Lecture 5: Finite differences 1

Sourendu Gupta

TIFR Graduate School

Computational Physics 1
February 17, 2010

Outline

The generic function is pathological

The Bolzano-Weierstrass function is defined for any odd positive integer b as

$$f(x) = \sum_{n=0}^{\infty} a^n \cos \pi b^n x, \qquad 0 < a < 1, \quad ab > 1 + 3\pi/2.$$

This was the first known example of a real function which is continuous at all points, but has no derivative anywhere. Functions of this kind are generic. The trajectory of Brownian motion is of this form. Path integrals in any non-trivial field theory are dominated by such paths: they are the essence of quantum theory.

Problem 1: What can you say about the accuracy of numerical computations of the BW function and its derivative from the series definition above? Assume that the machine precision is p. What is the coarsest grid on which you can tabulate the BW function and still get close to machine precision evaluation of the function through a polynomial interpolation?

Non-Newtonian machines

The **forward, backward and central differences** are

$$\Delta_h f = \frac{f(x+h)-f(x)}{h},$$
$$\nabla_h f = \frac{f(x)-f(x-h)}{h},$$
$$D_h f = \frac{f(x+h)-f(x-h)}{2h},$$

all have the same limit (over reals) as $h \longrightarrow 0$, and it is the derivative, $f'(x)$. The limit has no meaning for floating point numbers: as h becomes smaller and smaller, the differences in the numerator become smaller and catastrophic loss of significance sets in.

0.000122070312	0.540527344	0.540039062
6.10351562E-05	0.541015625	0.540039062
3.05175781E-05	0.541015625	0.5390625
1.52587891E-05	0.54296875	0.5390625
7.62939453E-06	0.546875	0.5390625
3.81469727E-06	0.546875	0.53125
1.90734863E-06	0.5625	0.53125
9.53674316E-07	0.5625	0.5
4.76837158E-07	0.625	0.5
2.38418579E-07	0.75	0.5
1.19209290E-07	1.	0.5
5.96046448E-08	1.	0.
2.98023224E-08	2.	0.
1.49011612E-08	4.	0.
7.45058060E-09	8.	0.
3.72529030E-09	16.	0.

h, $\Delta_h \sin(1.0)$ and $\nabla_h \sin(1.0)$ in single precision arithmetic.

Instability and inaccuracy

A Taylor series expansion shows that

$$\Delta_h f(x) = f'(x) + hf''(c), \qquad \text{where } x \le c \le x + h,$$

so for large h there is an **inaccuracy**. For small h one has to also take care of numerical inaccuracy. If ϵ_m is the machine precision, then

$$\underline{\Delta_h f(x)} = f'(x) + \frac{\epsilon_m}{h} + hf''(c).$$

Here (and later) a bar under a quantity will denote the result of a numerical computation. The $1/h$ term is a **numerical instability**.

> **Problem 2**: Perform the analysis of errors and instabilities for ∇_h and D_h. Write FORTRAN, f90 and C codes for computing all three difference coefficients for $\sin x$ and check whether these formulæ are accurate, *i.e.*, whether ϵ_m can be extracted from the behaviour of the differences. Can one extract the exact value of $f'(x)$ using computations at different h?

Outline

An impossible thing before breakfast

Given a set of knot points, $\{x_0, x_1, \cdots, x_N\}$ (assume that the points are ordered $x_0 < x_1 < \cdots < x_N$), and a set of function values at these points, $\{y_0, y_1, \cdots, y_N\}$, how do you find the function value at any point in the range $[x_0, x_N]$?
If you find an answer, $f(x)$, then I can always construct a new function

which coincides with $f(x)$ at every tabulated value and can be made to differ from $f(x)$ everywhere else by as much as we wish. Therefore, this is an ill-posed question.
We try to make the answer unique by asking how to construct polynomials which pass through the tabulated points. We can, of course, also ask that these polynomials satisfy some other properties, like smoothness, etc. We could also ask for rational functions, Fourier series, or any other well-posed question.

An impossible thing before breakfast

Given a set of knot points, $\{x_0, x_1, \cdots, x_N\}$ (assume that the points are ordered $x_0 < x_1 < \cdots < x_N$), and a set of function values at these points, $\{y_0, y_1, \cdots, y_N\}$, how do you find the function value at any point in the range $[x_0, x_N]$?

If you find an answer, $f(x)$, then I can always construct a new function

$$f(x) + g(x) \prod_{i=0}^{N} (x - x_i)$$

which coincides with $f(x)$ at every tabulated value and can be made to differ from $f(x)$ everywhere else by as much as we wish. Therefore, this is an ill-posed question.

We try to make the answer unique by asking how to construct polynomials which pass through the tabulated points. We can, of course, also ask that these polynomials satisfy some other properties, like smoothness, etc. We could also ask for rational functions, Fourier series, or any other well-posed question.

Polynomial fits and van der Monde matrices

Typically, you could find a polynomial of order N which passes through $N+1$ points—

$$P(x) = \sum_{i=0}^{N} c_i x^i, \qquad \text{where} \qquad P(x_i) = y_i \quad (0 \le i \le N).$$

This gives a set of linear equations to solve—

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^N \\ 1 & x_1 & \cdots & x_1^N \\ \vdots & \vdots & & \vdots \\ 1 & x_N & \cdots & x_N^N \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix}.$$

This $(N+1) \times (N+1)$ matrix is called a van der Monde matrix. It has very small determinant: typically $\det M \propto \exp[-\mathcal{O}(N^p)]$ where $p > 1$. As a result, inverting it by the usual methods leads to catastrophic loss of precision. So look for more clever methods.

Newton polynomials

For a given set of knot points, the Newton polynomials are, $P_0(x) = 1$ and

$$P_n(x) = \prod_{i=0}^{n}(x - x_i). \qquad \text{Note} \qquad P_n(x_\alpha) = 0 \quad \text{when } \alpha \leq n.$$

The Newton polynomial approximation is the linear combination

$$P(x) = \sum_{n=0}^{N} a_n P_n(x), \qquad \text{where} \qquad P(x_\alpha) = y_\alpha \text{ for } 0 \leq \alpha \leq N.$$

Solve this by inverting the matrix

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & x_1 - x_0 & \cdots & \prod_{n=0}^{N-1}(x_N - x_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix}.$$

Problem 3: Plot the Newton polynomials for a randomly chosen set of grid points. Bring the van der Monde matrix to the above form.

Divided differences

Solutions are the divided differences. Solve recursively starting from equation for a_0:

$$a_0 = y_0, \quad a_1 = \frac{y_1 - y_0}{x_1 - x_0}, \quad a_2 = \sum_{j=0}^{2} \frac{y_j}{\prod_{n=0,2|n\neq j}(x_j - x_n)} \cdots .$$

One usually displays this as the tableau:

$$\begin{array}{lllll} x_0 & y_0 & y_0^{(1)} & y_0^{(2)} & y_0^{(3)} \quad \cdots \\ x_1 & y_1 & y_1^{(1)} & y_1^{(2)} & \cdots \\ x_2 & y_2 & y_2^{(1)} & \cdots & \\ x_3 & y_3 & \cdots & & \end{array}$$

The elements of the tableau can be generated by the recursive formulæ

$$x_i^{(l+1)} = x_{i+1}^{(l)} - x_i^{(l)}, \qquad y_i^{(l+1)} = \left(y_{i+1}^{(l)} - y_i^{(l)}\right) / x_i^{(l+1)},$$

starting with $x_i^{(0)} = x_i$ and $y_i^{(0)} = y_i$.

A few problems

Problem 4: Show that when the knot points, x_i, approach each other, the Newton polynomial approximation approaches a truncated Taylor series. If these computations are performed in floating point arithmetic, is the limit well-defined? Given a certain machine precision, is there a limit to how closely the x_is can be spaced before catastrophic loss of precision renders the divided differences meaningless?

Problem 5: What relation does the Newton polynomial approximation bear to the Lagrange interpolation formula

$$P(t) = \sum_{j=0}^{N} y_j \frac{\prod_{n=0,N|n\neq j}(t - x_n)}{\prod_{n=0,N|n\neq j}(x_j - x_n)}?$$

Is there any way to make this approximation behave better than the divided differences at any fixed value of machine precision?

Problem 6: Are the derivative of the Lagrange interpolation formula continuous at the knot points?

Outline

Solving a finite difference equation

The equation $\Delta_h f = \alpha f$ gives

$$f(t+h) = (1+\alpha h) f(t) = (1+\alpha h)^{1+t/h} f(0).$$

In the limit of $h \to 0$, this is indeed the exponential function.
With the approximation Δ for the derivative, we neglect terms of order h^2.
The solution of the differential equation $f'(t) = \alpha f(t)$ would give

$$f(t+h) = \mathrm{e}^{h\alpha} f(t) \simeq \left[1 + h\alpha + \mathcal{O}(h^2)\right] f(t).$$

So, the solution of the difference equation is correct to the same order.
This will be generalized to **Euler's method**. Improvements can be obtained by solving the equation

$$\Delta_h f = \frac{1}{h}\left(\mathrm{e}^{h\alpha} - 1\right) f.$$

It is common to improve the equation by placing correction terms into the right hand side. It is also possible to use expressions for the derivative which are correct to higher order in h.

Instability and inaccuracy

The solution of the differential equation $f'(t) = -\alpha f(t)$ goes to zero in the limit $t \to \infty$ for all positive values of α and generic initial conditions. The equation $\Delta_h f = -\alpha f$ gives

$$f(t+h) = (1-\alpha h) f(t) = (1-\alpha h)^{1+t/h} f(0).$$

If $\alpha h > 2$ then $|f(t)| \to \infty$ in the limit $t \to \infty$. In other words, the solution is unstable.

In a region around $h = 2/\alpha$, one may have catastrophic loss of significance. Even if the loss per step is not catastrophic, errors could build up over many steps. In general, if the error at the n-th step is ϵ_n, one will have $|\epsilon_n| = K^n |\epsilon_1|$, where K is the error in evaluating the multiplicative factor. This is large also when αh is small.

If $1 - \alpha h$ were evaluated exactly, then an error made for some reason at any step of the computation propagates unchanged, *i.e.*, $K = 1$. Any error in computing this multiplicative factor increases K. Hence, the error grows exponentially, irrespective of the sign of α.

Stabilizing the solution

One way to stabilize the solution is to replace the difference equation by $\Delta_h f(t) = -\alpha f(t+h)$. Then

$$f(t+h) = \frac{1}{1+\alpha h} f(t) = \frac{1}{(1+\alpha h)^{1+t/h}} f(0).$$

This goes to zero in the limit of large t. Since the solution requires the evaluation of the derivative at the next step of the solution, a method such as this is called an **implicit method**.
Another stable solution is obtained by using the difference equation $\Delta_h f(t) = -\alpha[f(t) + f(t+h)]/2$. This has the solution

$$f(t+h) = \frac{1-\alpha h/2}{1+\alpha h/2} f(t) = \left(\frac{1-\alpha h}{1+\alpha h}\right)^{t/h} f(0).$$

Not only is this stable, one can also check that this is correct to $\mathcal{O}(h^2)$. Evaluating the derivative at the mid-point of the interval is a technique will be used in the **2nd order Runge-Kutta method**.

An oscillator: instabilities

The system of equations

$$\frac{d\mathbf{f}}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{f}, \quad \mathbf{f}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

has the solution $f_1(t) = \sin t$ and $f_2(t) = \cos t$. The corresponding forward difference equations, $\Delta_h \mathbf{f}(t) = M\mathbf{f}(t)$, where M is the 2×2 matrix above, have the solutions

$$\mathbf{f}(t+h) = \begin{pmatrix} 1 & h \\ -h & 1 \end{pmatrix} \mathbf{f}(t).$$

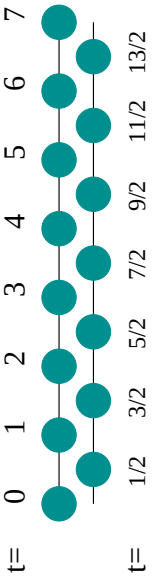
This matrix has eigenvalues $\lambda_\pm = 1 \pm ih$. Since $|\lambda_\pm| > 1$, errors in the computed solution will always grow.

Replace this by the implicit equation $\Delta_h \mathbf{f}(t) = M\mathbf{f}(t+h)$. Then the solution is

$$\mathbf{f}(t+h) = \begin{pmatrix} 1 & -h \\ h & 1 \end{pmatrix}^{-1} \mathbf{f}(t).$$

This matrix has eigenvalues $|\lambda_\pm| < 1$, so the solutions will always decay.

The leap-frog method



The oscillator should have a conserved quantity $|\mathbf{f}|^2$. In these methods $|\mathbf{f}(t+h)|^2 = |\lambda_\pm|^2|\mathbf{f}(t)|^2 = |\mathbf{f}(t)|^2 + \mathcal{O}(h^2)$. If one improves the conservation law, one may have a better algorithm.

Introduce a **staggered lattice** as here. Place f_1 at integer points nh and f_2 at half-integers $(n+1/2)h$. Place the f_1 on the integer lattice and the f_2 at the half integer points. Then the solutions of the difference equations are—

$$f_1(t+h) = f_1(t) + hf_2(t+\frac{h}{2}), \quad f_2(t+\frac{3h}{2}) = f_2(t+\frac{h}{2}) - hf_1(t+h).$$

The initial and final half steps are: $f_2(h/2) = f_2(0) - hf_1(0)/2$ and $f_2(nh) = f_2(nh - h/2) - hf_1(nh)/2$. Clearly this is equivalent to

$$f_1(t+h) = f_1(t) + hf_2(t) - \frac{h^2}{2}f_1(t), \quad f_2(t+h) = f_2(t) - \frac{h}{2}[f_1(t) + f_1(t+h)].$$

Therefore $|\mathbf{f}(t+h)|^2 = |\mathbf{f}(t)|^2 + \mathcal{O}(h^4)$.

Using the Euler and leapfrog methods

Problem 7: Write **Mathematica** functions which will take as input the vector $\mathbf{f}(t)$ and the quantity h and give back (1) the Euler updated vector, (2) the implicit Euler updated vector, and (3) the leap-frog updated vector, $\mathbf{f}(t+h)$. In each case use the function to evolve the vector $(0,1)^T$ through time 4π using $h = 2\pi/10$, $2\pi/20$, $2\pi/50$ and $2\pi/100$. Plot the phase space trajectories, and write out the initial and final values of $|\mathbf{f}|^2$ in each case.

Problem 8: Generalize the leap-frog method to solve the problem of the anharmonic oscillator, *i.e.*,

$$H = \frac{1}{2}p^2 + \frac{1}{4!}q^4.$$

Implement your solution in **Mathematica** and compare it with the results obtained using the inbuilt methods for solving differential equations. Change the precision of the Mathematica routines to see how this affects the solution.

Outline

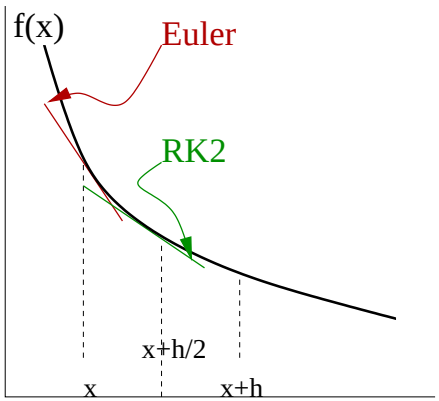
Solving differential equations

Euler's method of solving a system of first order differential equations, $\mathbf{f}'(x) = \mathbf{d}(\mathbf{f}(x), x)$, is

$$\mathbf{f}(x+h) = \mathbf{f}(x) + h\mathbf{d}(\mathbf{f}(x), x),$$

where $f_i(x)$ is the $i-th$ function. This is equivalent to solving the difference equation $\Delta_h \mathbf{f}(x) = \mathbf{d}(\mathbf{f}(x), x)$. The solution is correct to order h. The **second-order Runge-Kutta** algorithm (RK2) is defined by

$$\mathbf{f}(x+h) = \mathbf{f}(x) + h\mathbf{d}(\mathbf{f}(x+h/2), x+h/2), \quad \mathbf{f}(x+h/2) = \mathbf{f}(x) + \frac{h}{2}\mathbf{d}(f(x), x),$$



This is correct to $\mathcal{O}(h^2)$. One can also think of this as Euler's method on a **staggered lattice**. This corrects the right hand side of the equation by interpolating the derivative linearly between x and $x+h$. A quadratic interpolation gives rise to the **fourth order Runge-Kutta** algorithm.

Complexity of algorithms

Suppose that the dimension of the vectors **f** and **d** is M. Suppose also that the CPU time taken to perform an arithmetic operation is T_1, and that required for the evaluation of each component of **d** is T_2. It may happen that $T_2 \gg T_1$.
Then the time taken per step of the Euler method is $M(2T_1 + T_2)$, and the time per step in RK2 is exactly twice, *i.e.*, $2M(2T_1 + T_2)$. If the ODEs are integrated from x_i to x_f in N steps, *i.e.*, $h = (x_f - x_1)/N$, then the time taken for the Euler method is $MN(2T_1 + T_2)$, giving an error of $\mathcal{O}(N^{-2})$. RK2 takes exactly twice the time to get an error of $\mathcal{O}(N^{-3})$. A fair comparison would would be of the time taken to make comparable errors. To get an error of $\mathcal{O}(N^{-3})$ using Euler's method would require time of order $MN^{3/2}(2T_1 + T_2)$. We say that RK2 is faster because to get the same error, the time requirements are

$$T = \begin{cases} \mathcal{O}(N^{3/2}) & \text{for Euler's method,} \\ \mathcal{O}(N) & \text{for RK2.} \end{cases}$$

Example

Problem 9: The RK2 solution of the equations

$$\frac{dx}{dt} = v, \quad \frac{dv}{dt} = -x,$$

are

$$\begin{aligned}
x(nh + \frac{h}{2}) &= x(nh) + \frac{h}{2} v(nh) \\
v(nh + \frac{h}{2}) &= v(nh) - \frac{h}{2} x(nh) \\
x(nh + h) &= x(nh) + h v(nh + \frac{h}{2}) = \left(1 - \frac{h^2}{2}\right) x(nh) + \frac{h}{2} v(nh) \\
v(nh + h) &= v(nh) - h x(nh + \frac{h}{2}) = \left(1 - \frac{h^2}{2}\right) v(nh) - \frac{h}{2} x(nh).
\end{aligned}$$

Verify using `Mathematica` that this is indeed the correct solution of the equations upto and including the $\mathcal{O}(h^2)$ term.

A problem

Problem 10: Implement a leap-frog integration for the anharmonic oscillator

$$H = \frac{1}{2}p^2 + \frac{1}{4!}q^4$$

as a subroutine in f90 and c which is suitable for updating an ensemble of initial conditions; *i.e.*, it should take a set of (staggered) initial phase space positions (p_i, q_i) where $1 \leq i \leq N$ and a time step, h, and return the final phase space positions. Also implement a method for a half-step evolution of p_i in the beginning and end as a separate subroutine.

Take $N = 1000$ different initial conditions distributed uniformly within the phase space volume with $H \leq 1$. Find the density of points within the phase space volumes with $H \leq 1/4$, $1/2$ and 1. Evolve the systems using your program for trajectories of time duration 1, 2, 5 and 10 units. At the end of these times compute the same three phase space densities as initially. Are these conserved?

Outline

Recursive evaluation of the Lagrange interpolation formula

Let $P_{i:i+m}$ be the m-th order polynomial interpolating the function at x_i, x_{i+1}, $\cdots$, x_{i+m}. Then one has Neville's recurrence formula—

$$P_{i:i+m}(t) = \frac{1}{x_{i+m} - x_i} \begin{vmatrix} t - x_i & P_{i:i+m-1}(t) \\ t - x_{i+m} & P_{i+1:i+m}(t) \end{vmatrix}.$$

which starts from the constant polynomials $P_{i:i}(t) = y_i$.
If the error in $P_{i,i+1,\cdots i+m}$ is $\delta_{i,i+1,\cdots i+m}$, whose magnitude is bounded by δ, then the recurrence gives—

$$|\delta_{i,i+1,\cdots i+m}| \leq \delta \left(\left| \frac{x_{i+m} - x}{x_{i+m} - x_i} \right| + \left| \frac{x - x_i}{x_{i+m} - x_i} \right| \right) = \delta.$$

Therefore errors in the function values do not grow unless the knot points are so close that terms like $x_{i+m} - x_i$ are dominated by catastrophic loss of significance.
For a further improvement on this algorithm, see Numerical Recipes.

Neville's process as elimination

This can be seen as a process of elimination. Make the Taylor expansions of the function at x_0 and x_1—

$$\begin{aligned} f(t) &= f(x_0) + h_0 f'(x_0) + \frac{1}{2} h_0^2 f''(x_0) + \cdots \\ f(t) &= f(x_1) + h_1 f'(x_1) + \frac{1}{2} h_1^2 f''(x_1) + \cdots \end{aligned}$$

where $h_i = t - x_i$. Neville's process applied to these expansions gives

$$\begin{aligned} (h_1 - h_0) f(t) &= h_0 f(x_1) - h_1 f(x_0) + \left[h_0 h_1 \left\{ f'(x_1) + h_1 f''(x_1) + \cdots \right\} \right. \\ &\qquad \left. - h_0 h_1 \left\{ f'(x_0) + h_0 f''(x_0) + \cdots \right\} \right] + \cdots \\ &= h_0 f(x_1) - h_1 f(x_0) + \frac{h_0 h_1}{2} \left[h_0 f''(x_1) - h_1 f''(x_0) \right] + \cdots \end{aligned}$$

This removes the first derivative. What happens at the next step?

Is Lagrange interpolation a well-posed problem?

Problem 11: Write a Mathematica code which starts from a Taylor expansion of a function (and its derivatives) around some number of knot points. Apply successive steps of Neville's process to this and show that at each step of the process one more derivative is removed.

Problem 12: We have shown above that initial errors, δ, in the function values, y_i, do not grow when using Lagrange interpolation using Neville's recurrence. Check what happens to the Newton polynomials, divided differences, the Lagrange interpolation formula and Neville's recurrence when the knot point, x_i, are changed slightly. What happens when one of the knot-points is removed entirely? Is the behaviour equally stable (or unstable) for interpolation (*i.e.*, $x_0 \leq t \leq x_N$) and extrapolation ($t < x_0$ or $t > x_N$)?

Numerical integration

Problem 13: Given a polynomial approximation to tabulated data on functions, give expressions for integrating them. Note that the integration formulae should only involve the tabulated function values and constants. Read the Newton-Cotes integration formulae and check what kind of function interpolations they come from.

Problem 14: Consider integrals of functions with poles in the region of integration, such as

$$I = \int_{-\infty}^{\infty} dx \frac{P(x)}{x-1},$$

where $P(1) \neq 0$. These are often **regularized** using the Cauchy Principal Value prescription—

$$I = \lim_{\epsilon \to 0} \int_{-\infty}^{1-\epsilon} dx \frac{P(x)}{x-1} + \int_{1+\epsilon}^{\infty} dx \frac{P(x)}{x-1}.$$

Find an efficient method of evaluating the principal value. How does the value change if the integral is defined as the sum over the the two intervals $[-\infty, 1-5\epsilon]$ and $[1+\epsilon, \infty]$?

Smooth polynomials

Given knot points $\{x_0, x_1\}$ and function values $\{y_0, y_1\}$, one has a linear interpolation

$$g(t) = A(t)y_0 + B(t)y_1, \qquad \text{where} \qquad A(t) = \frac{t - x_1}{x_0 - x_1}, \quad B(t) = \frac{x_0 - t}{x_0 - x_1}.$$

The first derivative is constant in the interval (x_0, x_1). In the next interval, (x_1, x_2), the first derivative is again constant but a different value. As a result, the second derivative is discontinuous at the knot points.
If one had information on the second derivatives at the knot points, one could build an interpolating function

$$h(t) = A(t)y_0'' + B(t)y_1''.$$

Using this, one could try to make cubic interpolating function

$$f(t) = g(t) + (t - x_0)(t - x_1)h(t),$$

and arrange it to have continuous first derivatives at the knot points. This is the idea of a spline.

Cubic splines

In fact the first derivative is

$$f'(t) = g'(t)y_1 + (t - x_0 + t - x_1)h(t) + (t - x_0)(t - x_1)h'(t).$$

At the knot point x_1 one has the two expressions—

$$f'(x_1) = \frac{y_0 - y_1}{x_0 - x_1} - (x_0 - x_1)y_1'', \qquad f'(x_1) = \frac{y_1 - y_2}{x_1 - x_2} + (x_1 - x_2)y_1''.$$

Continuity of the derivative is obtained by equating these two expressions. This gives a solution for the unknown quantity—

$$y_1'' = \frac{1}{x_0 - x_2}\left[\frac{y_0 - y_1}{x_0 - x_1} - \frac{y_1 - y_2}{x_1 - x_2}\right].$$

One can eliminate y_i'' at every knot point except x_0 and x_N. At these two points one can give a boundary condition of one's choice.

A Fourier series

A periodic function $y(t) = y(t + L)$, has a compact representation as the Fourier series

$$y(t) = \sum_{k=0}^{N} f_k \exp\left(\frac{2\pi itk}{NL}\right).$$

The Fourier coefficients are given by the solution of a complex van der Monde matrix equation

$$\begin{pmatrix} 1 & z_0 & \cdots & z_0^N \\ 1 & z_1 & \cdots & z_1^N \\ \vdots & \vdots & & \vdots \\ 1 & z_N & \cdots & z_N^N \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix}.$$

where $z_j = \exp(2\pi i x_j/NL)$ and $|z_j| = 1$. If the knot points $x_j = jL$, then the z_js are roots of unity, and the sum over the elements of all but one row add up to zero. Since the rows orthogonal to each other, the matrix is invertible.

Fourier inversion

In fact, the definition of the Fourier transformation,

$$f_k = \frac{1}{N+1} \sum_{j=0}^{N} y_j \exp\left(-\frac{2\pi i x_j k}{NL}\right),$$

gives the inverse (z_j^* is the complex conjugate of z_j)

$$\frac{1}{N+1} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ z_0^* & z_1^* & \cdots & z_N^* \\ \vdots & \vdots & & \vdots \\ z_0^{*N} & z_1^{*N} & \cdots & z_N^{*N} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{pmatrix}.$$

Problem 15: Hadamard's bound states that for a real $(N+1) \times (N+1)$ matrix whose elements are bounded by unity, the determinant is at most $(N+1)^{(N+1)/2}$. What relation do the Fourier matrices hold to Hadamard matrices, *i.e.*, those real matrices which satisfy the bound?

Outline

References and further reading

1. Numerical Recipes, by W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Cambridge University Press. Treat this as a well written manual for a ready source of building blocks. However, be ready to go into the blocks to improve their performance.
2. E. H. Neville, "Iterative interpolation", *Journal of the Indian Mathematical Society*, Vol 20, p. 87 (1934).
3. Several points in this lecture are illustrated in the associated Mathematica notebooks. These notebooks also have additional examples. They are given in `tar.gz` format. Unpack them using the `tar -xvzf` command.