Physics 464/511 Lecture N Fall, 2015

1 Numerical Methods

Topics included in “Numerical Methods”, in particular Isaacson and Keller, “Analysis of Numerical Methods”

- Solutions of linear systems, matrix inversion
- Solution of nonlinear equations
- Eigenvalues and eigenvectors
- Polynomial approximation
- Differences
- Numerical integration
- Numerical solution of differential equations
  - Ordinary
  - Partial

I will discuss specific methods of numerical integration and ordinary differential equations. Obviously this is not a coherent coverage, even at a low level, of these fields.

1.1 Numerical Integration

One can clearly get an approximation to an integral $\int_0^L f(x)\,dx$ by evaluating the function at N points $x_i \in [0, L]$ and weighting the results,

$$I = \int_0^L f(x)\,dx \approx \bar{I} = \sum_{j=1}^N w_j f(x_j)$$

You probably know two methods — the trapezoid rule and Simpson's rule, which is

$$\int_0^L f(x)\,dx \approx \frac{L}{3(N-1)}\left[f(0) + 4\sum_{\text{odd } j=1}^{N-2} f\left(\frac{jL}{N-1}\right) + 2\sum_{\text{even } j=2}^{N-3} f\left(\frac{jL}{N-1}\right) + f(L)\right]$$

for odd N. Simpson's rule is clearly a bit more complex. Why is it better? How does one discuss the accuracy of an integration formula?

The evaluation of an integral numerically assumes continuity properties of the function, otherwise a finite number of points tells you nothing about the infinite number you didn't sample. Often, we know more than continuity — we know the function we are integrating is analytic, so it is differentiable as many times as we like.

Most integration formulas are designed to handle exactly polynomials up to a given order m called the **degree of precision**. The error will then be proportional to the $(m+1)$'st derivative of the function, and by dimensional analysis, to L^{m+2}.

Examples:[1]

Elementary trapezoid:

$$\int_0^h f(x)dx = \frac{h}{2}\left[f(0) + f(h)\right] - \frac{h^3}{12} f^{(2)}(\xi)$$

for some $\xi \in (0, h)$. Here $f^{(n)}$ means the n'th derivative of f, and $m = 1$.

Elementary Simpson:

$$\int_0^{2h} f(x)\,dx = \frac{h}{3}\left[f(0) + 4f(h) + f(2h)\right] - \frac{h^5}{90} f^{(4)}(\xi).$$

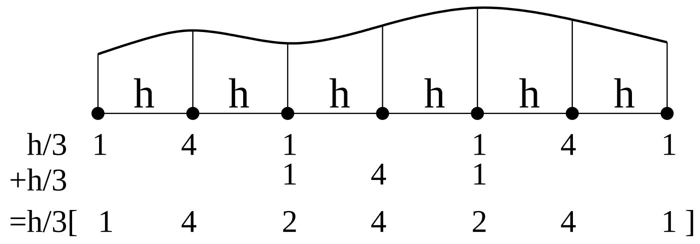
The last term is not evaluatible so we wish to make it small. The denominator helps. More importantly, if $h \ll$ the natural scale of variation of f, the extra factors of $h\frac{d}{dx}$ help also.

One way of making that small is to subdivide the interval into subintervals, each with a smaller h. The extended trapezoid and Simpson's rules

[1]These two formulas can be verified by expanding $f(x)$ in a Taylor series about the midpoint to the degree of precision and integrating.

mentioned before does just that, applying an elementary trapezoid or Simpson integration on neighboring intervals.

For Simpson:



Note by trisecting the interval, $h \to h/3$, so for less than 3 times the work I improve the accuracy by $3^5 = 243$ times. This subdivision is more effective the higher the degree of precision of the method.

One way to get methods of high degree of precision is to fit a polynomial of degree $N-1$ through the N points in question, and integrate the polynomial. Of course one doesn't want to do a fitting every time, but we may regard the fitting procedure as a linear function of the values $f(x_j)$, and then we can do the integration as a function of the $f(x_j)$ once, and it is then available to apply to an arbitrary (sufficiently differentiable) function.

If you apply this method to a set of evenly spaced intervals including the endpoints, you arrive at what is called **closed Newton-Cotes** formulas, of which the trapezoid and Simpson's rules are the two point and three point versions.

Because the N point Newton-Cotes formula fits a polynomial of order $N-1$ exactly, we see that the degree of precision is at least $N-1$. In fact it is N if N is odd, due to the symmetry.

The process of fitting the polynomial and integrating it, associated with such luminaries as Lagrange and Newton, is straightforward and dull. The formulas can be found in many places, *e.g.* Abramowitz and Stegun, page 886.

One can get an even better degree of precision if one is willing to take unequally spaced points. I will discuss **Gaussian integration**. For convenience, let the interval be $[-1, 1]$. This method is based on the ideas of orthogonal polynomials. Here we have $\int_{-1}^{1} f(x)w(x)\,dx$ with $w(x) = 1$, so the appropriate polynomials are Legendre.

We choose our N points x_j to be the zeros of $P_N(x)$. Then our integration

formula is

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{j=1}^{N} w_j f(x_j), \tag{1}$$

where the w_j need to be determined. First consider f to be fit by a polynomial g of order $N-1$ at the N points x_j: $f(x_j) = g(x_j)$. The N coefficients of g are well determined by these N conditions, and requiring that (1) be exact for all $N-1$'th order polynomials determines the N w_j's.

Now let f be fit by a polynomial h of order $2N-1$ at the N x_j's and N other points in $(-1, 1)$, so $f(x) = h(x)$ exactly at x_j and N other points, so $f(x) = h(x) + \mathcal{O}\left(f^{(2N)}(\xi)\right)$, and so $\int_{-1}^{1} f(x)\,dx = \int_{-1}^{1} h(x)\,dx$ to degree of precision $2N-1$.

As $h(x)$ is a polynomial of order $2N-1$, we can divide it by $P_N(x)$ to get

$$h(x) = q(x)P_N(x) + r(x)$$

where the quotient $q(x)$ and the remainder $r(x)$ are each polynomials of order $N-1$. Expand q in Legendre polynomials $q(x) = \sum_{\ell=0}^{N-1} a_\ell P_\ell(x)$, so

$$\int_{-1}^{1} h(x)\,dx = \sum_{0}^{N-1} a_\ell \underbrace{\int_{-1}^{1} P_\ell(x)P_N(x)\,dx}_{0} + \int_{-1}^{1} r(x)dx.$$

Now note that $h(x)$ and $g(x)$ and $f(x)$ all agree at the N points x_j, so

$$g(x_j) = h(x_j) = q(x_j)\underbrace{P_N(x_j)}_{0} + r(x_j), \qquad \text{or} \quad g(x_j) = r(x_j) \quad \text{at } N \text{ points } x_j.$$

g and r are each of order $N-1$ so $g(x) = r(x)$, and

$$\int_{-1}^{1} f(x)\,dx = \int_{-1}^{1} g(x)\,dx + \mathcal{O}f^{(2N)}(\xi) \approx \sum_{j=1}^{N} w_j f(x_j).$$

The simplest way to determine the w_j is to recognize that if $f = \left(\frac{P_N(x)}{x - x_j}\right)^2$, f is a polynomial of order $2N-2$, so the formula is exact for $\int f\,dx$. Notice that $f(x_k) = 0$ for $k \neq j$, while l'Hôpital tells us $f(x_j) = \left(P_N'(x_j)\right)^2$, so

$$\int_{-1}^{1} \left(\frac{P_N(x)}{x - x_j}\right)^2 = \left(P_N'(x_j)\right)^2 w_j.$$

This determines the weights, and shows they are all positive, which is an advantage in avoiding rounding error buildup. Because of the very high degree of precision, this is a very powerful tool. Tables of x_j and w_j are given to high accuracy in Abramowitz and Stegun for $N = 2, 3, \ldots, 10, 12, 16, 20, \ldots 96$. Fortran subroutines using some of these are available from me.

For functions which are analytic on the closed interval, Gaussian quadrature is highly effective. If there is a singularity, say at one endpoint, then $f^{(2N)}$ may not be small at all, and this is not a suitable approach.

If a branch point exists at the endpoint with a known behavior, it may be possible to use another set of orthogonal functions. For example, if f is known to behave like $(x+1)^\beta$ near $x = -1$, and like $(1-x)^\alpha$ near $x = +1$, then we can write

$$f = (1-x)^\alpha (1+x)^\beta g(x) = w(x) g(x).$$

If $g(x)$ is analytic, a good approximation to $\int w(x)g(x)$ can be found by approximating g by a finite polynomial, naturally written in terms of Jacobi Polynomials.

Similarly, $\int_0^\infty e^{-x} g(x)\, dx$ uses an expansion of g in Laguerre polynomials.

1.2 Numerical Techniques for ODE's

Suppose we have an n'th order ordinary differential equation in $y(x)$. This can be considered as n coupled first order ODE's in variables

$$\begin{aligned} y_0(x) &:= y(x) \\ y_1(x) &:= y'(x) \\ \vdots \quad & \quad \vdots \\ y_{n-1} &:= y^{(n-1)}(x). \end{aligned}$$

Most of the equations are trivial: $\dfrac{dy_j}{dx} = y_{j+1}$, but one is the original equation with $\dfrac{d^n y}{dx^n}$ replaced by $\dfrac{d}{dx} y_{n-1}$, and all other $\dfrac{d^j y}{dx^j}$ replaced by y_j. So consider y to be a vector, and consider the first order ODE

$$\frac{d\vec{y}}{dx} = \vec{f}(x, \vec{y}).$$

We wish to solve this equation with initial conditions $\vec{y} = \vec{y}_A$ at one point $x = x_A$.

The most straightforward way to do this is, at each point x_j, to calculate $\left.\dfrac{d\vec{y}}{dx}\right|_{x_j}$ and approximate

$$\vec{y}(x_{j+1}) = \vec{y}(x_j) + (x_{j+1} - x_j) \left.\frac{d\vec{y}}{dx}\right|_{x_j}.$$

If the x_j are evenly spaced with spacing h, this entails an error in $\vec{y}(x_{j+1}) - \vec{y}(x_j)$ of order h^2. These errors may accumulate in $y(x)$ as we proceed. As the number of steps to make a fixed change in x is proportional to $1/h$, the total error could be of order h.

We could do better if we could approximate $\vec{y}(x_{j+1}) - \vec{y}(x_j) = hy'(x_j + h/2)$ instead of using y' at the initial point. This would give an error $\propto h^3 \vec{y}^{(3)}$ instead of $h^2 \vec{y}^{(2)}$. But if $\vec{f}(x, \vec{y})$ depends on $\vec{y}$, we need an approximation for $\vec{y}(x_j + h/2)$ to evaluate this.

There are two approaches to finding $\vec{y}'(x_j + h/2)$ which would be given by f if we knew $\vec{y}(x_j + h/2)$. One is to make a sequence of evaluations

$$\begin{aligned} \vec{K}_0 &= \vec{f}(x_j, \vec{y}_j) \\ \vec{K}_1 &= \vec{f}(x_j + h/2, \vec{y}_j + \frac{1}{2} h \vec{K}_0) \\ \vec{K}_2 &= \vec{f}(x_j + h/2, \vec{y}_j + \frac{1}{2} h \vec{K}_1) \\ \vec{K}_3 &= \vec{f}(x_j + h, \vec{y}_j + h \vec{K}_2) \end{aligned}$$

$\vec{K}_0$ and $\vec{K}_3$ approximate from the two ends, while $\vec{K}_1$ and $\vec{K}_2$ are two approximations in the middle. Each has significant errors, but

$$\vec{y}_{j+1} = \vec{y}_j + \frac{h}{6}\left(\vec{K}_0 + 2\vec{K}_1 + 2\vec{K}_2 + \vec{K}_3\right)$$

has these cancelling out up to order h^5, so it is a reasonable method. It is called the **Runge-Kutta** method.

Another approach is to consider the intermediate points on the same footing, so we relabel x_j and h. We use

$$\bar{\vec{y}}_{j+1} = \vec{y}_{j-1} + 2h\vec{y}\,'_j$$

to predict the value at x_{j+1}. Note we need the two previous values.

Now calculate $\bar{\vec{y}}'_{j+1} = \vec{f}(x_{j+1}, \bar{\vec{y}}_{j+1})$. We can make another order h^2 calculation of $\vec{y}_{j+1}$ by $\vec{y}_{j+1} = \vec{y}_j + \frac{h}{2}\left(\vec{y}'_j + \vec{y}'_{j-1}\right)$, which is called the corrected value. If $\vec{y}_{j+1}$ is not close to $\bar{\vec{y}}_{j+1}$, we set the new $\bar{\vec{y}}_{j+1}$ predictor and correct again.

Fancier versions also exist, using not just the last two values but the last N, $N = 3, \cdots, 10$).

One problem with this method, called **predictor-corrector**, is that when you start off you only have one value of $\vec{y}(x_A)$, which is not enough to proceed. So programs which use predictor-corrector methods generally start with Runge-Kutta to generate the first N values, and then switch to predictor-corrector methods.

I have, of course, only scratched the surface. At some point in your career, if you are faced with some numerical calculation, I suggest you do some learning on this subject before implementing an overly naïve approach.