1 Solving equations by iteration

Suppose we want to solve

$$f(x) = 0. \tag{1}$$

There are no general formulas for this (except in a few simple cases). Thus, any solution depends almost entirely on numerical methods.

Here, f is a given function. A solution of (1) is a number $x = s$ such that $f(s) = 0$.

Examples are $x^2 - 3x + 2 = 0$ or $x^3 + x = 1$. These are called algebraic equations because the corresponding f is a polynomial. Solutions are called roots of the equation.

Other examples are $\sin x = 0.5x$, $\tan x = x$, $\cosh x = \sec x$ or $\cosh x \cos x = -1$. All of these can be written in the form (1). These are called transcendental equations because they involve transcendental functions. There are no general solutions to such equations.

1.1 Iteration methods

We can't solve equation (1) exactly, but we can find approximate solutions numerically. The basic idea is to guess an initial value (which may be a very poor guess) and then use that to refine a second approximation. The second approximation leads to a third, and so on. The idea is that each approximation gets closer and closer to the exact solution.

Iteration methods are generally easy to program because the computational operations are the same in each step. Only the data change from one step to the next. Furthermore, if a method converges, it is usually stable.

1.2 Fixed-point iteration

We transform (1) algebraically into the form

$$x = g(x). \tag{2}$$

Then we choose an x_0 and compute $x_1 = g(x_0)$, $x_2 = g(x_1)$ and so on. In general,

$$x_{n+1} = g(x_n) \qquad \text{for } n = 0, 1, 2, \ldots \tag{3}$$

A solution of (2) is called a fixed point of g. This is also a solution of (1), since we can algebraically return to the original form $f(x) = 0$.

Note: There may of course be many forms of (2) that satisfy (1). The behaviour of the corresponding iterative sequences $x_0, x_1, \ldots$ may differ accordingly. In particular, some sequences may converge faster than others.

Example.

$$\begin{aligned} f(x) &= 3x^2 + 2x - 1 = 0 \\ x &= \frac{1}{2} - \frac{3}{2}x^2 \\ x_n &= \frac{1}{2} - \frac{3}{2}x_{n-1}^2 \end{aligned}$$

Another possibility is

$$\begin{aligned} 3x^2 &= 1 - 2x \\ x_n &= \pm\sqrt{\frac{1}{3} - \frac{2}{3}x_{n-1}} \end{aligned}$$

although this has issues of definition, since the part under the square root can be negative and we have to deal with the plus or minus. But in theory this formulation will also converge to the same fixed point.

An iteration process defined by (3) is called convergent for an x_0 if the corresponding sequence $x_0, x_1, \ldots$ is convergent.

Theorem 1.1 (Convergence of fixed-point iteration)**.** *Let $x = s$ be a solution of $x = g(x)$ and suppose that g has a continuous derivative in some interval J containing s. Then if $|g'(x)| \le K < 1$ in J, the iteration process defined by* (3) *converges for any x_0 in J.*

Proof. By the mean value theorem of differential calculus, there is a t between x and s such that

$$g(x) - g(s) = g'(t)(x - s) \qquad \text{for } x \text{ in } J.$$

Since $g(s) = s$ and $x_1 = g(x_0), x_2 = g(x_1), \ldots$, we obtain from this that

$$\begin{aligned} |x_n - s| &= |g(x_{n-1}) - g(s)| \\ &= |g'(t)||x_{n-1} - s| \\ &\le K|x_{n-1} - s| \\ &= K|g(x_{n-2}) - g(s)| \\ &= K|g'(\tilde{t})||x_{n-2} - s| \\ &\le K^2|x_{n-2} - s| \\ &\vdots \\ &\le K^n|x_0 - s|. \end{aligned}$$

Since $K < 1$, we have $K^n \to 0$ and hence $|x_n - s| \to 0$ as $n \to \infty$.

□

A function g satisfying the condition in Theorem 1.1 is called a contraction because $|g(u) - g(v)| \le K|u - v|$ with $K < 1$.

Furthermore, K gives information on the speed of convergence. For example, if $K = 0.5$, then the accuracy increases by at least two digits in only seven steps because $0.5^7 < 0.01$.

1.3 Cobwebbing

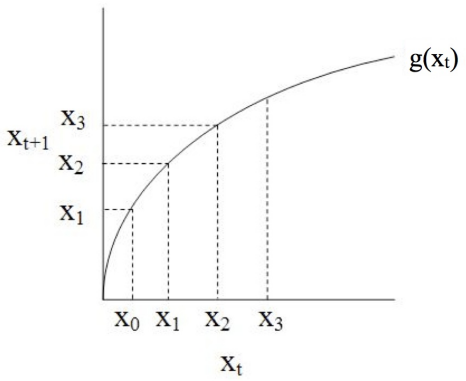
Suppose we have a discrete time dynamical system $x_{t+1} = g(x_t)$.

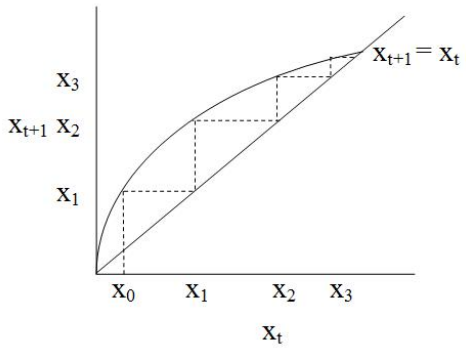
Graphically, we'd like to find out what happens if we start at x_0. This involves measuring each x_{t+1} and then plugging that back in for the next x_t.

Better way: Draw the line $x_{t+1} = x_t$ and then bounce off that. This line restarts each step, turning x_{t+1} into the next x_t (since $x_{t+1} = x_t$ on this line).

Using this method, we get
a) the steps and
b) a sense of the long-term behaviour.

This method is called cobwebbing.

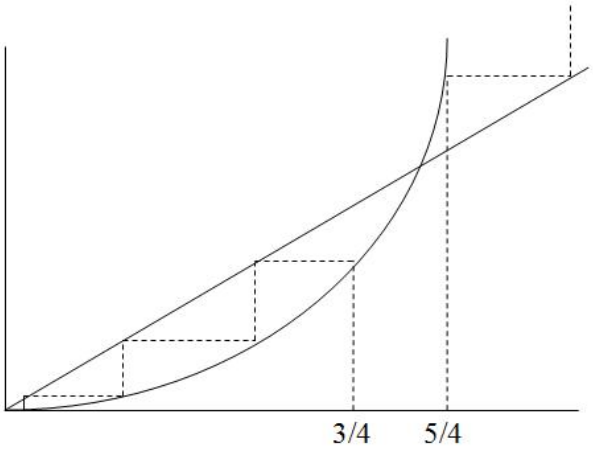




Example. Cobweb $x_{t+1} = x_t^2$ with the starting points $x_0 = \frac{3}{4}$, $x_0 = \frac{5}{4}$. Where do solutions go eventually?

For $x_0 = \frac{3}{4}$, solutions approach 0.

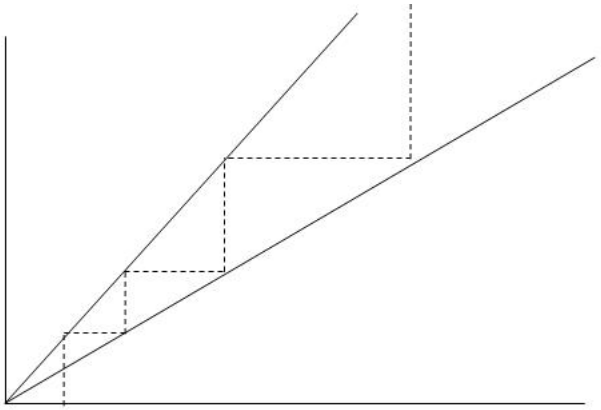
For $x_0 = \frac{5}{4}$, solutions approach ∞.



In this case, $|g'(x)| < 1$ in the region $[0, 1/4]$ so the process converges around the origin. Conversely, $|g'(x)| > 1$ in the region $[3/4, 5/4]$ so the process does not converge.

Example. Cobweb $a_{t+1} = 3a_t$ with $x_0 = 1$.

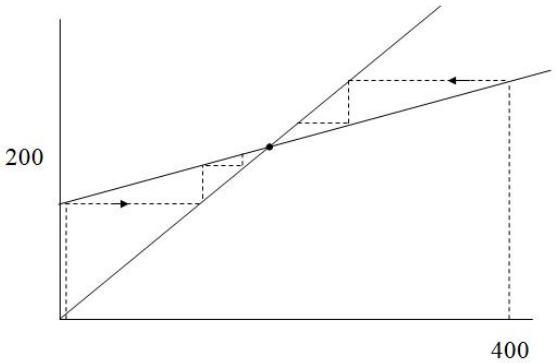
Solutions move away from 0 and approach ∞.



In this case, $|g'(x)| = 3 > 1$ for all x, so the process does not converge.

Example. $x_{t+1} = \frac{1}{2}x_t + 100,\ x_0 = 0,\ x_0 = 400$

Solutions approach 200.

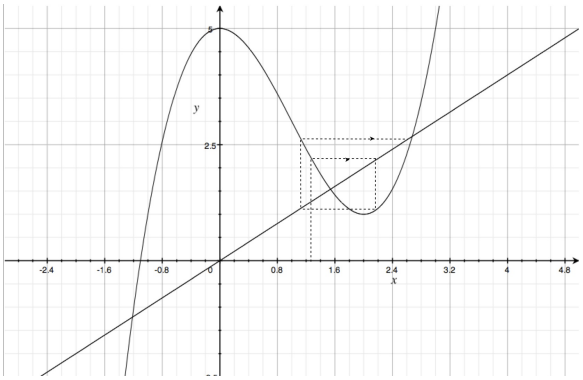


In this case $|g'(x)| = 0.5 < 1$, so the process always converges.

Example. $y_{t+1} = y_t^3 - 3y_t^2 + 5$

$x_0 = \frac{1}{4}$
$x_1 = 4.828125$
$x_2 = 47.615$
$x_3 = 101155.86$

$x_0 = \frac{3}{2}$
$x_1 = 1.625$
$x_2 = 1.369$
$x_3 = 1.943$
$x_4 = 1.0096$
$x_5 = 2.97119$



Very complex behaviour can be understood fairly easily. Note however that if $|g'(x)| = 1$ then we have no information.

2 Newton's Method

Newton's method is a procedure for finding approximate solutions to equations that can't be solved. We want to solve $f(x) = 0$. Let x^* be the exact solution, satisfying $f(x^*) = 0$. Let x_0 be some initial approximation of x^*. We want to find x_1, another approximation, which is closer to x^*.

The linear approximation $\hat{f}(x)$ of $f(x)$ near x_0 is

$$\hat{f}(x) = f(x_0) + f'(x_0)(x - x_0)$$

$\hat{f}(x)$ is a simple linear function, so we can solve $\hat{f}(x) = 0$. Then

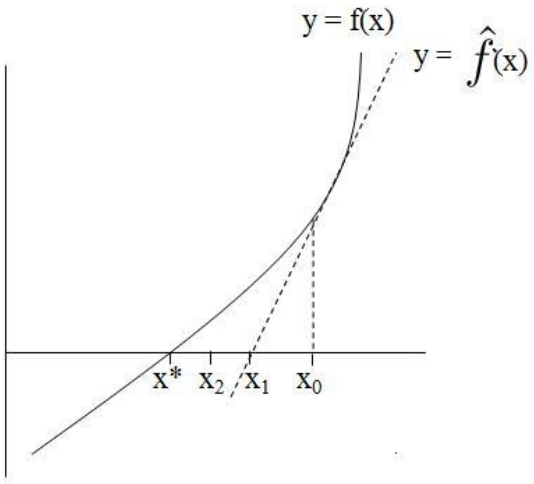
$$f(x_0) + f'(x_0)(x - x_0) = 0$$
$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

The solution of this equation is the new approximation x_1:

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

We can repeat this process to find $x_2, x_3, x_4, \ldots$ successive approximations of the exact solution, each of which is closer to x^* than the one before.

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}, \qquad x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}, \qquad x_4 = \ldots$$



Thus, the general formula for Newton's method is

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

Example. Find $\sqrt{5}$ accurate to 5 decimal places.

To do this, we solve $x^2 - 5 = 0$.

Set $f(x) = x^2 - 5$
Then $f'(x) = 2x$

What to choose for x_0? We know that $\sqrt{4} = 2$ and $\sqrt{5}$ is close to $\sqrt{4}$ so let's try $x_0 = 2$.

$$x_0 = 2 \qquad \text{1 d.p}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = x_0 - \frac{x_0^2 - 5}{2x_0} = \frac{x_0}{2} + \frac{2.5}{x_0}$$

$$x_1 = \frac{2}{2} + \frac{2.5}{2} = 2.25 \qquad \text{2 d.p}$$

$$x_2 = \frac{x_1}{2} + \frac{2.5}{x_1} = \frac{2.25}{2} + \frac{2.5}{2.25} = 2.23611$$

Question: Is this enough?

$$x_3 = \frac{x_2}{2} + \frac{2.5}{x_2} = 2.236067977915$$

Therefore, 3 decimal places must be correct.

$$x_4 = \frac{x_3}{2} + \frac{2.5}{x_3} = 2.236067977$$

Therefore, we have 9 decimal places of accuracy after 3 steps.
When Newton's method works, it converges to the exact solution very fast.

Newton's method is a DTDS $x_{t+1} = x_t - \dfrac{f(x_t)}{f'(x_t)}$

The solution $x^*$is an equilibrium: $x^* = x^* - \dfrac{f(x^*)}{f'(x^*)}$

$0 = \dfrac{f(x^*)}{f'(x^*)}$ But $f(x^*) = 0$ so we have $0 = 0$... so long as $f'(x^*) \neq 0$

Stability of x^*? Updating function is $h(x) = x - \dfrac{f(x)}{f'(x)}$

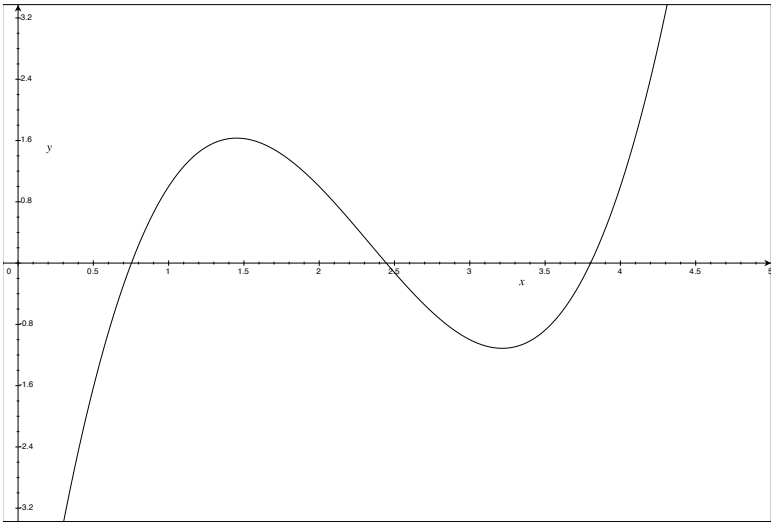
$$h'(x) = 1 - \frac{f'(x)f'(x) - f(x) \cdot f''(x)}{[f'(x)]^2} = \frac{[f'(x)]^2 - [f'(x)]^2 + f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f(x)]^2}$$

At equilibrium, we have $h'(x^*) = \dfrac{f(x^*)f''(x^*)}{[f'(x^*)]^2} = 0$ since $f(x^*) = 0$ and $f'(x^*) \neq 0$.

Since $|h'(x^*)| < 1$, this equilibrium is stable.

In fact, the smaller $|h'(x^*)|$, the faster the convergence. Since $|h'(x^*)| = 0$ for Newton's method, convergence is as fast as possible and we say that the equilibrium is superstable. But, it is only locally stable. So the initial point needs to be close to x^* or else it may fail.

Example. The function $f(x) = (x-2)^3 - (x-1)^2 + 2$ has a root near $x = 4$. What happens if you use Newton's method starting at $x = 2$ versus $x = 1.57$?



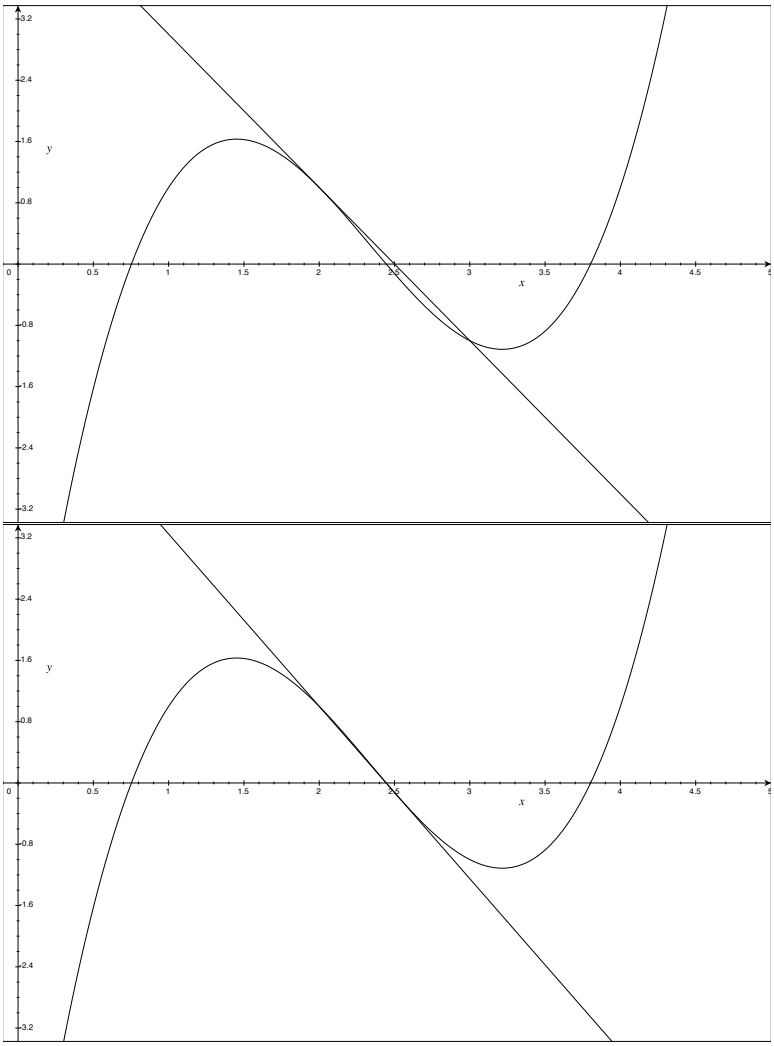
$$f(x) = (x-2)^3 - (x-1)^2 + 2$$
$$f'(x) = 3(x-2)^2 - 2(x-1)$$

$x_0 = 2 \qquad f(2) = 1 \qquad f'(2) = -2$

$x_1 = 2 - \dfrac{1}{-2} = 2.5$

$x_2 = 2.444$

$x_3 = 2.445$



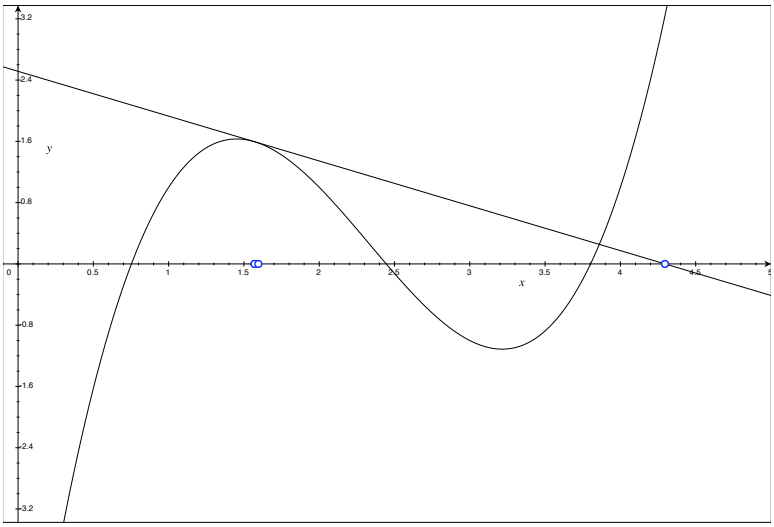
$x_0 = 1.57$

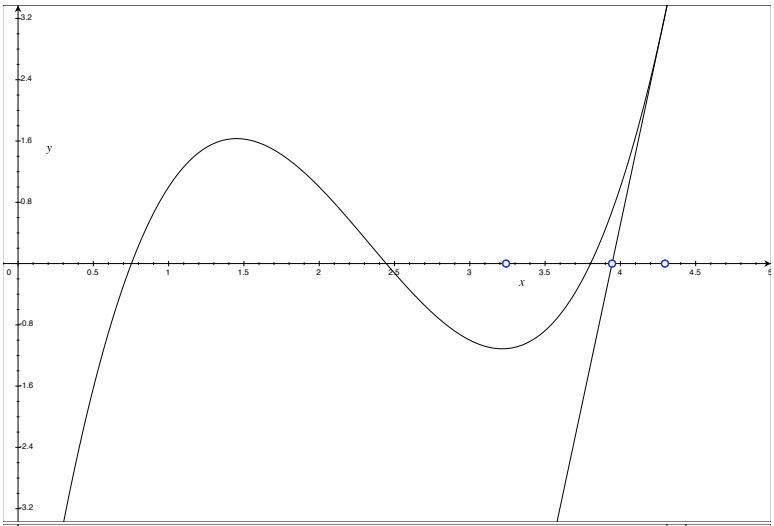
$x_1 = 4.2961$

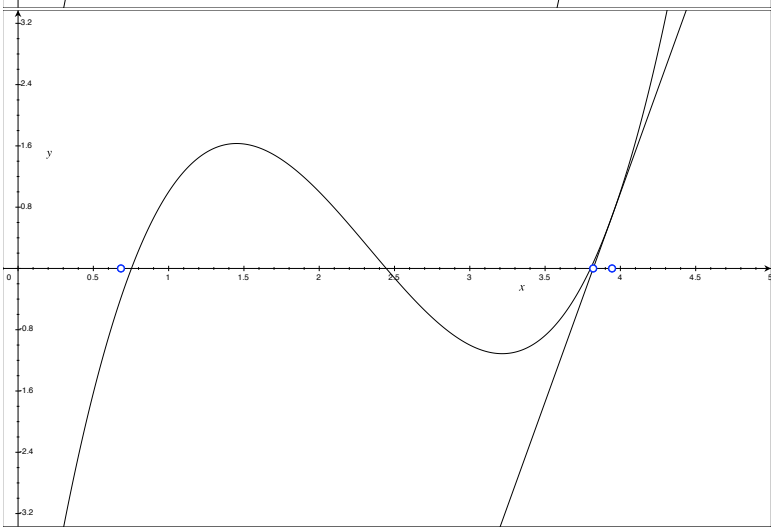
$x_2 = 3.9447$ (4)

$x_3 = 3.8195$

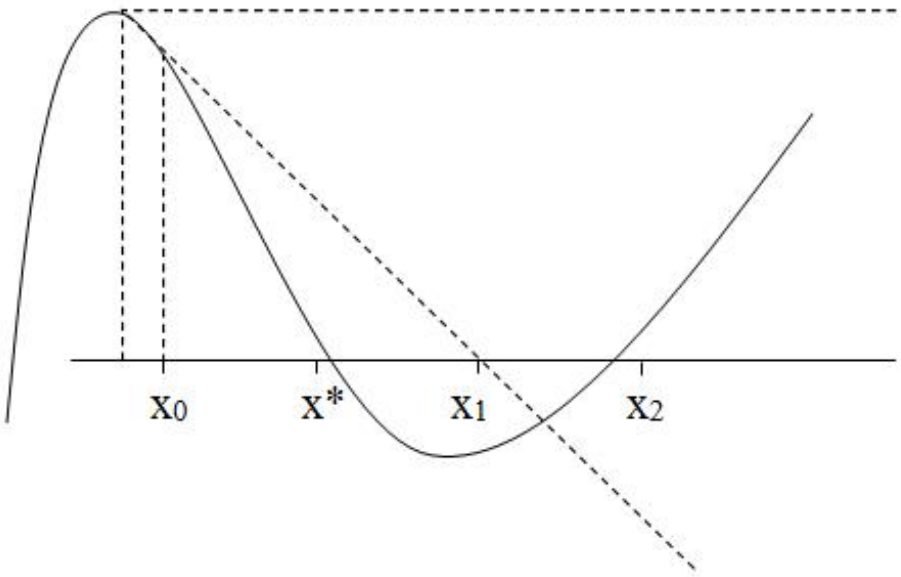
$x_4 = 3.8023$







If $f(x)$ has several roots close to each other, then Newton's method may converge to a different root, not the desired one. If $f(x)$ has a local min or max near the solution x^*, then we need to choose x_0 close to x^*. Otherwise, the next approximation may shoot off to infinity since at points of min/max $f' = 0$ and we are dividing by f'.



3 The Secant Method

We obtain the secant method from Newton's method if we replace the derivative $f'(x)$ with the difference quotient

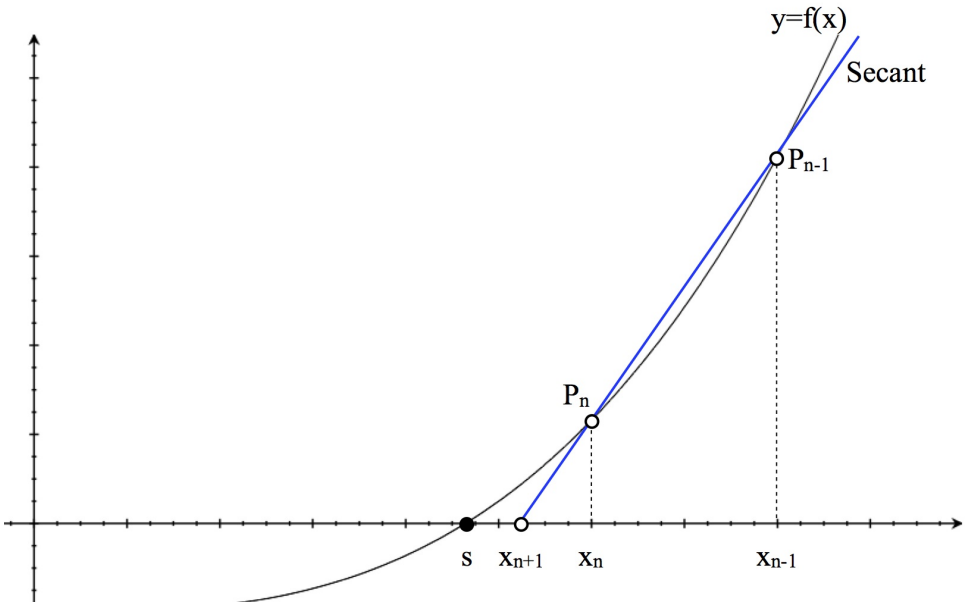
$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

Then instead of requiring a derivative in the denominator, we have

$$\boxed{x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}.} \tag{5}$$

Note that we need two starting values, x_0 and x_1.

Geometrically, we intersect the x-axis at x_{n+1} with the secant of $f(x)$ passing through P_{n-1} and P_n.



Example. Find the positive solution of $f(x) = x - 2\sin x = 0$ by the secant method, starting from $x_0 = 2$ and $x_1 = 1.9$.

Using (5), we have

$$x_{n+1} = x_n - \frac{(x_n - 2\sin x_n)(x_n - x_{n-1})}{x_n - x_{n-1} + 2(\sin x_{n-1} - \sin x_n)} = x_n - \frac{N_n}{D_n}$$

Numerical values are

n	x_{n-1}	x_n	N_n	D_n	$x_{n+1} - x_n$
1	2.000000	1.900000	-0.000740	-0.174005	-0.004253
2	1.900000	1.895747	-0.000002	-0.006986	-0.000252
3	1.895747	1.895747	0		0

$x_3 = 1.895747$ is accurate to six decimal places.

4 Interpolation

Suppose we have a finite number of points $x_0, x_1, \ldots x_n$ and some function $f(x)$ that satisfies

$$f(x_0) = f_0, \quad f(x_1) = f_1, \quad \cdots \quad f(x_n) = f_n$$

The general form of $f(x)$ may or may not be known. If known, it may have an explicit mathematical expression. If unknown, it may be estimated at the points $x_0, x_1, \ldots x_n$ from empirical data. We'd like to use the given information to estimate other values of x.

Interpolation involves finding approximate values of a function $f(x)$ for values of x that lie between $x_0, x_1, \ldots x_n$. Extrapolation involves estimating values of x that lie outside this range (ie $x < x_0$ or $x > x_n$).

We have $n+1$ known values, so one possibility is to find a polynomial $p_n(x)$ of degree n that fits the given values. Thus

$$p_n(x_0) = f_0, \quad p_n(x_1) = f_1, \quad \cdots \quad p_n(x_n) = f_n. \tag{6}$$

(In general, to fit a polynomial of degree n, we need $n+1$ points.)

We call $p_n(x)$ an interpolation polynomial and $x_0, x_1, \ldots x_n$ the nodes. If $f(x)$ is a mathematical function, we call $p_n(x)$ an polynomial approximation of f. Thus, we can use $p_n(x)$ to interpolate values of f for x's between x_0 and x_n or to extrapolate outside that interval.

Theorem 4.1. *A polynomial approximation satisfying* (6) *exists and is unique.*

Proof. We illustrate existence in subsequent sections.

Suppose a polynomial $q_n(x)$ satisfies

$$q_n(x_0) = f_0, \quad q_n(x_1) = f_1, \quad \cdots \quad q_n(x_n) = f_n.$$

Then $p_n(x) - q_n(x) = 0$ at $x_0, x_1, \ldots x_n$. However, a polynomial $p_n(x) - q_n(x)$ of degree n (or less) with $n+1$ roots must be identically zero. Thus $p_n(x) = q_n(x)$.

□

4.1 Lagrange interpolation

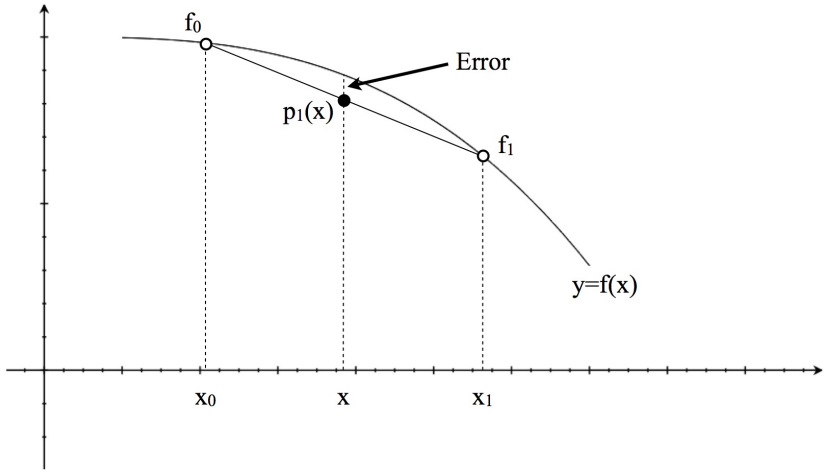
Lagrange interpolation involves multiplying each f_j by a polynomial that is

- 1 at x_j
- 0 at the other n nodes

and then taking the sum of these $n+1$ polynomials. This gives the unique interpolation polynomial of degree n or less.

4.2 Linear interpolation

The simplest interpolation is to draw a straight line through the points (x_0, f_0) and (x_1, f_1).



Thus the linear Lagrange polynomial p_1 is a sum

$$p_1 = L_0 f_0 + L_1 f_1$$

where L_0 is the linear polynomial that is 1 at x_0 and 0 at x_1. Similarly, L_1 is 1 at x_1 and 0 and x_0. We thus have

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} \qquad \text{and} \qquad L_1(x) = \frac{x - x_0}{x_1 - x_0}.$$

This gives the linear Lagrange polynomial

$$\begin{aligned} p_1(x) &= L_0(x) f_0 + L_1(x) f_1 \\ &= \frac{x - x_1}{x_0 - x_1} f_0 + \frac{x - x_0}{x_1 - x_0} f_1. \end{aligned}$$

The error is the difference between the estimated value and the actual value.

Example. If we know that $\ln 9.0 = 2.1972$ and $\ln 9.5 = 2.2513$, estimate $\ln 9.2$ using linear Lagrange interpolation and determine the error (to 4 decimal places).

We have $x_0 = 9.0$, $x_1 = 9.5$, $f_0 = 2.1972$ and $f_1 = 2.2513$. Thus, we have

$$L_0(9.2) = \frac{9.2 - 9.5}{9.0 - 9.5} = 0.6 \qquad \text{and} \qquad L_1(9.2) = \frac{9.2 - 9.0}{9.5 - 9.0} = 0.4.$$

Thus

$$\begin{aligned} p_1(9.2) &= L_0(9.2) f_0 + L_1(9.2) f_1 \\ &= 0.6(2.1972) + 0.4(2.2513) \\ &= 2.2188. \end{aligned}$$

Since $\ln 9.2 = 2.2192$, the error is $\epsilon = 2.2188 - 2.2192 = 0.0004$. Thus, the approximation is accurate to 3 decimal places.

4.3 Quadratic interpolation

The next level of approximation after linear is quadratic. We now need three points: (x_0, f_0), (x_1, f_1) and (x_2, f_2). This will lead us to a second-degree polynomial $p_2(x)$ given by

$$p_2(x) = L_0(x) f_0 + L_1(x) f_1 + L_2(x) f_2$$

where $L_0(x_0) = L_1(x_1) = L_2(x_2) = 1$ and $L_0(x_1) = L_0(x_2) = 0$ etc. We thus have

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}$$
$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$
$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

Example. If we know that $\ln 9.0 = 2.1972$, $\ln 9.5 = 2.2513$ and $\ln 11.0 = 2.3979$, estimate $\ln 9.2$ using quadratic Lagrange interpolation and determine the error (to 4 decimal places).

We have

$$L_0(x) = \frac{(x-9.5)(x-11.0)}{(9.0-9.5)(9.0-11.0)} = x^2 - 20.5x + 104.5 \qquad L_0(9.2) = 0.5400$$
$$L_1(x) = \frac{(x-9.0)(x-11.0)}{(9.5-9.0)(9.5-11.0)} = -\frac{1}{0.75}(x^2 - 20x + 99) \qquad L_1(9.2) = 0.4800$$
$$L_2(x) = \frac{(x-9.0)(x-9.5)}{(11.0-9.0)(11.0-9.5)} = \frac{1}{3}(x^2 - 18.5x + 85.5) \qquad L_2(9.2) = -0.0200.$$

Thus

$$\begin{aligned} p_2(9.2) &= L_0(9.2)f_0 + L_1(9.2)f_1 + L_2(9.2)f_2 \\ &= 0.5400(2.1972) + 0.4800(2.2513) - 0.0200(2.3979) \\ &= 2.2192 \end{aligned}$$

In this case, the error is $\epsilon = 2.2192 - 2.2192 = 0$, so the quadratic approximation is exact to 4 decimal places.

Example. Use the same data to estimate $\ln 8.0$ and determine the error (to 4 decimal places).

From the above, we have

$$L_0(8.0) = 4.5$$
$$L_1(8.0) = -4$$
$$L_2(8.0) = 0.5$$

Thus

$$\begin{aligned} p_2(8.0) &= L_0(8.0)f_0 + L_1(8.0)f_1 + L_2(8.0)f_2 \\ &= 4.5(2.1972) - 4(2.2513) + 0.5(2.3979) \\ &= 2.0811 \end{aligned}$$

The error is $\epsilon = 2.0811 - 2.0794 = 0.0017$. Thus, this quadratic approximation is only accurate to 2 decimal places.

We see here that extrapolation is much less accurate than interpolation.

4.4 General Lagrange interpolation polynomial

For general n, we have

$$\boxed{f(x) \approx p_n(x) = \sum_{k=0}^{n} L_k(x)f_k = \sum_{k=0}^{n} \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} f_k} \qquad (7)$$

where $L_k(x_k) = 1$ and $L_k(x_j) = 0$ for $j \neq k$.

Note that $p_n(x_k) = f_k$ since the numerator in (7) is zero for all $j \neq k$ and is equal to the denominator when $x = x_k$.

4.5 Error estimate

If $f(x)$ is itself a polynomial of degree n (or less), it must coincide with $p_n(x)$ because the $n+1$ points $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$ determine a unique polynomial. In this case, the error is zero and the $(n+1)$st derivative is zero.

We use this idea to determine the error for a general function: ie that the $(n+1)$st derivative of f should determine the error. It turns out that this is true if $f^{(n+1)}$ exists and is continuous.

The error is thus

$$\boxed{\epsilon_n(x) = f(x) - p_n(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(t)}{(n+1)!},} \tag{8}$$

where t is some value between x_0 and x_n if we interpolate (or between x_0, x and x_n if we extrapolate).

Thus $\epsilon_n(x) = 0$ at the nodes. We can also find error bounds if we take the max and min of $f^{(n+1)}$ on the interval $x_0 \le t \le x_n$ if we interpolate (or the interval also containing x if we extrapolate).

Example. Use (8) to estimate the error for the linear Lagrange approximation. Explain any discrepancy.

We have $n = 1$, so we need the second derivative of $f(t) = \ln t$. Since $f'(t) = 1/t$, $f''(t) = -1/t^2$ and thus

$$\epsilon_1(x) = (x - 9.0)(x - 9.5)\left(-\frac{1}{2t^2}\right)$$
$$\epsilon_1(9.2) = \frac{0.03}{t^2}$$

Since $f(t)$ is always increasing (ie monotone), the extreme values can only occur at the endpoints $t = 9.0$ and $t = 9.5$. We have

$$\epsilon_1(9.0) = 0.03/9^2 = 0.00037037$$
$$\epsilon_1(9.5) = 0.03/9.5^2 = 0.0003324.$$

So the maximum error should be less than 0.00038. But we found the error was 0.0004 in the linear Lagrange approximation. What's going on here?

The answer is round-off. Using 5 decimal places, we have

$$p_1(9.2) = 0.6(2.19722) + 0.4(2.25129) = 2.21885$$

with an error of $\epsilon = 2.21920 - 2.21885 = 0.00035$.

Thus, the discrepancy was caused by round-off, which was not taken into account in (8).

5 Newton's divided difference interpolation

Lagrange interpolation needs a polynomial of a specific degree to get the required accuracy. If the accuracy needs change, we would need a whole new polynomial. With Newton's interpolation, we can use our existing work and simply add in another term to increase the accuracy.

Let $p_{n-1}(x)$ be the $(n-1)$st Newton polynomial (to be determined). Thus

$$p_{n-1}(x_0) = f_0 \qquad p_{n-1}(x_1) = f_1 \qquad \cdots \qquad p_{n-1}(x_{n-1}) = f_{n-1}.$$

We then write the nth Newton polynomial as

$$p_n(x) = p_{n-1}(x) + g_n(x)$$

where $g_n(x)$ is an nth order polynomial.

We thus have

$$g_n(x) = p_n(x) - p_{n-1}(x)$$

and the requirement that

$$p_n(x_0) = f_0 \qquad p_n(x_1) = f_1 \qquad \cdots \qquad p_n(x_{n-1}) = f_{n-1} \qquad p_n(x_n) = f_n.$$

Notice that

$$\begin{aligned} p_{n-1}(x_0) &= p_n(x_0) \\ p_{n-1}(x_1) &= p_n(x_1) \\ &\vdots \\ p_{n-1}(x_{n-1}) &= p_n(x_{n-1}) \end{aligned}$$

It follows that

$$g(x_0) = g(x_1) = \cdots = g(n_{n-1}) = 0.$$

In other words, g already has n roots and is an nth order polynomial, so it must be of the form

$$g_n(x) = a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}).$$

We thus know everything about g except for the constant a_n. To determine this, we set $x = x_n$.

$$\begin{aligned} g_n(x_n) &= p_n(x_n) - p_{n-1}(x_n) \\ a_n(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1}) &= f_n - p_{n-1}(x_n) \\ a_n &= \frac{f_n - p_{n-1}(x_n)}{(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})} \end{aligned}$$

We will define the a_i's recursively. The real issue is finding $p_{n-1}(x_n)$, for which we do not have a formula.

If $n = 1$, then $p_{n-1}(x_{n-1}) = p_0(x_0) = f_0$. But p_0 is a 0th order polynomial, which is a constant, so $p_0(x) = f_0$ for all x. Thus $p_0(x_1) = f_0$.

We then have

$$\begin{aligned} a_1 &= \frac{f_1 - p_0(x_1)}{(x_1 - x_0)} \\ &= \frac{f_1 - f_0}{x_1 - x_0} \equiv f[x_0, x_1]. \end{aligned}$$

Next, we have

$$\begin{aligned} p_1(x) &= p_0(x) + g(x) \\ &= f_0 + a_1(x - x_0) \\ &= f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0) \\ &= f_0 + (x - x_0)f[x_0, x_1] \end{aligned}$$

Using this, we have, for $n = 2$,

$$\begin{aligned}
a_2 &= \frac{f_2 - p_1(x_2)}{(x_2 - x_0)(x_2 - x_1)} \\
&= \frac{f_2 - f_0 - (x_2 - x_0) f[x_0, x_1]}{(x_2 - x_0)(x_2 - x_1)} \\
&= \frac{f_2 - f_0 - (x_2 - x_0)\frac{f_1 - f_0}{x_1 - x_0}}{(x_2 - x_0)(x_2 - x_1)} \\
&= \frac{1}{x_2 - x_0}\left[\frac{f_2}{x_2 - x_1} - f_1 \frac{x_2 - x_0}{(x_1 - x_0)(x_2 - x_1)} - \frac{f_0}{x_2 - x_1} + \frac{f_0(x_2 - x_0)}{(x_1 - x_0)(x_2 - x_1)}\right] \\
&= \frac{1}{x_2 - x_0}\left\{\frac{f_2}{x_2 - x_1} - f_1 \frac{x_2 - x_1 + x_1 - x_0}{(x_1 - x_0)(x_2 - x_1)} - \frac{f_0}{x_2 - x_1}\left[\frac{x_1 - x_0 - x_2 + x_0}{x_1 - x_0}\right]\right\} \\
&= \frac{1}{x_2 - x_0}\left[\frac{f_2}{x_2 - x_1} - \frac{f_1}{x_2 - x_1} - \frac{f_1}{x_1 - x_0} - \frac{f_0}{x_2 - x_1} \cdot \frac{(x_1 - x_2)}{(x_1 - x_0)}\right] \\
&= \frac{1}{x_2 - x_0}\left[\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1}{x_1 - x_0} + \frac{f_0}{x_1 - x_0}\right] \\
&= \frac{1}{x_2 - x_0}\left[\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1 - f_0}{x_1 - x_0}\right] \\
&= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \equiv f[x_0, x_1, x_2]
\end{aligned}$$

Hence

$$\begin{aligned}
p_2(x) &= p_1(x) + g_2(x) \\
&= f_0 + (x - x_0) f[x_0, x_1] + a_2(x - x_0)(x - x_1) \\
&= f_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2]
\end{aligned}$$

For $n = k$, we thus have

$$p_k(x) = p_{k-1}(x) + (x - x_0)(x - x_1)\cdots(x - x_{k-1}) f[x_0, \ldots, x_k],$$

where

$$\boxed{a_k = f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}.}$$

We thus have Newton's divided difference interpolation formula:

$$\boxed{f(x) \approx f_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1}) f[x_0, x_1, \ldots, x_n]}$$

Example. Consider the following measurements:

x_j	7	9	9.5	12
$f(x_j)$	1.945910	2.197225	2.251292	2.484907

Compute $f(10)$.

We have

$$\begin{aligned}
f[x_0, x_1] &= \frac{2.197225 - 1.945910}{9 - 7} = 0.125657 \\
f[x_1, x_2] &= \frac{2.251292 - 2.197225}{9.5 - 9} = 0.108134 \\
f[x_2, x_3] &= \frac{2.484907 - 2.251292}{12 - 9.5} = 0.093446
\end{aligned}$$

Thus, our table becomes

x_j	7		9		9.5		12
$f(x_j)$	1.945910		2.197225		2.251292		2.484907
$f[x_j, x_{j+1}]$		0.125657		0.108134		0.093446	

Next, we have

$$f[x_0, x_1, x_2] = \frac{0.108134 - 0.125657}{9.5 - 7} = -0.007009$$
$$f[x_1, x_2, x_3] = \frac{0.093446 - 0.108134}{12 - 9} = -0.004896$$

Our table is now

x_j	7	9	9.5	12
$f(x_j)$	1.945910	2.197225	2.251292	2.484907
$f[x_j, x_{j+1}]$	0.125657	0.108134	0.093446	
$f[x_j, x_{j+1}, x_{j+2}]$	-0.007009	-0.004896		

Finally, we have

$$f[x_0, x_1, x_2, x_3] = \frac{-0.004896 - (-0.007009)}{12 - 7} = 0.0004226$$

Thus our final table is

x_j	7	9	9.5	12
$f(x_j)$	1.945910	2.197225	2.251292	2.484907
$f[x_j, x_{j+1}]$	0.125657	0.108134	0.093446	
$f[x_j, x_{j+1}, x_{j+2}]$	−0.007009	-0.004896		
$f[x_j, x_{j+1}, x_{j+2}, x_{j+3}]$	0.0004226			

where the boxed numbers are the coefficients we need. We thus have

$$f(x) \approx p_3(x) = 1.945910 + 0.125657(x - 7) - 0.007009(x - 7)(x - 9) + 0.0004226(x - 7)(x - 9)(x - 9.5)$$

At $x = 10$, we have

$$\begin{aligned} p_3(10) &= 1.945910 + 0.125657(10 - 7) - 0.007009(10 - 7)(10 - 9) + 0.0004226(10 - 7)(10 - 9)(10 - 9.5) \\ &= 2.3024879 \end{aligned}$$

The exact value is $f(10) = \ln(10) = 2.302585$, so this is accurate to four decimal places.

We can also examine the intermediate values from the other polynomials:

$$\begin{aligned} p_0(10) &= 1.945910 \\ p_1(10) &= 2.322881 \\ p_2(10) &= 2.301854 \\ p_3(10) &= 2.302488 \end{aligned}$$

The more terms we add, the more the accuracy increases. Notice how $p_0(10)$ is an underestimate, $p_1(10)$ is an overestimate and then $p_2(10)$ underestimates again.

5.1 The Newton–Gregory forward difference formula

Newton's divided difference formula is valid for arbitrarily spaced nodes, which happens frequently in experiments or measurements of real data. However, in many applications, the x_j's are regularly spaced, such as measurements that occur at regular time intervals. In this case, we can simplify the process somewhat.

If there are n points, then they satisfy

$$x_0, x_1 = x_0 + h, x_2 = x_0 + 2h, \ldots, x_n = x_0 + nh.$$

Define the first forward difference of f at x_j by

$$\Delta f_j = f_{j+1} - f_j$$

and the second forward difference of f at x_j by

$$\Delta^2 f_j = \Delta f_{j+1} - \Delta f_j.$$

Continuing like this, we define the kth forward difference of f at x_j by

$$\Delta^k f_j = \Delta^{k-1} f_{j+1} - \Delta^{k-1} f_j \qquad (k = 1, 2, \ldots)$$

Lemma 5.1.

$$f[x_0, x_1, \ldots, x_n] = \frac{1}{n! h^n} \Delta^n f_0$$

Proof. We use induction. For $k = 1$, $x_1 = x_0 + h$, so

$$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0} = \frac{1}{h}(f_1 - f_0) = \frac{1}{1!h} \Delta f_0$$

Now assume true for $n = k$, so that

$$f[x_0, x_1, \ldots, x_k] = \frac{1}{k! h^k} \Delta^k f_0 \tag{9}$$

What about $n = k + 1$? We have $x_{k+1} = x_0 + (k+1)h$ and, by shifting the $n = k$ assumption by one value,

$$f[x_1, \ldots, x_{k+1}] = \frac{1}{k! h^k} \Delta^k f_1.$$

(Note that this is the same as (9).)

Thus

$$\begin{aligned}
f[x_0, x_1, \ldots, x_k, x_{k+1}] &= \frac{f[x_1, \ldots, x_{k+1}] - f[x_0, \ldots, x_k]}{(k+1)h} && \text{by definition} \\
&= \frac{1}{(k+1)h} \left[\frac{1}{k! h^k} \Delta^k f_1 - \frac{1}{k! h^k} \Delta^k f_0 \right] && \text{using the assumption} \\
&= \frac{1}{(k+1)h(k! h^k)} \left[\Delta^k f_1 - \Delta^k f_0 \right] \\
&= \frac{1}{(k+1)! h^{k+1}} \Delta^{k+1} f_0 && \text{by definition.}
\end{aligned}$$

The result thus follows by induction. □

We have

$$\begin{aligned}
x - x_0 &= rh \\
x - x_1 &= x - x_0 + x_0 - x_1 \\
&= rh - h \\
&= (r-1)h \\
x - x_2 &= (r-2)h
\end{aligned}$$

etc.

Using Newton's divided difference formula,

$$f(x) \approx f_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \cdots + (x - x_0)(x - x_1) \cdots (x - x_{n-1}) f[x_0, x_1, \ldots, x_n],$$

we have

$$\begin{aligned}
f(x) &\approx f_0 + rh \frac{1}{1!h} \Delta f_0 + rh(r-1)h \frac{1}{2!h^2} \Delta^2 f_0 + \cdots + rh(r-1)h \cdots (r-n+1)h \frac{1}{n! h^n} \Delta^n f_0 \\
&= f_0 + r \Delta f_0 + \frac{r(r-1)}{2!} \Delta^2 f_0 + \cdots + \frac{r(r-1) \cdots (r-n+1)}{n!} \Delta^n f_0 \\
&= \sum_{s=0}^{n} \binom{r}{s} \Delta^s f_0
\end{aligned}$$

where $\binom{r}{s}$ are the binomial coefficients. This is called the Newton–Gregory forward difference interpolation formula.

Example. Compute $\sinh(0.16)$ from the following values using the Newton–Gregory forward difference formula.

x_j	0.1	0.2	0.3	0.4
$f(x_j)$	0.10016675	0.201336002	0.304520293	0.410752325

We have $r = \frac{x-x_0}{h} = \frac{0.16-0.1}{0.1} = 0.6$. Then we have

$$\Delta f_0 = 0.201336002 - 0.10016675 = 0.101169252$$
$$\Delta f_1 = 0.304520293 - 0.201336002 = 0.103184291$$
$$\Delta f_2 = 0.410752325 - 0.304520293 = 0.106232032$$

Thus, our table becomes

x_j	0.1	0.2	0.3	0.4
$f(x_j)$	0.10016675	0.201336002	0.304520293	0.410752325
Δf_j	0.101169252	0.103184291	0.106232032	

Next, we have

$$\Delta^2 f_0 = 0.103184291 - 0.101169252 = 0.002015039$$
$$\Delta^2 f_1 = 0.106232032 - 0.103184291 = 0.003047741$$

Our table is now

x_j	0.1	0.2	0.3	0.4
$f(x_j)$	0.10016675	0.201336002	0.304520293	0.410752325
Δf_j	0.101169252	0.10360273	0.106232032	
$\Delta^2 f_j$	0.002015039	0.003047741		

Finally, we have

$$\Delta^3 f_0 = 0.003047741 - 0.002015039 = 0.001032702$$

Thus our final table is

x_j	0.1	0.2	0.3	0.4
$f(x_j)$	**0.10016675**	0.201336002	0.304520293	0.410752325
Δf_j	**0.101169252**	0.10360273	0.106232032	
$\Delta^2 f_j$	**0.002015039**	0.003047741		
$\Delta^3 f_j$	**0.001032702**			

where the boxed numbers are the coefficients we need. We thus have

$$\begin{aligned} f(0.16) \approx p_3(0.16) &= 0.10016675 + 0.6(0.101169252) + \frac{0.6(0.6-1)}{2!}(0.002015039) + \frac{0.6(0.6-1)(0.6-2)}{3!}0.001032702 \\ &= 0.10016675 + 0.6(0.101169252) + \frac{0.6(-0.4)}{2}(0.002015039) + \frac{0.6(-0.4)(-1.4)}{6}0.001032702 \\ &= 0.10016675 + 0.060701551 - 0.000241804 + 0.000057831 \\ &= 0.160684328 \end{aligned}$$

The exact answer is $\sinh(0.16) = 0.160683541$, so we have five decimal places of accuracy (six with roundoff).

5.2 Error

The error in the Lagrangian interpolation method is

$$\epsilon_n(x) = f(x) - p_n(x) = (x-x_0)(x-x_1)\cdots(x-x_n)\frac{f^{(n+1)}(t)}{(n+1)!}$$

This error is a general estimate of error for any f approximated by an nth order polynomial, not just restricted to the Lagrangian method. We can thus apply it to the Newton–Gregory forward difference method as well.

In particular, we have

$$\begin{aligned} x - x_0 &= rh \\ x - x_1 &= (r-1)h \\ x - x_2 &= (r-2)h \\ &\vdots \\ x - x_n &= (r-n)h. \end{aligned}$$

Thus the error is

$$\boxed{\epsilon_n(x) = f(x) - p_n(x) = r(r-1)(r-2)\cdots(r-n)\frac{h^{n+1}}{(n+1)!}f^{(n+1)}(t)}$$

Example. Find the error in the cubic approximation of $\sinh(0.16)$.
We need the fourth derivative, so

$$\begin{aligned} f(x) &= \sinh x \\ f'(x) &= \cosh x \\ f''(x) &= \sinh x \\ f'''(x) &= \cosh x \\ f^{(iv)}(x) &= \sinh x \end{aligned}$$

Thus

$$\begin{aligned} \epsilon_3(0.16) &= 0.6(0.6-1)(0.6-2)(0.6-3)\frac{0.1^4}{4!}\sinh t \\ &= 0.6(-0.4)(-1.4)(-2.4)\frac{0.0001}{24}\sinh t \\ &= -0.00000336\sinh t \end{aligned}$$

where $0.1 \le t \le 0.4$. We do not know t precisely, but we can estimate it from the largest and smallest values of $\sinh t$. Since $\sinh t$ is an increasing function, the extreme values only happen at the endpoints, where $\sinh 0.1 = 0.100167$ and $\sinh 0.4 = 0.410752$. Thus

$$\begin{aligned} -0.00000336(0.410752325) \le \epsilon_3 \le -0.00000336(0.10016675) \\ -1.3801278 \times 10^{-6} \le \epsilon_3 \le -3.3656028 \times 10^{-7} \end{aligned}$$

Since $f(x) = p_3(x) + \epsilon_3(x)$, we have

$$\begin{aligned} p_3(0.16) - 1.3801278 \times 10^{-6} \le \sinh(0.16) \le p_3(0.16) - 3.3656028 \times 10^{-7} \\ 0.160684328 - 1.3801278 \times 10^{-6} \le \sinh(0.16) \le 0.160684328 - 3.3656028 \times 10^{-7} \\ 0.160682947 \le \sinh(0.16) \le 0.160683991 \end{aligned}$$

The exact value is 0.160683541 which lies between these two values.

7 Numerical Integration

In general, we can probably differentiate a given function. However, in general, we probably can't integrate it. Thus far, integrals fall into only a small number of categories:

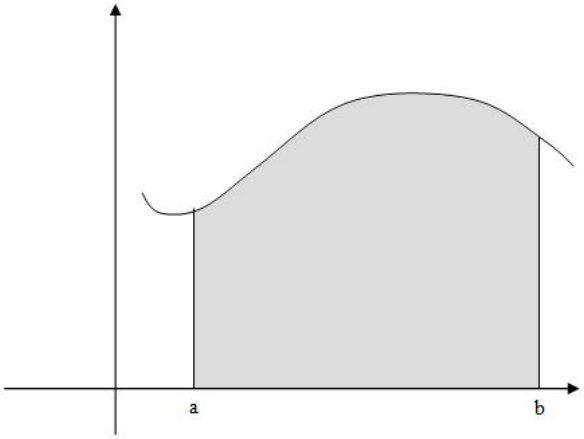
- We know its antiderivative
- Substitution
- Integration by parts

- Some rearrangement, like partial fractions, leads to one of the above options
- That's it.

However, integration is immensely useful. It tells us the area under a curve, which has all sorts of applications. Thus we need a way of estimating such areas. Formally, we need to determine

$$J = \int_a^b f(x)dx$$

where a and b are given values and f is either given analytically by a formula or empirically by a table of values.



If we know F, the antiderivative of f (ie $F'(x) = f(x)$), then

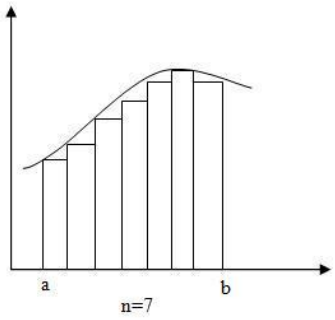
$$J = \int_a^b f(x)dx = F(b) - F(a).$$

However, if we don't know F, then we need to use numerical integration to approximate the area under the curve.

7.1 The rectangular rule

Recall the definition of an integral. It was defined via Riemann sums.

Definition 7.1. *The integral is defined as* $\int_a^b f(t)dt = \lim_{n\to\infty} \sum_{i=1}^{n-1} f(t_i)h$ *where the values* $t_0, ..., t_n$ *break the interval from* a *to* b *into* n *pieces, each of width* $h = \frac{b-a}{n}$.

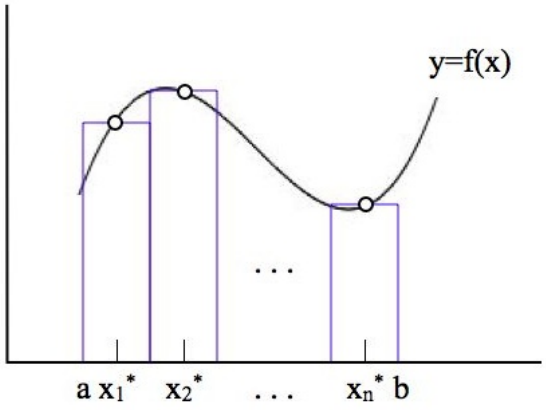


Here the Riemann sums are on the right, but it's equivalent to take them on the left or in the centre (since $h \to 0$ as $n \to \infty$).

The rectangular rule takes this idea and uses it to approximate the integral. Let x_j^* be the midpoint of the interval $[x_j, x_{j+1}]$. Then

$$J = \int_a^b f(x)dx \approx h\left[f(x_1^*) + f(x_2^*) + \cdots + f(x_n^*)\right]$$

That is, the area under the curve is approximately the area of each of the Riemann rectangles.



Obviously, the larger n is, the smaller h is and the more accurate the approximation is.

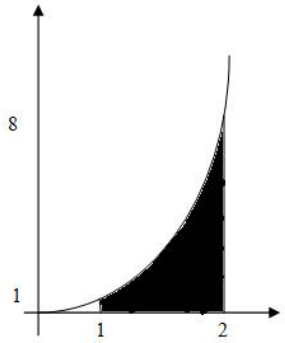
Example. Evaluate $\int_1^2 x^3 dx$ using the rectangular rule with 10 Riemann sums. Compare this to the actual value.

For 10 sums, we have $h = \frac{2-1}{10} = 0.1$. Thus, each interval has width 0.1, so the midpoints are $x_1^* = 1.05, x_2 = 1.15, \ldots, x_{10} = 1.95$.

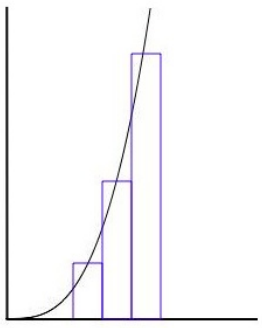
$$Area \approx 0.1(1.05^3 + 1.15^3 + 1.25^3 + 1.35^3 + 1.45^3 + 1.55^3 + 1.65^3 + 1.75^3 + 1.85^3 + 1.95^3) = 3.74625 \text{ units}^2$$

The actual value is

$$\begin{aligned}\int_1^2 x^3 dx &= \frac{x^4}{4}\Big|_1^2 \\ &= \frac{2^4}{4} - \frac{1}{4} \\ &= 3.75 \text{ units}^2\end{aligned}$$



The reason this is so accurate is because the rectangular rule does particular well with monotone functions.



What you miss out on the razzle dazzle, you pick up on the hurdy gurdy.

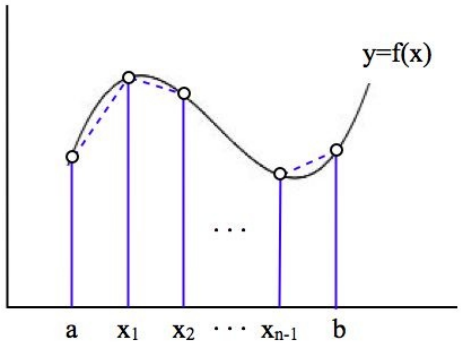
7.2 The trapezoidal rule

The trapezoidal rule is more accurate than the rectangular rule and can cope with nonmonotone functions. We take the same subdivision as before, but now approximate f by a broken line of segments (chords) with

endpoints $[a, f(a)], [x_1, f(x_1)], \ldots [x_{n-1}, f(x_{n-1})], [b, f(b)]$, where $x_1, x_2, \ldots, x_{n-1}$ are interior endpoints of the Riemann sums.

The area under the curve is then approximated by n trapezoids, with areas

$$\frac{1}{2}\left[f(a) + f(x_1)\right] h, \qquad \frac{1}{2}\left[f(x_1) + f(x_2)\right] h, \qquad \cdots, \frac{1}{2}\left[f(x_{n-1}) + f(b)\right] h,$$



Thus, the area under the curve is approximated by

$$J = \int_a^b f(x)dx \approx h\left[\frac{1}{2}f(a) + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(b))\right]$$

(Note that each interior point is counted twice, while the endpoints are only counted once. Also, there are actually $n + 1$ points to be evaluated.)

Example. Evaluate $J = \int_{-1}^{1} e^{-x^2} dx$ using the trapezoidal rule, with $n = 10$

Note that this is a function we can't integrate, yet we'd like to know the area, since this is the bell curve, which is frequently used in statistics. Formally, it's a probability distribution curve, which means that probabilities are calculated from the area under the curve.

First we have $n = \frac{1-(-1)}{10} = 0.2$. The computations are then:

j	xj	exp(-xj^2)	exp(-xj^2)
0	-1	0.36787944	
1	-0.8		0.52729242
2	-0.6		0.69767633
3	-0.4		0.85214379
4	-0.2		0.96078944
5	0		1
6	0.2		0.96078944
7	0.4		0.85214379
8	0.6		0.69767633
9	0.8		0.52729242
10	1	0.36787944	
Sums		0.73575888	7.07580396

The result is thus $J \approx 0.2[\frac{1}{2}(0.73575888) + 7.07580396] = 1.48873668$

Exercise. Evaluate $J = \int_{-1}^{1} e^{-x^2} dx$ using the rectangular rule, with $n = 10$. Explain anything surprising about this result.

7.2.1 Error bounds for the trapezoidal rule

Recall that the error in the Lagrange polynomial approximation was

$$\epsilon_n(x) = f(x) - p_n(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(t)}{(n+1)!}, \tag{10}$$

We thus have

$$f(x) - p_1(x) = (x - x_0)(x - x_1)\frac{f''(t)}{2},$$

where t depends on x and is between x_0 and x_1. Integrating from $a = x_0$ to $x_1 = x_0 + h$, we have

$$\int_{x_0}^{x_0+h} f(x)dx - \frac{h}{2}[f(x_0) + f(x_1)] = \int_{x_0}^{x_0+h} (x - x_0)(x - x_0 - h)\frac{f''(t(x))}{2}dx.$$

Let $v = x - x_0$ and note that $v(v - h) = v^2 - vh$ does not change sign in the interval $[0, h]$. Thus we can use the mean value theorem. Changing variables, we have

$$\begin{aligned}\int_{x_0}^{x_0+h} (x - x_0)(x - x_0 - h)\frac{f''(t(x))}{2}dx &= \int_0^h v(v - h)\frac{f''(t(v))}{2}dv \\ &= \int_0^h v(v - h)dv\frac{f''(\tilde{t})}{2}\end{aligned}$$

by the mean value theorem, where $\tilde{t}$ is some value between $v = 0$ and $v = h$ (ie between x_0 and x_1). We thus have

$$\begin{aligned}\int_0^h v(v - h)dv\frac{f''(\tilde{t})}{2} &= \int_0^h (v^2 - vh)dv\frac{f''(\tilde{t})}{2} \\ &= \left[\frac{v^3}{3} - \frac{v^2h}{2}\right]_0^h \frac{f''(\tilde{t})}{2} \\ &= \left(\frac{h^3}{3} - \frac{h^3}{2}\right)\frac{f''(\tilde{t})}{2} \\ &= -\frac{h^3}{12}f''(\tilde{t})\end{aligned}$$

The error ϵ of the trapezoidal rule is the sum of n such contributions from the n subintervals. Thus

$$\begin{aligned}\epsilon &= -n\frac{h^3}{12}f''(\tilde{t}) \\ &= -\frac{(b - a)^3}{12n^2}f''(\tilde{t})\end{aligned}$$

since $h = \frac{(b-a)}{n}$ and where $\tilde{t}$ is some suitable value between a and b. To deal with the $\tilde{t}$ issue, we take the largest and smallest values of f'' in the interval $[a, b]$. Call these M_2 and M_2^*, respectively. Then the error satisfies

$$\boxed{KM_2 \le \epsilon \le KM_2^* \qquad \text{where} \qquad K = -\frac{(b - a)^3}{12n^2}.}$$

Example. Estimate the error of the approximate value in the example $J = \int_{-1}^{1} e^{-x^2}dx$.
We have $a = -1$, $b = 1$ and $n = 10$, so

$$\begin{aligned}K &= -\frac{(b - a)^3}{12n^2} \\ &= -\frac{(1 - (-1))^3}{12(100)} \\ &= -\frac{1}{150}\end{aligned}$$

All we need to find is the second derivative and its appropriate bounds. Thus

$$\begin{aligned}f(x) &= e^{-x^2} \\ f'(x) &= -2xe^{-x^2} \\ f''(x) &= -2e^{-x^2} + 4x^2e^{-x^2} \\ f'''(x) &= 4xe^{-x^2} + 8xe^{-x^2} - 8x^3e^{-x^2} = 12xe^{-x^2} - 8x^3e^{-x^2}.\end{aligned}$$

(We need f''' in order to find out when f'' is maximal.)

Hence

$$f'''(x) = 0 \Rightarrow x = 0 \text{ or } 3 - 2x^2 = 0$$
$$x = \pm\sqrt{\frac{3}{2}}$$

Since $\pm\sqrt{\frac{3}{2}}$ are outside the interval $[-1, 1]$, the maxima and minima can only occur at either $x = 0$ or the endpoints $x = \pm 1$. Testing each, we have

$$f''(-1) = 0.73575888$$
$$f''(0) = -2$$
$$f''(1) = 0.73575888$$

Thus $M_2 = 0.73575888$ and $M_2^* = -2$.

It follows that the error satisfies

$$-0.73575888\frac{1}{150} \leq \epsilon \leq 2\frac{1}{150}$$
$$-0.004905258 \leq \epsilon \leq 0.01333333$$

The exact value of J must lie between

$$1.48873668 - 0.004905258 = 1.483831421 \text{ and } 1.48873668 + 0.01333333 = 1.50207001$$

(In fact, the exact value of J is 1.493648 to 6 decimal places.)

7.3 Simpson's Rule

The rectangular rule comes from piecewise constant approximation of f. The trapezoidal rule comes from piecewise linear approximation of f. Simpson's rule uses piecewise quadratic approximation.

Simpson's rule is relatively simple, but also very accurate, so it is sufficient for most problems.

Since we are dealing with parabolic interpolation, each parabola needs three points, rather than two as with the previous methods. This will increase our number of points. In particular, we need to divide the interval $[a, b]$ into an even number of equal subintervals. If we have $n = 2m$ subintervals, the length of each is

$$h = \frac{b-a}{2m}$$

and the endpoints are

$$x_0 = a, x_1, x_2, \ldots, x_{2m-1}, x_{2m} = b.$$

Consider the first two subintervals. We want to approximate $f(x)$ in the interval $x_0 \leq x \leq x_2 = x_0 + 2h$ by the Lagrange polynomial $p_2(x)$ passing through (x_0, f_0), (x_1, f_1) and (x_2, f_2).

Recall that the Lagrange formula for $p_2(x)$ was

$$p_2(x) = L_0(x)f_0 + L_1(x)f_1 + L_2(x)f_2$$

where

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}$$
$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$
$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

We thus have

$$\begin{aligned} p_2(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f_2 \\ &= \frac{(x-x_1)(x-x_2)}{2h^2}f_0 + \frac{(x-x_0)(x-x_2)}{-h^2}f_1 + \frac{(x-x_0)(x-x_1)}{2h^2}f_2. \end{aligned}$$

Let $s = \frac{x-x_1}{h}$, so that

$$\begin{aligned} x - x_1 &= sh \\ x - x_0 &= x - x_1 + x_1 - x_0 = sh + h = (s+1)h \\ x - x_2 &= x - x_1 + x_1 - x_2 = sh - h = (s-1)h. \end{aligned}$$

Thus

$$\begin{aligned} p_2(x) &= \frac{(x-x_1)(x-x_2)}{2h^2}f_0 + \frac{(x-x_0)(x-x_2)}{-h^2}f_1 + \frac{(x-x_0)(x-x_1)}{2h^2}f_2 \\ &= \frac{(sh)((s-1)h)}{2h^2}f_0 + \frac{((s+1)h)((s-1)h)}{-h^2}f_1 + \frac{((s+1)h)(sh)}{2h^2}f_2 \\ &= \frac{1}{2}s(s-1)f_0 - (s+1)(s-1)f_1 + \frac{1}{2}(s+1)sf_2. \end{aligned}$$

We now integrate x from x_0 to x_2, which corresponds to integrating s from -1 to 1. Since $dx = hds$, the result is

$$\begin{aligned} \int_{x_0}^{x_2} f(x)dx &\approx \int_{x_0}^{x_2} p_2(x) \\ &= \int_{-1}^{1} \left[\frac{1}{2}s(s-1)f_0 - (s+1)(s-1)f_1 + \frac{1}{2}(s+1)sf_2\right] hds \\ &= h\left[\frac{1}{2}\left(\frac{s^3}{3} - \frac{s^2}{2}\right)f_0 - \left(\frac{s^3}{3} - s\right)f_1 + \frac{1}{2}\left(\frac{s^3}{3} + \frac{s^2}{2}\right)f_2\right]_{-1}^{1} \\ &= h\left[\frac{1}{2}\left(\frac{1}{3} - \frac{1}{2}\right)f_0 - \left(\frac{1}{3} - 1\right)f_1 + \frac{1}{2}\left(\frac{1}{3} + \frac{1}{2}\right)f_2\right] \\ &\quad - h\left[\frac{1}{2}\left(\frac{-1}{3} - \frac{1}{2}\right)f_0 - \left(\frac{-1}{3} - (-1)\right)f_1 + \frac{1}{2}\left(\frac{-1}{3} + \frac{1}{2}\right)f_2\right] \\ &= h\left[\frac{1}{3}f_0 + (2 - \frac{2}{3})f_1 + \frac{1}{3}f_2\right] \\ &= \frac{h}{3}\left[f_0 + 4f_1 + f_2\right] \end{aligned}$$

A similar formula holds for the next two subintervals from x_2 to x_4, and so on. When we sum all these m formulas, we count the two extreme endpoints (x_0 and x_{2m}) just once, the odd midpoints 4 times (from the above formula) and the even endpoints twice (since they get counted at the end of one group and the beginning of another).

Putting it all together, we obtain Simpson's rule:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 2f_{2m-2} + 4f_{2m-1} + f_{2m}\right]$$

Example. Evaluate $J = \int_{-1}^{1} e^{-x^2}dx$ using Simpson's rule with $2m = 10$.
First note that $h = 0.2$ as before. The calculations are given by:

j	xj	exp(-xj^2)	exp(-xj^2)	exp(-xj^2)
0	-1	0.36787944		
1	-0.8		0.52729242	
2	-0.6			0.69767633
3	-0.4		0.85214379	
4	-0.2			0.96078944
5	0		1	
6	0.2			0.96078944
7	0.4		0.85214379	
8	0.6			0.69767633
9	0.8		0.52729242	
10	1	0.36787944		
Sums		0.73575888	3.75887243	3.31693153

Then the integral is

$$J \approx \frac{0.2}{3}\left[0.73575888 + 4(3.75887243) + 2(3.31693153)\right] = 1.493674111$$

Note that this is much more accurate than the answer we derive from the trapezoidal rule.

7.3.1 Error estimates for Simpson's rule

The error can be obtained similarly to the error in the trapezoidal rule, assuming that the fourth derivative of f exists and is continuous in the interval of integration. Thus let M_4 be the largest value of the fourth derivative of f in $[a, b]$ and M_4^* be the smallest value. Then

$$\boxed{CM_4 \le \epsilon_S \le CM_4^* \qquad \text{where} \qquad C = -\frac{(b-a)^5}{180(2m)^4}}$$

Note that $\epsilon_S = 0$ for both quadratic polynomial (which makes sense, since we are interpolating it with quadratic polynomials), but also for a cubic polynomial, since the fourth derivative of such a polynomial is zero.

Example. Estimate the error in the approximation of $J = \int_{-1}^{1} e^{-x^2} dx$ using Simpson's rule with $2m = 10$.

We already have

$$f'''(x) = 12xe^{-x^2} - 8x^3e^{-x^2}.$$

Thus

$$\begin{aligned}
f^{(iv)}(x) &= 12e^{-x^2} - 24x^2e^{-x^2} - 24x^2e^{-x^2} + 16x^4e^{-x^2} \\
&= 12e^{-x^2} - 48x^2e^{-x^2} + 16x^4e^{-x^2} \\
f^{(v)}(x) &= -24xe^{-x^2} - 96xe^{-x^2} + 96x^3e^{-x^2} + 64x^3e^{-x^2} - 32x^5e^{-x^2} \\
&= -120xe^{-x^2} + 160x^3e^{-x^2} - 32x^5e^{-x^2} \\
&= -8xe^{-x^2}(4x^4 - 20x^2 + 15) = 0 \\
\Rightarrow x = 0, x^2 &= \frac{20 \pm \sqrt{160}}{8} = 4.08, 0.91886
\end{aligned}$$

Thus, the points we need to check are $x = 0, \pm 1, \pm\sqrt{0.91886}$. We have

$$\begin{aligned}
f^{(iv)}(0) &= 12 \\
f^{(iv)}(-1) &= -7.357588823 \\
f^{(iv)}(1) &= -7.357588823 \\
f^{(iv)}(\sqrt{0.91886}) &= -7.419481177 \\
f^{(iv)}(-\sqrt{0.91886}) &= -7.419481177
\end{aligned}$$

Hence $M_4 = 12$ and $M_4^* = -7.419481177$. We also have

$$C = -\frac{2^5}{180(10)^4}.$$

The error is then

$$-0.000213333 \leq \epsilon_S \leq 0.000131901.$$

Hence J must lie between

$$1.493674111 - 0.000213333 = 1.493460777 \qquad \text{and} \qquad 1.493674111 + 0.000131901 = 1.493806012.$$

Thus our approximation is accurate to at least three decimal places. (Recall that the exact value of J is 1.493648 to 6 decimal places.)

The trapezoidal rule gave us 0 decimal places of accuracy, whereas Simpson's rule gives us many more.

This presents an obvious inversion: we used some given number of iterations and then determined the degree of accuracy. However, more likely a certain degree of accuracy will be required and the number of iterations then determined from it. This means that, given ϵ_S, we solve for $n = 2m$.

Example. What n should we choose in the Simpson's rule approximation of $J = \int_{-1}^{1} e^{-x^2} dx$ in order to get six decimal places of accuracy?

Note that $|M_4| = 12 > 7.419481177 = |M_4^*|$. Thus in order to get six decimal places of accuracy, we require

$$|CM_4| = \frac{12(2^5)}{180(2m)^4} = \frac{1}{2} \times 10^{-6}.$$

(The $\frac{1}{2}$ is so that the seventh decimal is less than 5, to avoid roundup.)

Solving for m, we have

$$m = \left[\frac{2 \cdot 10^6 \cdot 12 \cdot 32}{180 \cdot 2^4}\right]^{1/4} = 22.72$$

Thus we should choose $n = 2m = 46$.

Exercise. Use Simpson's rule with $n = 46$ to evaluate $J = \int_{-1}^{1} e^{-x^2} dx$. (Use Excel, don't do it by hand.)

7.4 Gaussian Quadrature

All our techniques thus far involve function values for equidistant x-values and give exact results for polynomials less than or equal to a certain degree. More generally, we may set

$$\int_{-1}^{1} f(x)dx \approx \sum_{j=1}^{n} A_j f_j \tag{11}$$

where $a = -1$ and $b = 1$, which can be accomplished by a linear transformation of scale. Specifically, we let $x = \frac{a+b}{2} + \frac{b-a}{2}t$. Then

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right) dt.$$

We determine the $2n$ constants $A_1, \ldots, A_n, x_1, \ldots, x_n$ so that (11) gives exact results for polynomials of degree k, where k is as high as possible. Since $2n$ is the number of coefficients of a polynomial of degree $2n-1$, it follows that $k \leq 2n-1$.

Gauss showed that exactness for polynomals of degree not exceeding $2n-1$ can be attained if and only if the values $x_1, \ldots, x_n$ are the n zeroes of the Legendre polynomial $P_n(x)$, where

$$P_0 = 1, P_1(x) = x, P_2(x) = \frac{1}{2}(3x^2 - 1), P_3(x) = \frac{1}{2}(5x^3 - 3x), \ldots$$

The coefficients can be determined from the following table:

n	Zeros		Coefficients
2	$\pm 1/\sqrt{3}$	$=\pm 0.5773502692$	1
3	0		8/9
	$\pm\sqrt{3/5}$	$=\pm 0.7745966692$	5/9
4	$\pm\sqrt{(15-\sqrt{120})/35}$	$=\pm 0.3399810436$	0.6521451549
	$\pm\sqrt{(15+\sqrt{120})/35}$	$=\pm 0.8611363116$	0.3478548451
	0		0.5688888889
5	± 0.5384693101		0.4786286705
	± 0.9061798459		0.2369268850

Example. Use Gaussian quadrature with $n = 2$ and $n = 3$ to calculate $\int_2^3 \frac{1}{\ln x} dx$.

First we need to make a transformation. When $x = 2$, $t = -1$ and when $x = 3$, $t = 1$. Thus $x = \frac{1}{2}(t+5)$ and $dx = \frac{1}{2}dt$. For $n = 2$, we have

$$\begin{aligned}\int_2^3 \frac{1}{\ln x} dx &= \int_{-1}^1 \frac{1}{\ln\left[\frac{1}{2}(t+5)\right]} \frac{1}{2} dt \\ &= \frac{1}{2}\left\{ \frac{1}{\ln\left[\frac{1}{2}\left(5-\frac{1}{\sqrt{3}}\right)\right]} + \frac{1}{\ln\left[\frac{1}{2}\left(5+\frac{1}{\sqrt{3}}\right)\right]} \right\} \\ &= \frac{1}{2}\left\{1.26009363 + 0.975070736\right\} = 1.117582183\end{aligned}$$

For $n = 3$, we have

$$\begin{aligned}\int_2^3 \frac{1}{\ln x} dx &= \int_{-1}^1 \frac{1}{\ln\left[\frac{1}{2}(t+5)\right]} \frac{1}{2} dt \\ &= \frac{1}{2}\left\{ \frac{5}{9}\frac{1}{\ln\left[\frac{1}{2}\left(5-\sqrt{\frac{3}{5}}\right)\right]} + \frac{8}{9}\frac{1}{\ln\frac{5}{2}} + \frac{5}{9}\frac{1}{\ln\left[\frac{1}{2}\left(5+\sqrt{\frac{3}{5}}\right)\right]} \right\} \\ &= \frac{1}{2}\left\{0.742753564 + 0.970094815 + 0.52950231\right\} = 1.118399305\end{aligned}$$

The Gaussian quadrature formula has the advantage of high accuracy. Its disadvantages are the irregular spacing of $x_1, \ldots, x_n$ and the inconvenient values of the coefficients.

8 Forward, Backward and Improved Euler Methods

8.1 Idea

Recall that the definition of the derivative was

$$\frac{dy}{dt} = \lim_{h\to 0} \frac{y(t+h) - y(t)}{h}$$

This suggests that one might approximate the derivative by

$$\frac{dy}{dt} \approx \frac{y(t+h) - y(t)}{h}$$

for small enough h.

8.2 Forward Euler

Suppose that we are given

$$\frac{dy}{dt} = f(y,t)$$

We approximate at some $t = t^*$

$$\frac{y(t^*+h) - y(t^*)}{h} \approx f(y(t^*), t^*)$$

This can be rearranged to give

$$y(t^*+h) = y(t^*) + hf(y(t^*), t^*)$$

Thus, if we are given the value of the function y at some t then we can approximate the value at time $t+h$.

8.3 Notation

A notation that is often used is given by dividing the t axis into intervals of uniform length by defining

$$t_i = ih$$

where h is the length of each subinterval (assumed to be small). We also write

$$y_i = y(t_i)$$

Then our approximation can be written as

$$y_{i+1} = y_i + hf(y_i, t_i)$$

The initial condition is y_0 which is given. Then the first few values are

$$\begin{aligned} y_1 &= y_0 + hf(y_0, t_0) \\ y_2 &= y_1 + hf(y_1, t_1) \\ y_3 &= y_2 + hf(y_2, t_2) \end{aligned}$$

This is an explicit method since we are not required to solve any algebraic equation in going to the next step. We have approximated a continuous problem by a discrete one.

Example. Apply the forward Euler method to

$$y' = x + y, \qquad y(0) = 0 \tag{12}$$

using $h = 0.2$. Find y_1, y_2, y_3, y_4, y_5.

We have $f(x,y) = x + y$, so

$$y_{n+1} = y_n + 0.2(x_n + y_n).$$

Since $h = 0.2$, the x values are $(x_0, x_1, x_2, x_3, x_4, x_5) = (0, 0.2, 0.4, 0.6, 0.8, 1.0)$.

Thus

$$\begin{aligned} y_1 &= y_0 + 0.2(x_0 + y_0) \\ &= 0 + 0.2(0+0) = 0 \\ y_2 &= y_1 + 0.2(x_1 + y_1) \\ &= 0 + 0.2(0.2 + 0) = 0.04 \\ y_3 &= y_2 + 0.2(x_2 + y_2) \\ &= 0.04 + 0.2(0.4 + 0.04) = 0.128 \\ y_4 &= y_3 + 0.2(x_3 + y_3) \\ &= 0.128 + 0.2(0.6 + 0.128) = 0.2736 \\ y_5 &= y_4 + 0.2(x_4 + y_4) \\ &= 0.2736 + 0.2(0.8 + 0.2736) = 0.48832 \end{aligned}$$

8.4 Error

The relative error is given by

$$error = \left| \frac{y(t_i) - y_i}{y(t_i)} \right|.$$

Example. Consider the initial value problem

$$\frac{dy}{dt} = -y, \; y(0) = 1$$

The analytic solution is

$$y(t) = e^{-t}$$

The forward Euler method gives the approximation

$$y_{i+1} = y_i - hy_i.$$

On the interval from 0 to 1 using a spacing of 0.1 find the y_i.

$$\begin{aligned} y_0 &= y(0) = 1 \\ y_1 &= y_0 - 0.1y_0 = 1 - 0.1 = 0.9 \\ y_2 &= 0.9 - (0.1)(0.9) = 0.81 \\ y_3 &= 0.81 - (0.1)(0.81) = 0.729 \end{aligned}$$

$$\begin{aligned} y(0) &= 1 && \text{error} = 0 \\ y(0.1) &= e^{-0.1} = 0.9048 && \text{error} = 0.00535 \\ y(0.2) &= e^{-0.2} = 0.8187 && \text{error} = 0.01066 \end{aligned}$$

Exercise. Show that

$$y(x) = e^x - x - 1$$

is the analytical solution to (12). Use this to find the error in the Forward Euler approximation of that example.

8.5 Error analysis

We would like to know how the error varies with the spacing h. This can be done by considering the Taylor Series:

$$y(t_i + h) = y(t_i) + hy'(t_i) + \frac{1}{2}h^2y''(t_i) + ...$$

We can solve for the first derivative to get

$$y'(t_i) = \frac{y(t_i + h) - y(t_i)}{h} - \frac{1}{2}hy''(t_i) + ...$$

Recall that $t_{i+h} = t_i + h$ and so the previous formula is

$$y'(t_i) = \frac{y_{i+1} - y_i}{h} - \frac{1}{2}hy''(t_i) + ...$$

The first term is the one we used in our approximation. The second order term gives us an estimate of the error. We don't know what the second derivative is but as long as it is finite we can see that the error goes as h for h small.

$$E = \frac{1}{2}hy''(t)$$

The forward Euler method is called a first-order method, because we only take constant terms and the term containing the first power of h. Thus, we are ignoring terms of order h^2, h^3, etc. If h is small, the error is thus dominated by h. This is called the local truncation error.

8.6 Backward Euler

We previously did the approximation at $t = t_i$. We could have used $t = t_{i+1}$. This gives the backward Euler method

$$y_{i+1} = y_i + hf(y_{i+1}, t_{i+1})$$

This is an implicit method since we now have to solve an equation for y_{i+1}.

Example. Backward Euler: Find the first three values of the backward Euler approximation for

$$y' = 3t - y^2, \qquad y(0) = \frac{3}{4}, \qquad h = \frac{1}{2} \qquad \text{(solutions satisfying } y(t) \geq 0)$$

$$\begin{aligned}
y_{n+1} &= y_n + hf(t_{n+1}, y_{n+1}) \\
&= y_n + \frac{1}{2}(3t_{n+1} - y_{n+1}^2) \\
2y_{n+1} &= 2y_n + 3t_{n+1} - y_{n+1}^2 \\
y_{n+1}^2 + 2y_{n+1} &= 2y_n + 3t_{n+1} \\
y_{n+1}^2 + 2y_{n+1} + 1 &= 2y_n + 3t_{n+1} + 1 \\
(y_{n+1} + 1)^2 &= 2y_n + 3t_{n+1} + 1 \\
y_{n+1} &= -1 + \sqrt{1 + 2y_n + 3t_{n+1}} \\
y_1 &= -1 + \sqrt{1 + 2(\frac{3}{4}) + 3(1/2)}, \qquad (t_0 = 0, \qquad t_1 = \frac{1}{2}) \\
&= -1 + \sqrt{4} \\
&= -1 + 2 = 1 \\
y_2 &= -1 + \sqrt{1 + 2(1) + 3} \qquad (t_2 = 1) \\
&= -1 + \sqrt{6} \\
y_3 &= -1 + \sqrt{1 + 2(-1 + \sqrt{6}) + 3(3/2)} \qquad (t_3 = \frac{3}{2}) \\
&= -1 + \sqrt{7/2 + 2\sqrt{6}}
\end{aligned}$$

Example. Consider the equation

$$y' = \sin(x), \ y(0) = 1.0$$

The forward Euler method gives

$$y_{i+1} = y_i + h\sin(y_i)$$

and the backward Euler method gives

$$y_{i+1} = y_i + h\sin(y_{i+1})$$

To apply the backward Euler method, we would have to solve a nonlinear equation at each step. We can't go any further with backward Euler by hand; this is where the computer comes in.

8.7 Error analysis and stability

The accuracy of the backward Euler is not better than the forward Euler. However, the backward Euler has better stability properties.

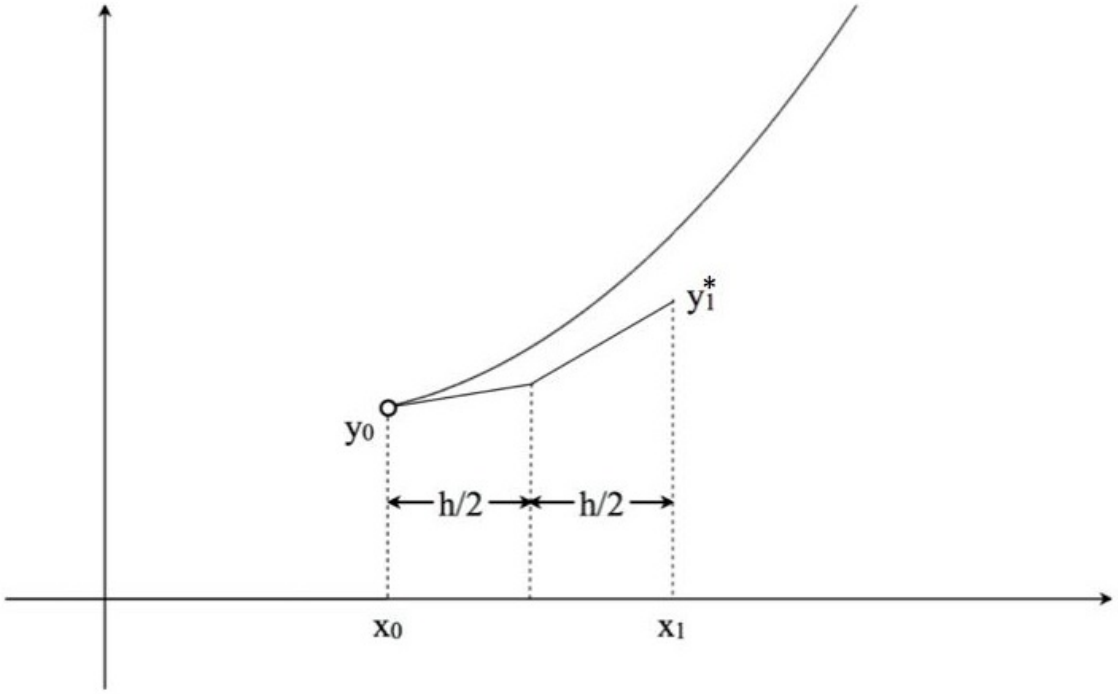
8.8 Improved Euler

A mixture of the backward and forward Euler turns out to be more accurate. We get the method by taking an average:

$$\frac{y_{i+1} - y_i}{h} = \frac{1}{2}f(y_{i+1}, t_{i+1}) + \frac{1}{2}f(y_i, t_i)$$

which can be rewritten as

$$y_{i+1} - 0.5hf(y_{i+1}, t_{i+1}) = y_i + 0.5hf(y_i, t_i)$$



In the interval from x_n to $x_n + \frac{1}{2}h$, we approximate the solution y by the straight line through (x_n, y_n) with slope $f(x_n, y_n)$ and we then continue along the straight line with slope $f(x_{n+1}, y_{n+1})$ until x reaches x_{n+1}.

We would still have to solve an algebraic equation at each step and so the method is slower than the forward Euler but it is more accurate for any given h so we do not have to take so many steps.

Using the notations

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_{n+1}, y^*_{n+1}) \end{aligned}$$

where y^*_{n+1} is the approximated value of y_{n+1}, we can summarise the method as follows. For $n = 0, 1, \ldots, N-1$, we calculate

$$\begin{aligned} x_{n+1} &= x_n + h \\ k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_{n+1}, y_n + k_1) \\ y_{n+1} &= y_n + \frac{1}{2}(k_1 + k_2). \end{aligned}$$

Example. Apply the improved Euler method to $y' = x + y, y(0) = 0$ using $h = 0.2$ as before for the interval $0 \le x \le 2$ and calculate the error.

We have

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ &= 0.2(x_n + y_n) \\ k_2 &= hf(x_{n+1}, y_{n+1}) \\ &= 0.2\{[x_n + 0.2] + [y_n + 0.2(x_n + y_n)]\}. \end{aligned}$$

Then

$$\begin{aligned} y_{n+1} &= y_n + \frac{0.2}{2}(2.2x_n + 2.2y_n + 0.2) \\ &= y_n + 0.22(x_n + y_n) + 0.02 \end{aligned}$$

Hence

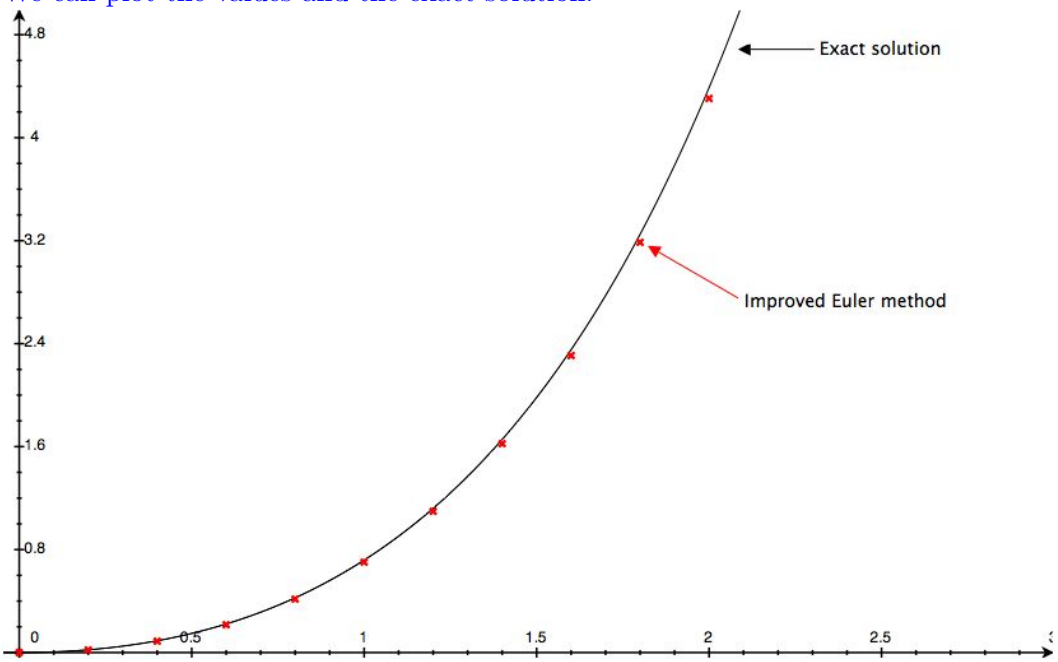
$$
\begin{aligned}
y_1 &= y_0 + 0.22(x_0 + y_0) + 0.02 \\
&= 0 + 0.22(0 + 0) + 0.02 = 0.02 \\
y_2 &= y_1 + 0.22(x_1 + y_1) + 0.02 \\
&= 0.02 + 0.22(0.2 + 0.02) + 0.02 = 0.0884 \\
y_3 &= y_2 + 0.22(x_2 + y_2) + 0.02 \\
&= 0.0884 + 0.22(0.4 + 0.0884) + 0.02 = 0.215848 \\
y_4 &= y_3 + 0.22(x_3 + y_3) + 0.02 \\
&= 0.215848 + 0.22(0.6 + 0.215848) + 0.02 = 0.41533456 \\
y_5 &= y_4 + 0.22(x_4 + y_4) + 0.02 \\
&= 0.41533456 + 0.22(0.8 + 0.41533456) + 0.02 = 0.7027081
\end{aligned}
$$

etc.

The values are thus:

n	xn	yn	0.22xn+1.22yn+0.02	Exact value (e^x-x-1)	Error
0	0	0	0.02	0	0
1	0.2	0.02	0.0884	0.021402758	0.0014028
2	0.4	0.0884	0.215848	0.091824698	0.0034247
3	0.6	0.215848	0.41533456	0.2221188	0.0062708
4	0.8	0.41533456	0.702708163	0.425540928	0.0102064
5	1	0.702708163	1.097303959	0.718281828	0.0155737
6	1.2	1.097303959	1.62271083	1.120116923	0.0228130
7	1.4	1.62271083	2.307707213	1.655199967	0.0324891
8	1.6	2.307707213	3.1874028	2.353032424	0.0453252
9	1.8	3.1874028	4.304631415	3.249647464	0.0622447
10	2	4.304631415		4.389056099	0.0844247

We can plot the values and the exact solution.



Thus, the Improved Euler method is a good approximation, much better than the Forward Euler method. The Improved Euler method is a second-order method, because the truncation error per step is of order h^3.

9 Runge-Kutta method

The Runge-Kutta method is the method most used by software programs to numerically solve ODEs. It has the advantage of being easy to program, needs no special starting procedure, no estimation and makes light demands on storage.

Consider the differential equation

$$\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0.$$

We want to solve the differential equation for $x_0 \le x \le x_N$.

Take a series of the equidistant points

$$x_1 = x_0 + h, x_2 = x_0 + 2h, \ldots, x_N = x_0 + Nh.$$

For $n = 0, 1, \ldots, N-1$, compute

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\ k_3 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right) \\ k_4 &= hf\left(x_n + h, y_n + k_3\right). \end{aligned}$$

Then

$$\begin{aligned} x_{n+1} &= x_n + h \\ y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

The truncation error per step is of the order h^5, so this is a fourth-order method. As a result, it is sometimes called the fourth-order Runge-Kutta method, because there are higher-order Runge-Kutta methods based on the same principles.

The downside is that the hand calculations are laborious, except in very simple examples. However, this is not a problem for the computer. This method is the most common method used in Matlab to solve ODEs.

Example. Solve

$$y' = x + y, \qquad y(0) = 0$$

using five steps of the Runge-Kutta method and a stepsize of $h = 0.2$.

We have $f(x, y) = x + y$. Then

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ &= 0.2(x_n + y_n) \\ k_2 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\ &= 0.2(x_n + 0.1 + y_n + 0.5k_1) \\ &= 0.2[x_n + 0.1 + y_n + 0.5(0.2(x_n + y_n))] \\ &= 0.22(x_n + y_n) + 0.02 \\ k_3 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right) \\ &= 0.2(x_n + 0.1 + y_n + 0.5k_2) \\ &= 0.2[x_n + 0.1 + y_n + 0.5(0.22(x_n + y_n) + 0.02)] \\ &= 0.222(x_n + y_n) + 0.022 \\ k_4 &= hf\left(x_n + h, y_n + k_3\right) \\ &= 0.2(x_n + 0.2 + y_n + k_3) \\ &= 0.2[x_n + 0.2 + y_n + 0.222(x_n + y_n) + 0.022] \\ &= 0.2444(x_n + y_n) + 0.0444. \end{aligned}$$

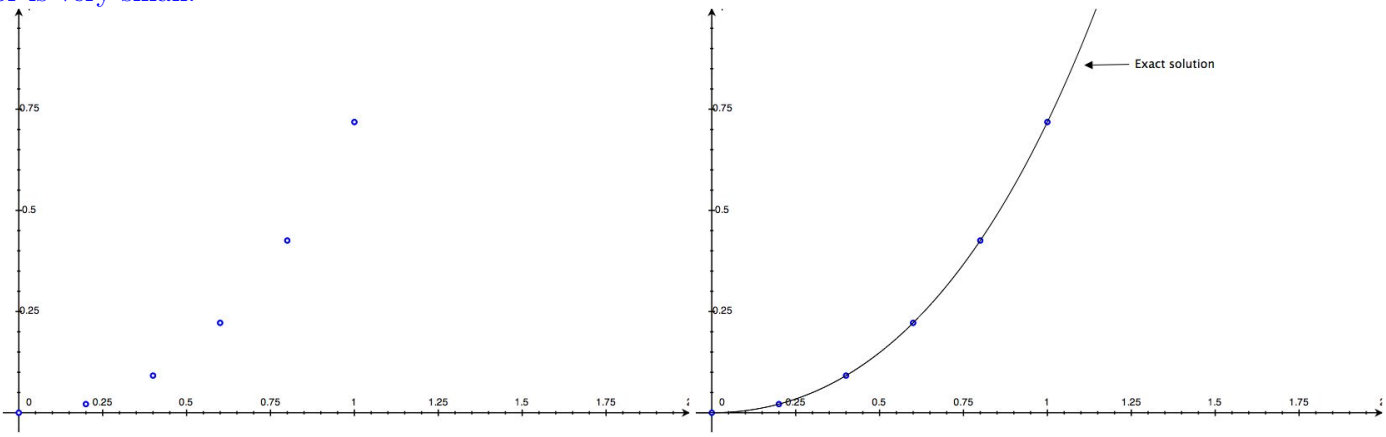
We thus have

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ &= y_n + \frac{1}{6}\{[0.2(x_n + y_n)] + 2[0.22(x_n + y_n) + 0.02] + 2[0.222(x_n + y_n) + 0.022] + [0.2444(x_n + y_n) + 0.0444]\} \\ &= 0.2214x_n + 1.2214y_n + 0.0214 \end{aligned}$$

(Note that, in general, we won't find an explicit formula like this.)
Thus we can compute the values:

n	xn	yn	0.2214xn+1.2214yn+0.0214	Exact value (e^x-x-1)	Error
0	0	0	0.0214	0	0
1	0.2	0.0214	0.09181796	0.021402758	0.0000028
2	0.4	0.09181796	0.222106456	0.091824698	0.0000067
3	0.6	0.222106456	0.425520826	0.2221188	0.0000123
4	0.8	0.425520826	0.718251137	0.425540928	0.0000201
5	1	0.718251137		0.718281828	0.0000307

For this simple example, we have the exact solution, so we can compare with those values and see that the error is very small.



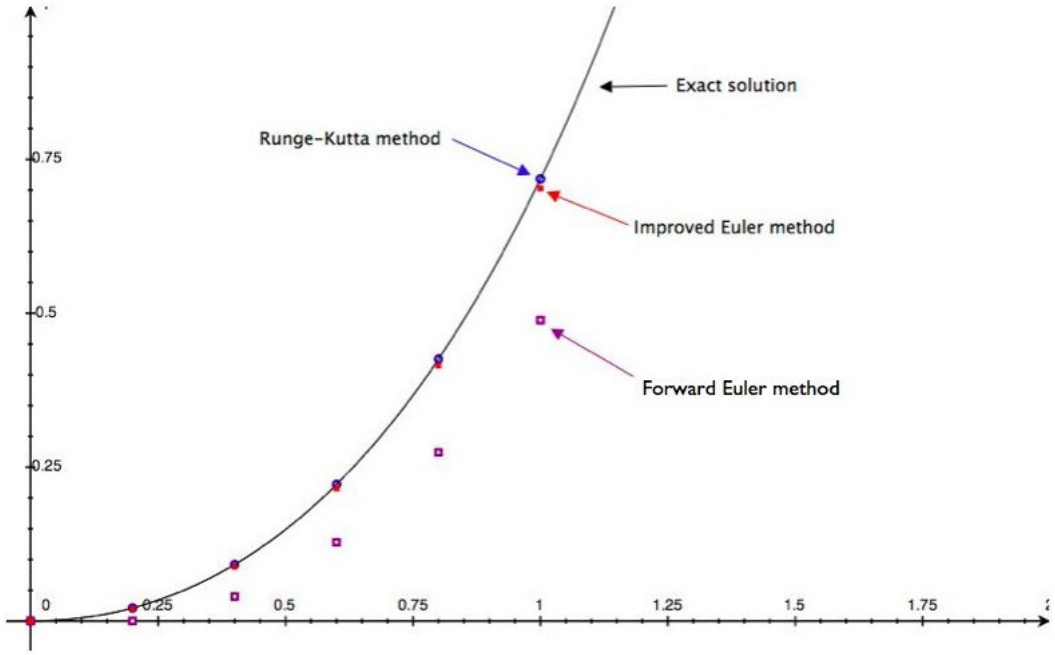
Exercise. Compute the next five values.

9.1 Error in the Runge-Kutta method

The error is much smaller than the other numerical ODE solvers we've seen. For the above example, we have the following values:

n	xn	Forward Euler	Improved Euler	Runge-Kutta	Exact value (e^x-x-1)	Forward Euler Error	Improved Euler Error	Runge-Kutta Error
0	0	0	0	0	0	0	0	0
1	0.2	0	0.02	0.0214	0.02140276	0.02140276	0.00140276	0.0000028
2	0.4	0.04	0.0884	0.09181796	0.0918247	0.0518247	0.0034247	0.0000067
3	0.6	0.128	0.2158	0.22210646	0.2221188	0.0941188	0.0063188	0.0000123
4	0.8	0.274	0.4153	0.42552083	0.42554093	0.15154093	0.01024093	0.0000201
5	1	0.489	0.7027	0.71825114	0.71828183	0.22928183	0.01558183	0.0000307

The errors in the forward and improved Euler are coarse enough to be visible with the eye.



Example. Solve the initial value problem

$$y' = (y - x - 1)^2 + 2, \qquad y(0) = 1.$$

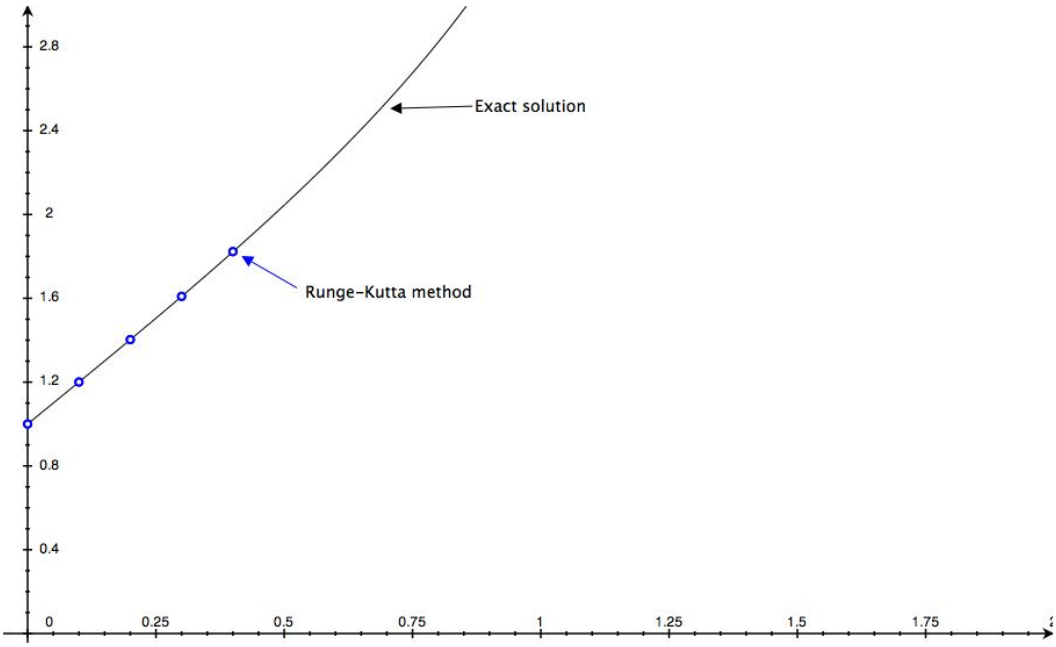
by the Runge-Kutta method for $0 \le x \le 0.4$ with step $h = 0.1$.

We have

$$\begin{aligned}
f(x,y) &= (y-x-1)^2 + 2 \\
k_1 &= hf(x_n, y_n) \\
&= 0.1(y_n - x_n - 1)^2 + 0.2 \\
k_2 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\
&= 0.1[(y_n + \frac{1}{2}k_1) - (x_n + \frac{1}{2}h) - 1]^2 + 0.2 \\
&= 0.1[(y_n + \frac{1}{2}(0.1(y_n - x_n - 1)^2 + 0.2)) - (x_n + \frac{1}{2}h) - 1]^2 + 0.2 \\
k_3 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right) \\
&= 0.1[(y_n + \frac{1}{2}k_2) - (x_n + \frac{1}{2}h) - 1]^2 + 0.2 \\
&= 0.1[(y_n + \frac{1}{2}(0.1[(y_n + \frac{1}{2}(0.1(y_n - x_n - 1)^2 + 0.2)) - (x_n + \frac{1}{2}h) - 1]^2 + 0.2)) - (x_n + \frac{1}{2}h) - 1]^2 + 0.2 \\
k_4 &= hf(x_n + h, y_n + k_3) \\
&= 0.1[(y_n + k_3) - (x_n + h) - 1]^2 + 0.2 \\
&= 0.1[(y_n + (0.1[(y_n + \frac{1}{2}(0.1[(y_n + \frac{1}{2}(0.1(y_n - x_n - 1)^2 + 0.2)) - (x_n + \frac{1}{2}h) - 1]^2 + 0.2)) \\
&\quad - (x_n + \frac{1}{2}h) - 1]^2 + 0.2)) - (x_n + h) - 1]^2 + 0.2
\end{aligned}$$

However, as we can see, these formulas are unweildy, so let's just use the k_i's directly in Excel.

n	xn	yn	k1	k2	k3	k4	y_{n+1}	Exact value	Error
			0.1*(yn-xn-1)^2+0.2	0.1*(yn+0.5*k1-xn-0.5*0.1-1)^2+0.2	0.1*(yn+0.5*k2-xn-0.5*0.1-1)^2+0.2	0.1*(yn+k3-xn-0.1-1)^2+0.2			
0	0	1	0.2	0.20025	0.200251252	0.201005031	1.200334589	1	0
1	0.1	1.200334589	0.201006703	0.202275208	0.202294383	0.20410585	1.402709878	1.200334672	0.0000001
2	0.2	1.402709878	0.204109129	0.206490492	0.206551303	0.209564248	1.609336039	1.402710036	0.0000002
3	0.3	1.609336039	0.209568879	0.213258372	0.213393055	0.217869989	1.822792993	1.60933625	0.0000002
4	0.4	1.822792993	0.217875391	0.223206446	0.223463969	0.229839667		1.822793219	0.0000002



The exact solution doesn't come from an explicit solution, but from using a numerical solver of the ODES (slightly more advanced that the Runge-Kutta method, so at least we aren't comparing two Runge-Kutta solutions here).