

Quick Introduction to Numerical Methods

For more details, see S17 short option.

Numbers vs numerals

A computer stores representations of numbers rather than numbers themselves.

- integers: one-to-one mapping up to limits. Then we might get a funny “overflow” behavior where, for instance, the sum of two large positive numbers gives a negative result.
- reals: where does the fraction begin? Rounding errors mean that the order of operations can be important (addition isn’t associative anymore). Loss of significance results from subtracting similarly large numbers.

Sometimes it is useful to rephrase formulae:

- nested form of polynomials:

$$x^3 - 6x^2 + 3x - 0.149 = [(x - 6)x + 3]x - 0.149$$

LHS has 4 multiplications and 3 additions. RHS has fewer (2) multiplications. This is faster, and because of fewer operations, there is less potential for loss of significance.

- quadratic formula: This is a problem if b^2 is much greater than $4ac$.
Use web example with $x^2 + 62.1x + 1 = 0$. Also can choose the example. Note that in these cases, neither formula produces two accurate answers; you have to choose the more accurate answer based upon where you add or subtract the discriminant.

Roots of nonlinear equations

Bisection method: use web example (also saw last week as a Matlab script). This is pretty brute force, but is guaranteed to get you an answer if there is one in the interval.

Faster: regula falsi (“false position”) method. Instead of choosing midpoint, use a linear interpolation: for a function $f(x)$ and current endpoints a and b , choose the trial point at

$$\frac{af(b) - bf(a)}{f(b) - f(a)}$$

and adjust endpoints accordingly.

Another possibility: instead of using the points as interval endpoints, use them as two successive approximations. Result is the secant method:

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

Stop when tolerance condition or maximal iteration condition is satisfied. Use web example.

This doesn't need a solution to be in the interval. On the other hand, it may find a solution nowhere close to the first two guesses.

But why use two guesses? Could also do it with a single point and the tangent. Take limit of secant method to get the Newton-Raphson method:

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

Again, stop when tolerance or maximal iteration condition satisfied. Use web example.

Advantage is that this method converges rapidly (though like secant, maybe nowhere close to initial guess). Disadvantage here is that you need to know the derivative, which may also be expensive to calculate. Roots of the derivative are also a problem.

So secant and Newton methods are often used to refine answers obtained by other techniques such as bisection.

Solving differential equations

This is a huge class of problems and methods. Even just among practical scripts, you find these in CO14, CO51, CO52, CO54, CO56. This is the majority of computing practicals in Prelims.

Basic methods start from the Taylor series:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2!}f''(x_0) + \dots + \frac{(x - x_0)^n}{n!}f^{(n)}(x_0) + \dots$$

(This reminds me of a professor from an earlier generation telling me that most physics is contained in the expression $e^x \approx 1 + x$.)

One can derive approximate derivative expressions from this: let $f_0 = f(x_0)$, $f_1 = f(x_0 + h)$, $f_{-1} = f(x_0 - h)$, etc. Then

$$\begin{aligned} f_1 &= f_0 + hf'_0 + \frac{h^2}{2!}f''_0 + \frac{h^3}{3!}f'''_0 + \dots \\ f_{-1} &= f_0 - hf'_0 + \frac{h^2}{2!}f''_0 - \frac{h^3}{3!}f'''_0 + \dots \end{aligned}$$

from which one gets

$$\begin{aligned} f'_0 &= \frac{f_1 - f_0}{h} + O(h) \\ f'_0 &= \frac{f_0 - f_{-1}}{h} + O(h) \\ f'_0 &= \frac{f_1 - f_{-1}}{2h} + O(h^2) \\ f''_0 &= \frac{f_1 - 2f_0 + f_{-1}}{h^2} + O(h^2) \end{aligned}$$

Now, one could substitute these expressions immediately into a number of differential equations and solve for one step in terms of previous steps. In fact, this is often done for partial differential equations.

But we'll also take a different approach which in a sense uses the statement of the problem to greater effect.

Consider a first-order ordinary differential equation with an initial condition:

$$\begin{aligned}y' &= f(x, y) \\ y(x_0) &= y_0\end{aligned}$$

Note that higher-order ODE's can be rephrased as a system of first-order differential equations simply by defining the derivatives of y as new variables. Numerical methods can then be extended to these systems in obvious (but somewhat tedious) ways.

From the Taylor theorem, we have an approximation

$$y(x_k + h) \approx y(x_k) + hy'(x_k)$$

but $y'(x_k) = f(x_k, y_k)$, so we plug it in (using y_k as a previously obtained approximation to the solution):

$$y(x_k + h) \approx y(x_k) + hf(x_k, y_k)$$

with the starting point $y(x_0) = y_0$. This is known as Euler's method.

We can extend this by performing a numerical integration on f . Here we use the simplest, the "trapezium" rule:

$$\begin{aligned}y(x_{k+1}) - y(x_k) &= \int_{x_k}^{x_{k+1}} f(x, y) dx \\ &\approx \frac{h}{2} (f(x_k, y(x_k)) + f(x_{k+1}, y(x_{k+1})))\end{aligned}$$

This is an implicit equation because $y(x_{k+1})$, the unknown, appears on both sides of the equation and there's no way to solve it. But we can use the Euler method as a first approximation:

$$y_{k+1} = y_k + \frac{h}{2} [f(x_k, y_k) + f(x_{k+1}, y_k + hf(x_k, y_k))]$$

which is a little easier to read if divided up into several steps:

$$\begin{aligned}m_1 &= hf(x_k, y_k) \\ m_2 &= hf(x_{k+1}, y_k + m_1) \\ y_{k+1} &= y_k + \frac{1}{2}(m_1 + m_2)\end{aligned}$$

In pictures, you can imagine that you first make a linear approximation to estimate y' , and then use that to evaluate the next step.

This is called the “Improved Euler” method. It is also known as a Runge-Kutta method of order 2, and is an example of something called a “predictor-corrector” method.

A popular method is the Runge-Kutta method of order 4:

$$\begin{aligned}m_1 &= hf(x_k, y_k) \\m_2 &= hf\left(x_k + \frac{h}{2}, y_k + \frac{1}{2}m_1\right) \\m_3 &= hf\left(x_k + \frac{h}{2}, y_k + \frac{1}{2}m_2\right) \\m_4 &= hf(x_k + h, y_k + m_3) \\y_{k+1} &= y_k + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)\end{aligned}$$

In steps:

1. m_1 is the first full estimate.
2. m_2 is a first midpoint estimate.
3. m_3 is a correction on the midpoint estimate.
4. m_4 is a correction on the full estimate.
5. and then you have the weighted average of the predictions.

This is a good balance of speed (4 f evaluations) and accuracy and stability. This just needs one starting point, and each step only depends on the previous one. So it’s called “self-starting” and “one-step”.

There are more advanced methods (the one labelled “predictor-corrector” among the web examples is a very nice one combining the Adams-Bashforth and Adams-Moulton methods), and they can be faster and more stable, but they often require more starting points. If these aren’t available, Runge-Kutta of a suitable order is often used to start the whole thing off.