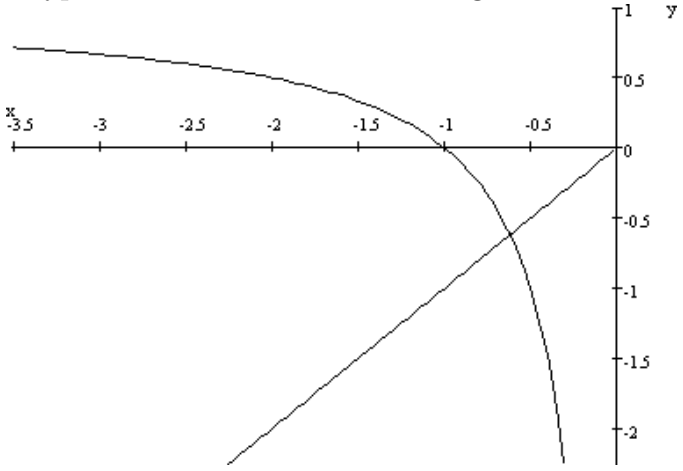
Remark 18. *Because of these two propositions, if $\phi \in C^1$ (i.e. the first derivative is continuous), if α is a fixed point of ϕ, and if $|\phi'(\alpha)| < 1$, then there is a small interval around α such that if the initial guess x_0 is in that interval, then the iteration converges to α. By contrast, if $|\phi'(\alpha)| > 1$, there is no initial point x_0 such that the iteration will converge to α, unless $x_n = \alpha$ for some finite n.*

In the example considered above, $\phi(x) = 1 + \frac{1}{x}$, and $\phi'(x) = -\frac{1}{x^2}$. The function ϕ is graphed below, along with the line $y = x$ for reference.



Note that if x_0 is chosen in the interval $[1, 2]$, the hypotheses of the iteration convergence theorem are satisfied.

Next, we graph the part of $y = \phi(x)$ with x negative. Observe that for $x_0 \in [-.8, -.1]$, the hypotheses of the iteration divergence theorem are satisfied.



Another way to explore such iterations is to use what is known as a **cobweb plot**. See http://en.wikipedia.org/wiki/File:CobwebConstruction.gif for an example with iteration function $\phi(x) = rx(1-x)$, with $r \approx 2.8$. Note that this is a famous example, called the logistic map, which is important in the study of chaos and dynamical systems.

4.7. **Roots of polynomials.** The methods above can all be used to solve an equation $f(x) = 0$, but several other methods are designed specifically for finding the real and complex zeros of a polynomial. Before introducing some of these methods, we will first investigate some inherent difficulties in numerical calculations of zeros of polynomials.

If $p(x) = c_0 + c_1 x + ... + c_n x^n$ is a polynomial, we say that the polynomial is **ill-conditioned** if small changes in the coefficients yield large changes in the roots of the polynomial. In 1959, James H. Wilkinson (see http://en.wikipedia.org/wiki/James_H._Wilkinson) showed the following example.

Let

$$\begin{aligned} w(x) &= (x-1)(x-2)...(x-20) \\ &= x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + ... + (20!) \end{aligned}$$

The zeros of this polynomial are $x = 1, 2, ..., 20$. If the coefficient -210 of x^{19} is changed by by 2^{-23} to -210.0000001192 and all other coefficients are unchanged, then the roots become (accurate to within 0.00001):

$$\begin{aligned}
&1.00000, 2.00000, 3.00000, 4.00000, 5.00000,\\
&6.00001, 6.99970, 8.00727, 8.91725, 20.84691,\\
&10.09527 \pm 0.64350i\\
&11.79363 \pm 1.65233i\\
&13.99236 \pm 2.51883i\\
&16.73074 \pm 2.81262i\\
&19.50244 \pm 1.94033i
\end{aligned}$$

Clearly, these roots have changed greatly after the small change in the x^{19} coefficient.

The example above shows that maximum precision should be used whenever one is trying to approximate roots of a polynomial, using any method.

The good news is that there is another example of Wilkinson that has a diametrically opposed property to the last one. Let

$$\begin{aligned}
w_2(x) &= \left(x - \frac{1}{2}\right)\left(x - \frac{1}{2^2}\right)...\left(x - 2^{-20}\right)\\
&= x^{20} - \left(\frac{1}{2}\frac{1 - 2^{-20}}{1 - \frac{1}{2}}\right)x^{19} + ... + 2^{-210}\\
&= x^{20} - \left(1 - 2^{-20}\right)x^{19} + ... + 2^{-210}
\end{aligned}$$

It turns out that every root of this polynomial is stable will respect to small perturbations of the polynomial's coefficients. Thus, this polynomial is **well-conditioned**.

In attempting to solve for roots of polynomials, we certainly may use any of the root-finding methods that we have analyzed that work for general sufficiently smooth functions. However, there are specific methods that are designed for polynomials that either have faster convergence properties or are able to find complex number roots as well.

The first method we will describe is called **Laguerre's Method**. Interestingly, the idea of this method is based on a very special situation, but the resulting formulas work amazingly well for general polynomials. We start with a polynomial with complex roots $x_1, ..., x_n$:

$$\begin{aligned}
P_n(z) &= (z - x_1)...(z - x_n)\\
&= \prod_{j=1}^{n}(z - x_j).
\end{aligned}$$

Let

$$\begin{aligned}
G &= \frac{P_n'(z)}{P_n(z)} = \sum_{j=1}^{n}\frac{1}{z - x_j},\\
H &= -G' = \frac{\left(P_n'(z)\right)^2 - P_n''(z)\,P_n(z)}{\left(P_n(z)\right)^2} == \sum_{j=1}^{n}\frac{1}{\left(z - x_j\right)^2}.
\end{aligned}$$

We now make the (completely ridiculous) assumption that the root x_1 is far from $x_2, ..., x_n$, and that $x_2 \approx x_3 \approx ... \approx x_n$. Suppose that the initial guess $z = x$ is very close to x_1, and let

$$\begin{aligned} a &= x - x_1 \\ b &= x - x_2 \approx x - x_3 ... \approx x - x_n. \end{aligned}$$

Taking these approximately equal numbers to be equal, we have

$$\begin{aligned} G &= \frac{1}{a} + \frac{n-1}{b} \\ H &= \frac{1}{a^2} + \frac{n-1}{b^2}. \end{aligned}$$

From the first equation we have

$$\frac{1}{n-1}\left(G - \frac{1}{a}\right) = \frac{1}{b}$$

Substituting into the equation for H, we have

$$H = \frac{1}{a^2} + \frac{n-1}{(n-1)^2}\left(G - \frac{1}{a}\right)^2,$$

so that

$$\begin{aligned} a^2 H &= 1 + \frac{a^2}{n-1}\left(G^2 - \frac{2G}{a} + \frac{1}{a^2}\right), \text{ or} \\ 0 &= n - 1 - (n-1)\, a^2 H + a^2 G^2 - 2aG + 1, \text{ or} \\ 0 &= \left(-(n-1)\, H + G^2\right) a^2 + (-2G)\, a + n \end{aligned}$$

Then the quadratic formula implies

$$\begin{aligned} a &= \frac{2G \mp \sqrt{4G^2 + 4n\left((n-1)\, H + G^2\right)}}{2\left(-(n-1)\, H + G^2\right)}, \\ a &= \frac{G \mp \sqrt{-(n-1)\, G^2 + n(n-1)\, H}}{\left(-(n-1)\, H + G^2\right)}. \end{aligned}$$

Then

$$\begin{aligned} a &= \frac{\left(G \mp \sqrt{-(n-1)\, G^2 + n(n-1)\, H}\right)\left(G \pm \sqrt{-(n-1)\, G^2 + n(n-1)\, H}\right)}{\left(-(n-1)\, H + G^2\right)\left(G \pm \sqrt{-(n-1)\, G^2 + n(n-1)\, H}\right)} \\ a &= \frac{G^2 - \left(-(n-1)\, G^2 + n(n-1)\, H\right)}{\left(-(n-1)\, H + G^2\right)\left(G \pm \sqrt{-(n-1)\, G^2 + n(n-1)\, H}\right)} \\ a &= \frac{nG^2 - n(n-1)\, H}{\left(-(n-1)\, H + G^2\right)\left(G \pm \sqrt{-(n-1)\, G^2 + n(n-1)\, H}\right)} \\ a &= \frac{n}{G \pm \sqrt{-(n-1)\, G^2 + n(n-1)\, H}}. \end{aligned}$$

Note that G and H are computed entirely from $P_n(z)$, so that ostensibly one can solve for $a = x - x_1$. Plugging into the formula above, letting $x_1 = x_{k+1}$, $x = x_k$, we have

$$x_{k+1} = x_k - \frac{nP_n(x_k)}{P_n'(x_k) \pm \sqrt{(n-1)\left((n-1)\left(P_n'(x_k)\right)^2 - nP_n''(x_k)P_n(x_k)\right)}}$$

In each step, the $\pm$ is chosen so that the denominator has absolute value as large as possible (this also includes the possibility of considering the two opposite square roots of a complex number).

Even though this method is based on a rare assumption, the application of this method to general polynomials yields very nice convergence properties. When the root to be found is simple, this method has cubic convergence. If the root has multiplicity at least two, the convergence is linear. Unlike the Newton-Raphson method, this iteration is almost guaranteed to converge to some root of the polynomial, no matter how bad the initial approximation is.

In all polynomial root-finding methods, the primary goal should be to find a single root of the polynomial. If a single root x_1 is found, then the polynomial can be divided by $(x - x_1)$ to obtain a lower degree polynomial, and the process can be continued. In what follows, the **Bairstow-Hitchcock algorithm** works to find quadratic factors of the form $x^2 + px + q$; once all such factors are found, the quadratic formula yields the approximate roots. It is especially convenient to find quadratic factors with $p, q \in \mathbb{R}$ for polynomials with real coefficients.

To find quadratic factors of a polynomial $P_n(z) = a_0 z^n + a_1 z^{n-1} + ... + a_n$, we divide by $z^2 + pz + q$ (assuming we have such a polynomial). We have

$$\begin{aligned} P_n(z) &= a_0 z^n + a_1 z^{n-1} + ... + a_n \\ &= \left(z^2 + pz + q\right)\left(b_0 z^{n-2} + b_1 z^{n-3} + ... + b_{n-2}\right) + R_1 z + R_0, \end{aligned}$$

where by comparing coefficients we have

$$\begin{aligned} b_{-1} &= 0, b_0 = a_0, \\ b_k &= a_k - pb_{k-1} - qb_{k-2} \text{ for } 1 \leq k \leq n \\ R_1 &= a_{n-1} - pb_{n-2} - qb_{n-3} = b_{n-1} \\ R_0 &= a_n - qb_{n-2} = b_n + pb_{n-1}, \end{aligned} \tag{3}$$

where the remainder coefficients satisfy $R_1 = R_0 = 0$ if and only if the quadratic factor divides $P_n(z)$ evenly. The formulas above give a recursion formula for finding the coefficients b_k of the quotient polynomial, and all of the b_js can be expressed in terms of the $a_0, ..., a_n$ and p, q. This recursion formula is actually an example of "synthetic division".

Suppose now that we start with initial guesses p, q, and we then try to improve the guesses by changing p to $p + \Delta p$ and q to $q + \Delta q$. By doing this, we hope to get the new R_0 and R_1 (functions of p and q) closer to zero. So we use a variant of Newton's method (applicable to two variables p and q) to obtain the equations below. Let the new remainder coefficients be denoted $\widetilde{R_0}$, $\widetilde{R_1}$, and so we set

$$\begin{aligned} \widetilde{R_1} &\approx R_1 + \frac{\partial R_1}{\partial p}\Delta p + \frac{\partial R_1}{\partial q}\Delta q = 0, \\ \widetilde{R_0} &\approx R_0 + \frac{\partial R_0}{\partial p}\Delta p + \frac{\partial R_0}{\partial q}\Delta q = 0. \end{aligned}$$

This approximation amounts to approximating $R_j(p,q)$ by the tangent plane to $w = R_j(p,q)$ at the initial p, q, and then choosing the Δp, Δq so that both tangent plane approximations of R_0 and R_1 are zero at $(p+\Delta p, q+\Delta q)$. Anyway, we use the formulas for R_0, R_1 to compute the partial derivatives, and we get the equations

$$\widetilde{R_1} \approx b_{n-1} + \frac{\partial b_{n-1}}{\partial p}\Delta p + \frac{\partial b_{n-1}}{\partial q}\Delta q = 0 \tag{4}$$

$$\widetilde{R_0} \approx b_n + pb_{n-1} + \left(\frac{\partial b_n}{\partial p} + b_{n-1} + p\frac{\partial b_{n-1}}{\partial p}\right)\Delta p + \left(\frac{\partial b_n}{\partial q} + p\frac{\partial b_{n-1}}{\partial q}\right)\Delta q = 0 \tag{5}$$

Subtracting p times (4) from (5), we get

$$\widetilde{R_0} - 6\widetilde{R_1} \approx b_n + \left(\frac{\partial b_n}{\partial p} + b_{n-1}\right)\Delta p + \frac{\partial b_n}{\partial q}\Delta q = 0 \tag{6}$$

We now wish to tidy-up the formulas. By differentiating (3), we get

$$b_k = a_k - pb_{k-1} - qb_{k-2}$$

$$\frac{\partial b_k}{\partial p} = -b_{k-1} - p\frac{\partial b_{k-1}}{\partial p} - q\frac{\partial b_{k-2}}{\partial p} \tag{7}$$

$$\frac{\partial b_k}{\partial q} = -p\frac{\partial b_{k-1}}{\partial q} - b_{k-2} - q\frac{\partial b_{k-2}}{\partial q} \tag{8}$$

We now have that for all $k \geq 1$,

$$\frac{\partial b_{k-1}}{\partial p} = \frac{\partial b_k}{\partial q} \tag{9}$$

Proof of (9):

$$\frac{\partial b_0}{\partial p} = 0 = \frac{\partial}{\partial q}(a_1 - pa_0) = \frac{\partial b_1}{\partial q}.$$

Assume (9) is true for k, and then the $k+1$ case is (by the formulas (7) and (8) above)

$$\begin{aligned}\frac{\partial b_k}{\partial p} &= -b_{k-1} - p\frac{\partial b_{k-1}}{\partial p} - q\frac{\partial b_{k-2}}{\partial p} \\ &= -b_{k-1} - p\frac{\partial b_k}{\partial q} - q\frac{\partial b_{k-1}}{\partial q} \\ &= \frac{\partial b_{k+1}}{\partial q}\end{aligned}$$

By induction, (9) is true for k. □

Next, let

$$c_{j-1} = -\frac{\partial b_j}{\partial p} = -\frac{\partial b_{j+1}}{\partial q};\ c_{-1} = 0, c_0 = a_0 = b_0$$

$$\overline{c_{n-1}} = -\frac{\partial b_n}{\partial p} - b_{n-1} = c_{n-1} - b_{n-1}$$

Then the formula (7) yields

$$c_k = b_k - pc_{k-1} - qc_{k-2}.$$

Thus the c_j satisfy the same recursive relations and initial conditions that the constants b_j satisfy, with the a_j replaced by the b_j. Our equations (4) and (6) become

$$\begin{aligned} b_{n-1} + \frac{\partial b_{n-1}}{\partial p}\Delta p + \frac{\partial b_{n-1}}{\partial q}\Delta q &= 0 \\ b_n + \left(\frac{\partial b_n}{\partial p} + b_{n-1}\right)\Delta p + \frac{\partial b_n}{\partial q}\Delta q &= 0 \end{aligned}$$

$$\begin{aligned} c_{n-2}\Delta p + c_{n-3}\Delta q &= b_{n-1} \\ \overline{c_{n-1}}\Delta p + c_{n-2}\Delta q &= b_n \end{aligned}$$

We can then solve for Δp and Δq, and then our new approximations to p and q are $p + \Delta p$ and $q + \Delta q$.

Put this information together to produce an iteration, we obtain the **Bairstow-Hitchcock algorithm to find a quadratic factor of $P_n(z)$:**

(1) Input coefficients $a_0, ..., a_n$, error tolerance ε, choose random p_0 and q_0 to start, and let $j = 0$.
(2) Set $b_0 = c_0 = a_0$, $b_{-1} = c_{-1} = 0$, $p = p_j$, $q = q_j$.
(3) Use the recursions for $1 \leq k \leq n$ to determine b_k and c_k :

$$\begin{aligned} b_k &= a_k - pb_{k-1} - qb_{k-2} \\ c_k &= b_k - pc_{k-1} - qc_{k-2}, \end{aligned}$$

and also compute $\overline{c_{n-1}} = c_{n-1} - b_{n-1}$
(4) Solve for Δp and Δq using the equations

$$\begin{aligned} c_{n-2}\Delta p + c_{n-3}\Delta q &= b_{n-1} \\ \overline{c_{n-1}}\Delta p + c_{n-2}\Delta q &= b_n \end{aligned}$$

(we could write the exact formulas.)
(5) Let $p_{j+1} = p_j + \Delta p$ and $q_{j+1} = q_j + \Delta q$ and $j = j + 1$.
(6) If $\Delta p^2 + \Delta q^2 > \varepsilon$ go back to step (2).
Otherwise, stop, and $x^2 + p_j x + q_j$ is the quadratic factor.

The convergence in this method is quadratic for most well-conditioned polynomials.

5. Numerical Regression

5.1. **One-variable linear regression.** Suppose that we are given a set of ordered pairs $\{(x_i, y_i)\}_{1 \leq i \leq N}$ we wish to find the equation of a line $y = mx + b$ that most closely fits the data. Or, more generally, suppose that you have k given functions $\{g_j(x)\}_{1 \leq j \leq k}$ and wish to find constants $a_1, ..., a_k$ such that the equation $y = \sum_{j=1}^{k} a_j g_j(x)$ most closely fits the data.

One problem that we have is finding a function to minimize. For instance, suppose that we wish to minimize the quantity $\sum_{i=1}^{N} \left| y_i - \sum_{j=1}^{k} a_j g_j(x_i) \right|$. Or to minimize the sum of the distances between the curve $y = \sum_{j=1}^{k} a_j g_j(x)$ and the points (x_i, y_i). Unfortunately, neither of these minimization procedures will yield a unique solution $(a_1, ..., a_k)$ for the coefficients. For example, if the set of four points $\{(0,0), (1,1), (0.5,1), (1.5,0)\}$ is "fit" to a line of the form $y = mx + b$, the solution (m, b) that most closely "fits" the data is $m = 0$ and $0 \leq b \leq 1$. In other words, any line of the form $y = f(x) = b$ with $0 \leq b \leq 1$ yields the

minimum possible value of $|0-f(0)|+|1-f(1)|+|1-f(0.5)|+|0-f(1.5)|$ (which is 2, by the way).

The **least squares regression method** provides a function that always has a unique minimum under reasonable assumptions. The **objective function** that we try to minimize is $F(a_1,...,a_k)=\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{k}a_jg_j(x_i)\right)^2$ — hence the name "least squares."

How do we solve for the "best" constants $a_1,...,a_k$. The idea is to think of $F(a_1,...,a_k)$ as the square of the length of a vector in N-dimensional space. The vectors $X=(x_1,...,x_N)$ and $Y=(y_1,...,y_N)$ are in the space, and also the vectors $G_1=(g_1(x_1),...,g_1(x_N))$, $G_2=(g_2(x_1),...,g_2(x_N))$, ..., $G_k=(g_k(x_1),...,g(x_N))$ are k vectors in N-dimensional space. Each $\sum_{j=1}^{k}a_jg_j(x_i)$ is a component of the vector $\sum_{j=1}^{k}a_jG_j$, another vector in N-dimensional space that is a linear combination of $G_1,...,G_k$. Then the objective function $F(a_1,...,a_k)=\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{k}a_jg_j(x_i)\right)^2$ is the square of the length of the vector $Y-\sum_{j=1}^{k}a_jG_j$; that is, $F(a_1,...,a_k)=\left\|Y-\sum_{j=1}^{k}a_jG_j\right\|^2$. We must find the values of the constants $a_1,...,a_k$ that make this quantity a minimum. Now, the set of all vectors $\sum_{j=1}^{k}a_jG_j$ is a k-dimensional plane inside N-dimensional space. The optimal value of the constants $a_1,...,a_k$ will correspond to the point of the k-plane that is closest to the point Y. This point is uniquely determined and is the projection of the vector Y onto the k-plane spanned by $G_1,...,G_k$.

For convenience, we may write $\sum_{j=1}^{k}a_jG_j$ as a matrix multiplication:

$$\begin{aligned}\sum_{j=1}^{k}a_jG_j &= \sum_{j=1}^{k}a_j\begin{pmatrix}G_j(x_1)\\G_j(x_2)\\\vdots\\G_j(x_N)\end{pmatrix}\\ &= \begin{pmatrix}G_1(x_1)&G_2(x_1)&...&G_k(x_1)\\G_1(x_2)&G_2(x_2)&...&G_k(x_2)\\\vdots&\vdots&\ddots&\vdots\\G_1(x_N)&G_2(x_N)&...&G_k(x_N)\end{pmatrix}\begin{pmatrix}a_1\\a_2\\\vdots\\a_k\end{pmatrix}\end{aligned}$$

Let

$$G=\begin{pmatrix}G_1(x_1)&G_2(x_1)&...&G_k(x_1)\\G_1(x_2)&G_2(x_2)&...&G_k(x_2)\\\vdots&\vdots&\ddots&\vdots\\G_1(x_N)&G_2(x_N)&...&G_k(x_N)\end{pmatrix},\ a=\begin{pmatrix}a_1\\a_2\\\vdots\\a_k\end{pmatrix},$$

so

$$F(a_1,...,a_k)=F(a)=\|Y-Ga\|^2.$$

Next, how do we find the point $(a_1,...,a_k)$ corresponding to the point closest to Y in the k-plane? We know that $F(a_1,...,a_k)=\left\|Y-\sum_{j=1}^{k}a_jG_j\right\|^2$ is a real-valued function of k real variables, and it is a paraboloid that has a unique absolute minimum. So this function has a single critical point at which it is minimized. We find this by setting the gradient to zero

(i.e. set each $\frac{\partial F}{\partial a_j} = 0$). We have

$$\begin{aligned} F(a_1, ..., a_k) &= \|Y - Ga\|^2 \\ &= \sum_{s=1}^{N} \left((Y - Ga)_s \right)^2 \\ &= \sum_{s=1}^{N} \left(Y_s - \sum_{p=1}^{k} G_{sp} a_p \right)^2 \end{aligned}$$

$$\begin{aligned} 0 &= \frac{\partial F}{\partial a_j}(a) = \sum_{s=1}^{N} 2 \left(Y_s - \sum_{p=1}^{k} G_{sp} a_p \right) (-G_{sj}) \\ &= -2 \sum_{s=1}^{N} \left(G_{sj} Y_s - \sum_{p=1}^{k} G_{sj} G_{sp} a_p \right) \\ &= -2 \sum_{s=1}^{N} G_{sj} Y_s + 2 \sum_{p=1}^{k} \sum_{s=1}^{N} G_{sj} G_{sp} a_p \\ &= -2 \sum_{s=1}^{N} G_{js}^T Y_s + 2 \sum_{p=1}^{k} \sum_{s=1}^{N} G_{js}^T G_{sp} a_p \\ &= -2 \left(G^T Y \right)_j + 2 \left(G^T G a \right)_j = -2 \left(G^T Y - G^T G a \right)_j \end{aligned}$$

Thus is true for each j, so we have the matrix equation

$$G^T G a = G^T Y$$

As long as $G^T G$ is invertible (assumption!), we may solve

$$a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} = \left(G^T G \right)^{-1} G^T Y.$$

Recall that $G_{ij} = g_j(x_i)$. This gives the formula for the coefficients of the regression equation.

5.2. **Ordinary linear regression.** For the particular example of **ordinary linear regression**, $g_1(x) = 1$ and $g_2(x) = x$. Then the matrix G is

$$G = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix},$$

So

$$G^T G = \begin{pmatrix} 1 & 1 & ... & 1 \\ x_1 & x_2 & ... & x_N \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} = \begin{pmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}$$

Then

$$\begin{aligned}\left(G^TG\right)^{-1} &= \begin{pmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}^{-1} \\ &= \frac{1}{N\sum x_i^2 - \left(\sum x_i\right)^2}\begin{pmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & N \end{pmatrix},\end{aligned}$$

So

$$\begin{aligned}\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} &= \left(G^TG\right)^{-1}G^TY \\ &= \frac{1}{N\sum x_i^2 - \left(\sum x_i\right)^2}\begin{pmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & N \end{pmatrix}\begin{pmatrix} 1 & 1 & ... & 1 \\ x_1 & x_2 & ... & x_N \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \\ &= \frac{1}{N\sum x_i^2 - \left(\sum x_i\right)^2}\begin{pmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & N \end{pmatrix}\begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix} \\ &= \frac{1}{N\sum x_i^2 - \left(\sum x_i\right)^2}\begin{pmatrix} \sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i \\ -\sum x_i \sum y_i + N\sum x_i y_i \end{pmatrix}\end{aligned}$$

Thus, the equations for the linear function that provides the least squares fit of the data is

$$y = a_1 + a_2 x,$$

where

$$\begin{aligned} a_1 &= \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{N\sum x_i^2 - \left(\sum x_i\right)^2} \\ a_2 &= \frac{-\sum x_i \sum y_i + N\sum x_i y_i}{N\sum x_i^2 - \left(\sum x_i\right)^2}. \end{aligned}$$

5.3. **Polynomial regression.** Another particular example occurs when we try to fit a polynomial of degree d to the data. Then $k = d + 1$ and

$$g_1(x) = 1,\ g_2(x) = x,\ g_3(x) = x^2, ..., g_{d+1}(x) = x^d.$$

Then the matrix G is

$$G = \begin{pmatrix} 1 & x_1 & x_1^2 & ... & x_1^d \\ 1 & x_2 & x_2^2 & ... & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & ... & x_N^d \end{pmatrix},$$

So

$$G^TG = \begin{pmatrix} 1 & 1 & ... & 1 \\ x_1 & x_2 & ... & x_N \\ x_1 & x_2 & ... & x_N \\ \vdots & \vdots & \ddots & \vdots \\ x_1^d & x_2^d & ... & x_N^d \end{pmatrix} \begin{pmatrix} 1 & x_1 & x_1^2 & ... & x_1^d \\ 1 & x_2 & x_2^2 & ... & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & ... & x_N^d \end{pmatrix}$$

$$= \begin{pmatrix} N & \sum x_i & \sum x_i^2 & ... & \sum x_i^d \\ \sum x_i & \sum x_i^2 & \ddots & \ddots & \sum x_i^{d+1} \\ \sum x_i^2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sum x_i^{2d-2} & \sum x_i^{2d-1} \\ \sum x_i^d & \sum x_i^{d+1} & ... & \sum x_i^{2d-1} & \sum x_i^{2d} \end{pmatrix}$$

$$G^TY = \begin{pmatrix} 1 & 1 & ... & 1 \\ x_1 & x_2 & ... & x_N \\ x_1 & x_2 & ... & x_N \\ \vdots & \vdots & \ddots & \vdots \\ x_1^d & x_2^d & ... & x_N^d \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^d y_i \end{pmatrix}$$

So

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{d+1} \end{pmatrix} = \left(G^TG\right)^{-1} G^TY$$

$$= \begin{pmatrix} N & \sum x_i & \sum x_i^2 & ... & \sum x_i^d \\ \sum x_i & \sum x_i^2 & \ddots & \ddots & \sum x_i^{d+1} \\ \sum x_i^2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sum x_i^{2d-2} & \sum x_i^{2d-1} \\ \sum x_i^d & \sum x_i^{d+1} & ... & \sum x_i^{2d-1} & \sum x_i^{2d} \end{pmatrix}^{-1} \begin{pmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^d y_i \end{pmatrix}.$$

By solving the equations above, we get the equations for the polynomial function that provides the least squares fit of the data:

$$y = a_1 + a_2 x + ... + a_{d+1} x^d.$$

An interesting question comes up: how do we know that the matrix G^TG is invertible? Consider the upper $(d+1) \times (d+1)$ square part G^{upper} of the matrix G:

$$V = G^{\text{upper}} = \begin{pmatrix} 1 & x_1 & x_1^2 & ... & x_1^d \\ 1 & x_2 & x_2^2 & ... & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{d+1} & x_{d+1}^2 & ... & x_{d+1}^d \end{pmatrix}.$$

This is the famous Vandermonde matrix V, which has determinant

$$\det V = \prod_{1 \le i < j \le d+1} (x_j - x_i),$$

which is nonzero as long as the independent data values are distinct. Thus the range of the transformation associated to G followed by the project to the firs $d+1$ dimensions contains all of $\mathbb{R}^{d+1}$. Then G^T maps this upper part to all of $\mathbb{R}^{d+1}$. Therefore, the range of the transformation associated to the $(d+1) \times (d+1)$ matrix $G^T G$ is all of $\mathbb{R}^{d+1}$, and thus in this case, $G^T G$ is always invertible.

5.4. **Correlation coefficient.** We would like to have a measurement of how closely the data agrees with the regression equation. The **correlation coefficient** r gives us a real number between 0 and 1 that provides us with such a measurement. With notation as above, recall the the vector Y of data y-values is projected to the k-dimensional plane spanned by the vectors G_1, ..., G_k in order to get the optimal coefficients $a_1, ..., a_k$ such that y_i is close to $\sum_{j=1}^{k} a_j g_j(x_i)$. If Y is actually contained in this plane, then the data has a perfect fit with the regression equation. If θ is the angle between the Y-vector and the plane spanned by G_1, ..., G_k, then the correlation coefficient might be defined to be

$$r = \cos\theta.$$

Also,

$$r^2 = \cos^2\theta = 1 - \sin^2\theta$$

is known as the **coefficient of determination**, which is the same as

$$r^2 = 1 - \frac{\|Y - Ga\|^2}{\|Y\|^2}$$

This seems to agree with the literature only if the average value of the y_j is zero. The reason is that the data is first **centered** (i.e. x_j is replaced by $x_j - \overline{x}$ and y_j is replaced by $y_j - \overline{y}$). The typical formula for r in ordinary linear regression is

$$r = \frac{(Y - \overline{y}1) \cdot (X - \overline{x}1)}{\|Y - \overline{y}1\| \, \|X - \overline{x}1\|},$$

where $1 = (1, 1, 1, ..., 1) \in \mathbb{R}^N$.

Since $\cos\theta = \frac{v \cdot w}{\|v\|\|w\|}$ for two vectors v and w separated by an angle θ, we have

$$r = \frac{Y \cdot (Ga)}{\|Y\| \, \|Ga\|},$$

where

$$a = \left(G^T G\right)^{-1} G^T Y.$$

Thus,

$$r = \frac{Y \cdot \left(G \left(G^T G\right)^{-1} G^T Y\right)}{\|Y\| \left\|G \left(G^T G\right)^{-1} G^T Y\right\|}$$

Index