Introduction to Numerical Methods

Jemma Shipton, John Thuburn and Beth Wingate
University of Exeter

Comments and corrections to: j.shipton@exeter.ac.uk

This module Introduction to Numerical Methods comprises roughly 9 hours of lectures and 9 hours of computing laboratory practicals. In addition there will be some exercises for you to try in your own time. The module is not assessed.

Timetable

Lectures and computing practicals	Mon	31 October	9:30-13:30
	Mon	7 November	9:30-13:30
	Mon	14 November	9:30-13:30
	Mon	21 November	9:30-13:30
	Mon	5 December	9:30-13:30
	Mon	12 December	9:30-13:30

Note that there is NO SESSION on Monday 28th November.

Week 1 Introduction; Taylor series; root finding; interpolation

Week 2 Numerical integration; numerical differentiation; time stepping schemes

Week 3 Diffusion equation; advection equation; spectral methods and finite element methods

Week 4 Gravity waves: staggered grids; semi-implicit time stepping

Week 5 Nonlinear problems: Burgers equation; conservation; finite volume methods

Week 6 Introduction to compatible finite element methods

Introduction

The mathematical equations used to model real world phenomena like the weather are usually too difficult to solve analytically; they can only be solved using methods of approximation. These methods of approximation usually involve vast amounts of arithmetic so they are carried out on computers. This module introduces some of the key ideas underlying these methods of approximation or "numerical methods".

1 Accuracy

We obviously want numerical solutions to be as accurate as possible, but what we judge as 'accurate' might depend on the problem being solved. For example, we might want to

- minimize the root mean square error in some field;
- avoid negative values of water vapour or chemical concentration
- satisfy certain conservation properties (e.g. mass, momentum, energy, etc.);
- capture certain essential properties of the physical system such as geostrophic balance;
- etc.

2 Taylor series

Many numerical methods are based on Taylor series, and Taylor series are fundamental to analysing the accuracy and other properties of numerical methods.
If the function f is infinitely differentiable at x then values of f in some neighbourhood of x are given by

$$f(x+\varepsilon) = f(x) + \varepsilon f'(x) + \frac{\varepsilon^2}{2} f''(x) + ... \quad (1)$$

$$= \sum_{k=0}^{k=n} \frac{\varepsilon^k}{k!} f^k(x) + O(\varepsilon^{n+1}) \quad (2)$$

($g = O(\varepsilon^p)$ means g/ε^p remains bounded as $\varepsilon \to 0$.)

3 Finding roots of nonlinear equations

Suppose we wish to solve $f(x) = 0$ for some scalar function f of a scalar variable x. A sequence of approximate solutions $\left\{x^{(m)}; m = 1, 2, ...\right\}$ *converges* to the true solution x^* if

$$x^{(m)} \to x^* \quad \text{as} \quad m \to \infty. \tag{3}$$

3.1 Fixed point iteration

Rewrite $f(x) = 0$ in the form $g(x) = x$.
Iterate: $x^{(m+1)} = g(x^{(m)})$.
This converges provided g satisfies the Lipschitz condition $|g(x) - g(y)| \leq L\,|x - y|$ for some constant $L < 1$ and for all x and y in the domain of interest.

3.2 Bracketing methods

3.2.1 Bisection

Suppose we have $x^{(l)}$, $x^{(r)}$ with $f(x^{(l)})$ and $f(x^{(r)})$ of opposite sign. Let

$$x^{(n)} = \frac{1}{2}\left(x^{(l)} + x^{(r)}\right). \tag{4}$$

Retain whichever of $f(x^{(l)})$ and $f(x^{(r)})$ has the opposite sign to $f(x^{(n)})$ and discard the other; rename as $x^{(l)}$, $x^{(r)}$, etc., and iterate.
This method is robust but slow to converge: the bracket size is halved at each iteration.

3.2.2 Secant method

Suppose we have $x^{(m-1)}$, $x^{(m)}$ and the corresponding $f(x^{(m-1)})$ and $f(x^{(m)})$. Let

$$x^{(m+1)} = \frac{x^{(m-1)} f(x^{(m)}) - x^{(m)} f(x^{(m-1)})}{f(x^{(m)}) - f(x^{(m-1)})}. \tag{5}$$

Then discard $x^{(m-1)}$, $f(x^{(m-1)})$ and iterate.
This usually converges much quicker than the bisection method, but can behave badly if $f(x^{(m)}) - f(x^{(m-1)})$ becomes small. It is not really a bracketing method as the root is not guaranteed to lie between $x^{(m)}$ and $x^{(m+1)}$.

3.2.3 Regula falsi

Like the bisection method, regula falsi is a bracketing method. However, the new value $x^{(n)}$ is constructed as in the secant method:

$$x^{(n)} = \frac{x^{(l)} f(x^{(r)}) - x^{(r)} f(x^{(l)})}{f(x^{(r)}) - f(x^{(l)})}. \tag{6}$$

Again, we retain whichever of $f(x^{(l)})$ and $f(x^{(r)})$ has the opposite sign to $f(x^{(n)})$ and discard the other; rename as $x^{(l)}$, $x^{(r)}$, etc., and iterate. This ensures that the two retained values of x always bracket the root, making this method more robust than the secant method. However, convergence is usually somewhat slower.

3.3 Newton-Rapheson method

Suppose we are able to compute the first derivative $f'(x)$ as well as $f(x)$. Given $x^{(m)}$, $f(x^{(m)})$ and $f'(x^{(m)})$, let

$$x^{(m+1)} = x^{(m)} - \frac{f(x^{(m)})}{f'(x^{(m)})} \tag{7}$$

and iterate.

4 Representing continuum data

Atmospheric fields such as velocity and temperature can (for almost all practical purposes) be regarded as defined over a continuous range of space and time. In this sense we would, in principle, need an infinite amount of information to represent such a field. In practice, finite computing power and computer memory mean we must represent continuum data by a finite set of values. There are various ways to do this, including

- a set of point data values: **the grid point representation**;
- a series expansion $f(x) = \sum_{k=1}^{N} a_k \phi_k(x)$ for some set of known basis functions $\phi_k(x)$: the ϕs may be locally nonzero, such as piecewise constants or hat functions—then we have a **finite element representation**; or they may be global function such as sines and cosines—then we have a **spectral representation**;
- **finite volume representation**, in which the data values represent averages over grid cells of the quantity concerned, rather than point values.

Truncation errors arise when we represent continuous data by a finite set of values, and when we approximate operations like derivatives by inexact expressions such as finite differences. They are nearly always more important than *roundoff errors*.

5 Interpolation

Suppose we know the values of a function $f(x)$ at the set of grid points $x_k,\ k = 1,\ 2,\ \ldots$

$$f_k = f(x_k).$$

We can estimate the value of f at a point x not necessarily on the grid by interpolation. Suppose x lies between x_k and x_{k+1}.

5.1 Linear interpolation

The equation of the straight line through (x_k, f_k) and (x_{k+1}, f_{k+1}) is

$$\widehat{f}(x) = \left(\frac{x - x_{k+1}}{x_k - x_{k+1}}\right) f_k + \left(\frac{x - x_k}{x_{k+1} - x_k}\right) f_{k+1}.$$

This may be written compactly as

$$\widehat{f}(x) = (1 - \beta) f_k + \beta f_{k+1}$$

where $\beta = (x - x_k)/(x_{k+1} - x_k)$.
It can be shown that

$$\widehat{f}(x) = f(x) + O(\Delta x^2),$$

that is, $\widehat{f}(x)$ is a second order accurate estimate of the true value $f(x)$.

5.2 Lagrange interpolation of arbitrary order

Linear interpolation is not accurate enough for some purposes, particularly for semi-Lagrangian advection. More accurate interpolation is possible using a bigger *stencil* of data points.
The equation of the polynomial of degree $N - 1$ through the N points (x_1, f_1), (x_2, f_2), ..., (x_N, f_N) is

$$\begin{aligned}
\widehat{f}(x) &= \frac{(x - x_2)(x - x_3)\ldots(x - x_N)}{(x_1 - x_2)(x_1 - x_3)\ldots(x_1 - x_N)} f_1 \\
&+ \frac{(x - x_1)(x - x_3)\ldots(x - x_N)}{(x_2 - x_1)(x_2 - x_3)\ldots(x_2 - x_N)} f_2 \\
&+ \ldots \\
&+ \frac{(x - x_1)(x - x_2)\ldots(x - x_{N-1})}{(x_N - x_1)(x_N - x_2)\ldots(x_N - x_{N-1})} f_N.
\end{aligned}$$

This gives an N^{th} order accurate interpolation formula.
The linear interpolation formula given above is the special case with $N = 2$.

5.3 Cubic Lagrange interpolation

Another special case is cubic lagrange interpolation ($N = 4$). If the grid spacing Δx is constant and x lies between x_k and x_{k+1} then

$$\begin{aligned}\widehat{f}(x) &= -\frac{1}{6}\beta(1-\beta)(2-\beta)f_{k-1} + \frac{1}{2}(1+\beta)(1-\beta)(2-\beta)f_k \\ &\quad +\frac{1}{2}(1+\beta)\beta(2-\beta)f_{k+1} - \frac{1}{6}(1+\beta)\beta(1-\beta)f_{k+2}\end{aligned}$$

where $\beta = (x - x_k)/(x_{k+1} - x_k) = (x - x_k)/\Delta x$. This is fourth order accurate.

5.4 Hermite cubic interpolation

Lagrange interpolation is continuous from one grid interval to the next, since the interpolating polynomials always pass through the data values at the ends of each grid interval. However, the derivatives of the interpolating polynomials are not necessarily continuous. We can obtain an interpolation that is is continuous and has continuous derivatives using Hermite cubic interpolation.
Suppose we know both $f(x)$ and its derivative $f'(x)$ at the grid points x_k, $k = 1, 2, \ldots$

$$f_k = f(x_k), \quad f'_k = f'(x_k).$$

The cubic polynomial that passes through (x_k, f_k) and (x_{k+1}, f_{k+1}) with derivative f'_k at x_k and derivative f'_{k+1} at x_{k+1} is

$$\begin{aligned}\widehat{f}(x) &= \frac{(x-x_k)^2}{(x_{k+1}-x_k)^2}\left[\left(f'_{k+1} - \frac{2f_{k+1}}{(x_{k+1}-x_k)}\right)(x-x_{k+1}) + f_{k+1}\right] \\ &+ \frac{(x-x_{k+1})^2}{(x_{k+1}-x_k)^2}\left[\left(f'_k - \frac{2f_k}{(x_k-x_{k+1})}\right)(x-x_k) + f_k\right].\end{aligned}$$

If we know the values of f'_k as well as those of f_k then we can use this formula directly. If we only know the values of f_k then we can estimate the derivatives, for example using one of the finite difference formulas given in section **??**, before applying the above formula.

5.5 Cubic spline interpolation

We can construct a piecewise cubic interpolation formula, known as a *cubic spline*, that satifies $\widehat{f}(x_k) = f_k$ for all k and also has continuous first and second derivatives everywhere. The interpolation formula is non-local: the cubic fit in any interval is affected by *all* the data values. However, the coefficients of the cubics can be

found efficiently by solving a tridiagonal linear system, which requires order N operations for N data values. See, e.g. *Numerical Recipes* for more details.
Like cubic Lagrange interpolation, cubic splines are fourth order accurate. But they have a smaller constant in front of the leading error term, and so are a bit more accurate in practice.