Overview over Numerical Methods

Harald O. Jeschke

Institut für Theoretische Physik
Goethe-Universität Frankfurt
Email: jeschke@itp.uni-frankfurt.de

July 16, 2009

List of topics

General numerical methods

1. Numerical integration
2. Polynomial and spline interpolation
3. Minimization: Conjugate gradient, genetic algorithms
4. Solving integral equations
5. Numerical differentiation
6. Differential equations: Finite difference
7. Differential equations: Finite elements
8. Eigenvalue problems

Methods of Theoretical Physics

1. Exact Diagonalization
2. Monte Carlo
3. Quantum Monte Carlo

Motivation

- **Evaluation of theoretical models**: Many theoretical approaches involve a numerical evaluation of resulting equations as a final step. This is not a trivial part: Success of a theory often depends crucially on the feasibility of its numerical evaluation.
- **Making contact to real materials**: Nearly all measurable quantities in real materials are off limits for analytical computation. If you want to compare a theory to experiment, numerical methods are needed to account for the complexity of chemical interactions, real lattice structures, interplay of various phenomena present at the same time, and so on.
- **Introducing computational physics**: This area of physics is of growing importance as computers become more powerful and as more and more nontrivial aspects of experiment and technology can be computed or simulated, yet it is hardly mentioned in physics classes.

My perspective

Fields of research	Methods I use
Tight binding molecular dynamics on time dependent potential energy surfaces	Matrix diagonalization Integration of differential equations Fast Fourier transform
Dynamical mean field theory for lattice models (Hubbard, Anderson)	Integral equations Splines Exact diagonalization
Ab initio density functional theory	Minimization techniques

Numerical integration: Introduction I

A function f that is continous in an interval I_x has antiderivatives F that only differ by a constant, and

$$\frac{dF(x)}{dx} = F'(x) = f(x), \quad x \in I_x \tag{1}$$

The number $I(f : \alpha, \beta)$ is called the definite integral of f over $[\alpha, \beta]$, and we have the Fundamental Theorem of Calculus:

$$I(f : \alpha, \beta) := \int_{\alpha}^{\beta} dx\, f(x) = F(\beta) - F(\alpha), \quad [\alpha, \beta] \in I_x\,, \tag{2}$$

then f is called integrable in $[\alpha, \beta]$.
In practice, many integrals $I(f : \alpha, \beta)$ need to be calculated approximatively by socalled quadrature formulas.

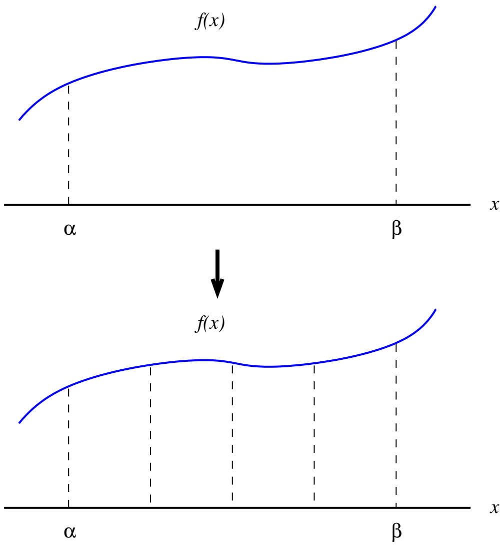
Numerical integration: Introduction II

The reason for the need of numerical integration can be

- f has an antiderivative F that cannot be represented in closed form (for example $f(x) = e^{-x^2}$).
- f is only known on a discrete mesh $x_k \in [\alpha, \beta]$.
- The determination of the antiderivative is too involved.
- Multidimensional integrals.
- Integrating numerically becomes very important in the solution of differential equations.

Numerical integration: Introduction III

Consider finding the area under a reasonably smooth curve f(x) in the interval $[\alpha, \beta]$. It is tempting to sample the function at regular intervals with uniformly spaced ordinates:



Numerical integration: Introduction IV

We could evaluate the area for a number n of ordinates, then cut each interval in half by adding $n - 1$ additional ordinates and evaluate again, until we reach the desired accuracy.

- The most common assumption is that $f(x)$ can be well represented by a polynomial of some degree if only we go high enough in the degree.
- This is equivalent to saying that $f(x)$ has a good Taylor's series expansion almost everywhere, but in reality that is far from guaranteed!
- One should remember that no polynomial has an asymptote, neither a horizontal nor a vertical one!

Integration on equidistant meshes: Trapezoid rule I

To obtain the simplest quadrature formulas, we can write the quadrature in the form

$$I \approx \sum_{i=0}^{N} W_i f_i \tag{3}$$

with evaluation points x_i, $f_i = f(x_i)$, W_i is the weight of the i-th point, and we evaluate at $N + 1$ points. Thus, the simplest formula is

$$I \approx W_0 f_0 + W_1 f_1 \tag{4}$$

where $x_0 = \alpha$ and $x_1 = \beta$ are the limits of integration.

Trapezoid rule II

For the approximation to be useful, we require it to be exact for the simplest integrands, $f(x) = 1$ and $f(x) = x$. As these are the first terms of a Taylor's series, this approximation will converge to the exact result as the integration region is made smaller, for any $f(x)$ that has a Taylor series expansion. Thus, we require that

$$\int_{x_0}^{x_1} dx\, 1 = x_1 - x_0 \overset{!}{=} W_0 + W_1 \quad \text{and} \tag{5}$$

$$\int_{x_0}^{x_1} dx\, x = \frac{x_1^2 - x_0^2}{2} \overset{!}{=} W_0 x_0 + W_1 x_1 \tag{6}$$

This gives us two equations for the two unknown weights, and thus

$$W_0 = W_1 = \frac{x_1 - x_0}{2} \tag{7}$$

Trapezoid rule III

With this, we arrive at the **trapezoid rule** given by

$$\int_{x_0}^{x_1} dx\, f(x) \approx \frac{h}{2}(f_0 + f_1) \tag{8}$$

where $h = \beta - \alpha$. Displaying also Lagrange's expression for the remainder term in the Taylor series expansion, we have

$$\int_{x_0}^{x_1} dx\, f(x) = \frac{h}{2}(f_0 + f_1) - \frac{h^3}{12} f^{(2)}(\xi)\,, \tag{9}$$

where ξ is some point within the region of integration.

Simpson's rule

To derive the next higher integration rule for $N = 2$, we write

$$\int_{x_0}^{x_2} dx\, 1 = x_2 - x_0 \overset{!}{=} W_0 + W_1 + W_2\,, \tag{10}$$

$$\int_{x_0}^{x_1} dx\, x = \frac{x_2^2 - x_0^2}{2} \overset{!}{=} W_0 x_0 + W_1 x_1 + W_2 x_2 \quad \text{and} \tag{11}$$

$$\int_{x_0}^{x_1} dx\, x^2 = \frac{x_2^3 - x_0^3}{3} \overset{!}{=} W_0 x_0^2 + W_1 x_1^2 + W_2 x_2^2 \tag{12}$$

Solving for the weights, we find **Simpson's rule**

$$\int_{x_0}^{x_1} dx\, f(x) = \frac{h}{3}(f_0 + 4f_1 + f_2) - \frac{h^5}{90} f^{(4)}(\xi)\,, \tag{13}$$

Non-equidistant mesh integration: Gaussian Quadrature

We can give up equal spacing of the ordinates and choose the location of the ordinates so as to optimize the estimation of the area (still assuming the integrand is well represented by a polynomial). This method is called **Gaussian quadrature**. The best locations for the ordinates turn out to be the roots of the Legendre polynomial of appropriate degree. If the region of integration is normalized to span $[-1,+1]$ the simplest of the Gaussian quadrature rules are (x_i is the abscissa and y_i the corresponding ordinate):

$$\begin{aligned}
G_2 &= y_{-1} + y_1 \qquad \text{with } x_{\pm 1} = \pm 0.57735 \\
G_3 &= 0.88889 y_0 + 0.55556(y_{-1} + y_1) \\
&\quad \text{with } x_0 = 0,\ x_{\pm 1} = \pm 0.77460 \\
G_4 &= 0.65215(y_{-1} + y_1) + 0.03485(y_{-2} + y_2) \\
&\quad \text{with } x_{\pm 1} = \pm 0.33998,\ x_{\pm 2} = \pm 0.86114 \\
G_5 &= 0.56889 y_0 + 0.47863(y_{-1} + y_1) + 0.23693(y_{-2} + y_2) \\
&\quad \text{with } x_0 = 0,\ x_{\pm 1} = \pm 0.0.53847,\ x_{\pm 2} = \pm 0.90618
\end{aligned} \tag{14}$$

Gaussian Quadrature

A general interval $[\alpha, \beta]$ can be transformed to the interval $[-1, +1]$ by the transformation $t = (2x - a - b)/(b - a)$:

$$\int_\alpha^\beta dx\, f(x) = \int_{-1}^1 dt\, f\left(\frac{(\beta-\alpha)t + \beta + \alpha}{2}\right) \frac{b-a}{2} \tag{15}$$

leading to the Gaussian quadrature formula

$$\int_\alpha^\beta dx\, f(x) = \frac{b-a}{2} \sum_{j=1}^{n} c_{n,j} f\left(\frac{(\beta-\alpha)r_{n,j} + \beta + \alpha}{2}\right) \tag{16}$$

Further methods:

Richardson extrapolation

Adaptive quadrature schemes

Practice: Use of numerical libraries

- Numerical recipes: Books in Fortran77, Fortran 90 and C available online, *e.g.*
  `http://www.library.cornell.edu/nr/cbookcpdf.html`
  But only the simplest algorithms (implementations fitting on a book page) are included (no adaptive integration).
- IMSL (International Mathematical and Statistical Library). Very large, but commercial:
  `http://www.absoft.com/Products/Libraries/imsl.html`
- NAG (Numerical Algorithms Group). Very large, available in Fortran77, Fortran90 and C, also commercial, but often available in universities. Full documentation available online (free). See
  `http://www.nag.co.uk/`
  NAG is installed on the CSC (Center for Scientific Computing, Frankfurt).

Interpolation

Interpolation is a way to approximate a function that is only given at a finite number of supporting points between these supports. In contrast to curve fitting, interpolating functions are exact at the supporting points. Interpolation can be important for a number of purposes:

- If calculation of a function is computationally expensive so that only a limited number of evaluations are possible.
- For the calculation of integrals or derivatives of a function that is only available on a mesh.
- For the construction of smooth and flexible shapes in computer graphics.

Lagrange polynomials I

If a function is known at several points, it can be interpolated following a method of Lagrange. **Lagrange's interpolating polynomial** $p(x)$ can be derived from a Taylor's series at the supporting points, *e.g.* x_1 and x_2:

$$f(x_1) = f(x) + (x_1 - x)f'(x) + \dots\,, \tag{17}$$
$$f(x_2) = f(x) + (x_2 - x)f'(x) + \dots\,. \tag{18}$$

Truncating at the first order compromises the equality, so we introduce an approximate function $p(x)$:

$$f(x_1) = p(x) + (x_1 - x)p'(x)\,, \tag{19}$$
$$f(x_2) = p(x) + (x_2 - x)p'(x)\,. \tag{20}$$

This gives us two equations for the two unknowns $p(x)$ and $p'(x)$; solving for $p(x)$ gives a linear function in x:

$$p(x) = \frac{x - x_2}{x_1 - x_2} f(x_1) + \frac{x - x_1}{x_2 - x_1} f(x_2) \tag{21}$$

Lagrange polynomials II

This linear interpolation gained from two points could have been obtained in other ways, but in this form it shows that the contribution of the function value $f(x_2)$ to the approximation is weighted by the distance between x and x_1, varying smoothly from 0 to 1.
A higher order approximation can easily be obtained, but the function needs to be known at an additional point. Again, we truncate the tree equations after replacing $f(x)$ by $p(x)$:

$$f(x_1) = f(x) + (x_1 - x)f'(x) + \frac{(x_1 - x)^2}{2}f''(x) + \dots\,, \tag{22}$$

$$f(x_2) = f(x) + (x_2 - x)f'(x) + \frac{(x_2 - x)^2}{2}f''(x) + \dots\,, \tag{23}$$

$$f(x_3) = f(x) + (x_3 - x)f'(x) + \frac{(x_3 - x)^2}{2}f''(x) + \dots\,, \tag{24}$$

Lagrange polynomials III

If we now solve for $p(x)$, this yields the quadratic interpolating polynomial:

$$\begin{aligned} p(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} f(x_1) + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} f(x_2) \\ + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} f(x_3) \end{aligned} \tag{25}$$

The general form of the Lagrange interpolation polynomial of order $n-1$ can be written as

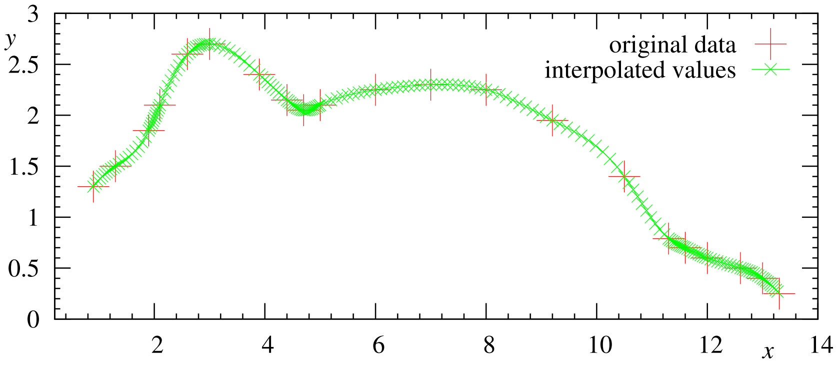
$$\begin{aligned} p_n(x) = f(x_1) l_{1,n}(x) + \ldots + f(x_n) l_{n,n}(x) = \sum_{j=1}^{n} l_{j,n}(x) f(x_j) \\ \text{with } l_{j,n}(x) = \frac{(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \end{aligned} \tag{26}$$

Note that $l_{j,n}(x_i) = \delta_{ij}$.

Better interpolation methods

- If derivatives of the function are available: Hermite polynomials
- Continous interpolation of a function **with continous derivatives** if derivatives of the approximated function unknown: **Cubic splines**.

Spline example (data points representing the upper profile of a flying duck); Plot of the result together with the original points:



Minimization: Golden section search I

Finding the minimum of a function in one dimension

The simplest strategy for finding a minimum of a function is bracketing, similar to bisection in root finding. But in contrast to root finding where the best strategy is to continuously half the search interval, the selection of an optimal new abscissa point is different in the case of minimization.

While a root is bracketed by two points a and b if the signs of $f(a)$ and $f(b)$ are opposite, we need three points to bracket a minimum: $a < b < c$ with the property $f(a) > f(b)$ and $f(c) > f(b)$. Now if we choose a new point x between b and c, we can have $f(b) < f(x)$ leading to the new bracketing triplet (a, b, x), or $f(b) > f(x)$ leading to the triplet (b, x, c).

Golden section search II

Now for a strategy to choose the new point x given (a, b, c). If b is a fraction w of the way between a and c:

$$\frac{b-a}{c-a} = w \qquad \frac{c-b}{c-a} = 1 - w \tag{27}$$

and the new trial point x is an additional fraction z beyond b:

$$\frac{x-b}{c-a} = z \tag{28}$$

Then the next bracketing segment will be either $w + z$ or $1 - w$ in length. In order to minimize the worst case possibility, we will try to make them equal:

$$z = 1 - 2w \tag{29}$$

This makes $|b - a|$ equal to $|x - c|$. But now w is still undetermined. We can find it by demanding that w was also chosen optimally.

Golden section search III

The scale similarity implies that x should be the same fraction of the way from b to c as was b from a to c, or

$$\frac{z}{1-w} = w \tag{30}$$

Together, Eqs. (29) and (30) yield

$$w^2 - 3w + 1 = 0 \curvearrowright w = \frac{3-\sqrt{5}}{2} \approx 0.38197 \tag{31}$$

Thus in a bracketing triplet (a, b, c), b has a relative distance of 0.38197 from a and of 0.61803 from c. These fractions correspond to the golden section so that the minimization is also called **golden section search**. The convergence of this method is linear, meaning that additional significant figures are won linearly with additional function evaluations.

Precision is machine limited

It is important to note that determination of a minimum can only be done up to a precision corresponding to the square root of the machine precision; *e.g.* for `double` $3 \cdot 10^{-8} \approx \sqrt{10^{-15}}$. This can be understood considering the Taylor expansion close to the minimum

$$f(x) \approx f(b) + \frac{1}{2} f''(b)(x-b)^2 \tag{32}$$

The second term will be negligible against the first, *i.e.* a factor of the floating point precision ϵ smaller, if

$$|x-b| < \sqrt{\epsilon}|b| \sqrt{\frac{2|f(b)|}{b^2 f''(b)}} \tag{33}$$

Multidimensional Minimization: Steepest Descent I

The first idea for minimization in N dimensions is to reduce the task to subsequent onedimensional minimizations.

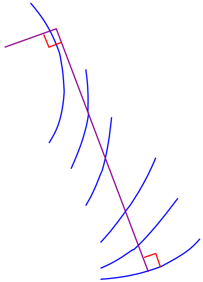
Algorithm of the **steepest descent** method:

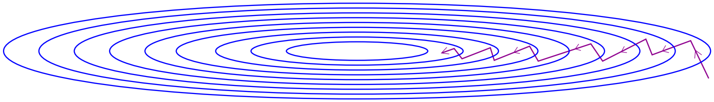
Start at point $\mathbf{P}_0$. As many times as needed, move from point $\mathbf{P}_i$ to the point $\mathbf{P}_{i+1}$ by minimizing along the line from $\mathbf{P}_i$ in the direction of the local downhill gradient $-\nabla f(\mathbf{P}_i)$.

This algorithm is not very good as it will perform many small steps in going down a long, narrow valley even if the valley has perfect quadratic analytic form.

Steepest Descent II

The figures show how the steepest descent directions zigzag, and how a descent starts off perpendicular to a contour line and proceeds until it is parallel to another in its local minimum, forcing a right angle turn.

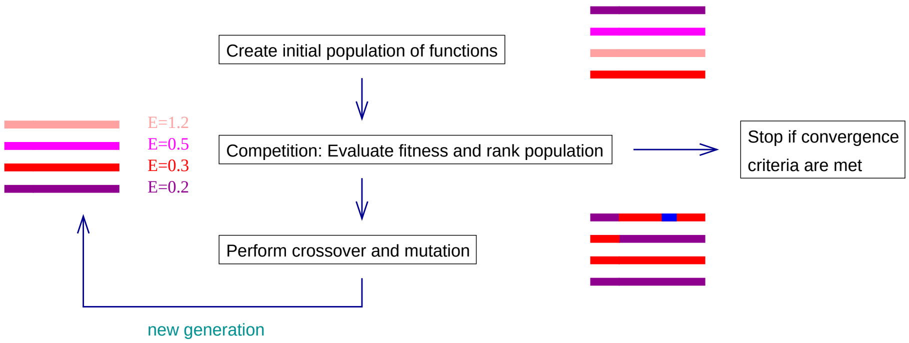




The problem of this method is that we need to cycle many times through all N basis vectors in order to reach the minimum. It would be desirable to improve the choice of minimization directions for the N dimensional function, in order to proceed along valley directions or to choose "non-interfering" directions in which minimization along one direction doesn't spoil the previous minimizations along other directions.
$\rightarrow$ **Conjugate gradient minimization**

Genetic Algorithms

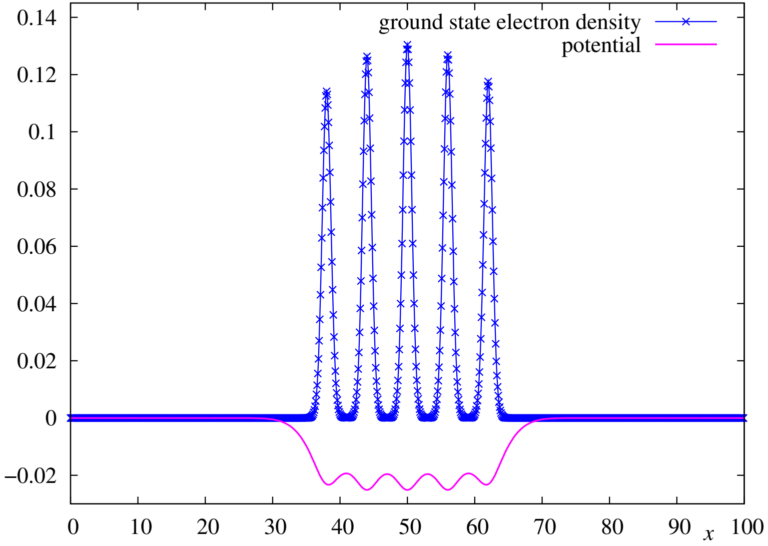
Flow chart of GA optimization


Create initial population of functions
E=1.2
E=0.5
E=0.3
E=0.2
Competition: Evaluate fitness and rank population
Stop if convergence criteria are met
Perform crossover and mutation
new generation

These steps are common to many GA methods; they differ by

1. the way the system to be optimized is represented
2. the rules of the competition
3. the way crossover and mutation are implemented

Genetic Algorithms: 1D Schrödinger equation



Numerical differentiation I

The first order derivative of a function $f(x)$ at a point x is defined as the limit

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{34}$$

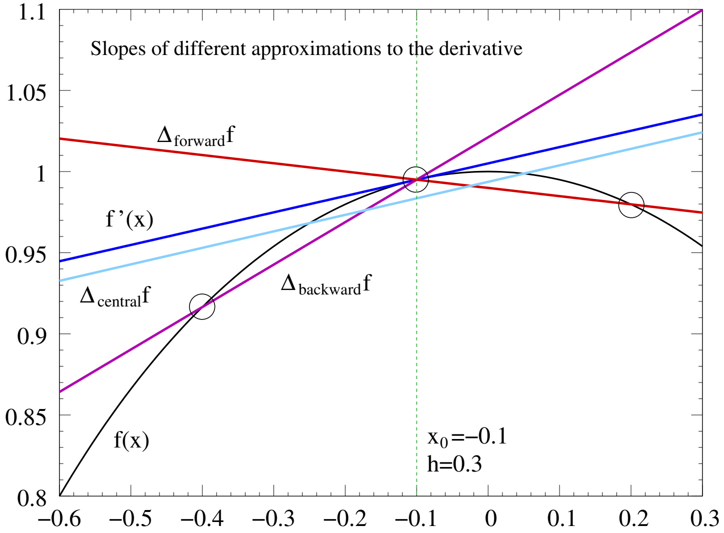
In numerical calculations, it is impossible to work directly with such limiting values. Therefore, we have to work with *finite differences* $\Delta f(x)$ like

$$\Delta_{\mathrm{forward}} f(x) \equiv f(x+h) - f(x) \tag{35}$$

known as the forward difference and approximate the derivative with the quotient $\Delta f(x)/h$ for finite h. Other differences that can be used are backward and central differences:

$$\Delta_{\mathrm{backward}} f(x) \equiv f(x) - f(x-h) \tag{36}$$
$$\Delta_{\mathrm{central}} f(x) \equiv f(x+h/2) - f(x-h/2) \tag{37}$$

Numerical differentiation II



Numerical differentiation III

With the notation $f(x_i) \equiv f_i$ we can write for the central difference approximation to the derivative at mesh point k

$$f'_k = \frac{f_{k+1/2} - f_{k-1/2}}{h} \tag{38}$$

In case the use of half intervals is inconvenient, the central difference can also be taken over two subintervals

$$f'_k = \frac{f_{k+1} - f_{k-1}}{2h} \tag{39}$$

Generalizing Eq. (38) to other derivatives, we obtain a central difference form of the second derivative

$$f''_k = \frac{f'_{k+1/2} - f'_{k-1/2}}{h} = \frac{f_{k+1} - 2f_k + f_{k-1}}{h^2} \tag{40}$$

For many applications, central differences are preferred over forward or backward differences.

Numerical differentiation IV

The reason can be seen from a Taylor expansion of $f(x+h)$ at x:

$$f(x+h) = f(x) + \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f^{(3)}(x) + \dots \tag{41}$$

Rearrangement yields

$$f'(x) = \frac{1}{h}[f(x+h) - f(x)] - \frac{h}{2!}f''(x) - \dots \tag{42}$$

which shows that the error in the forward difference approximation is of order h times the second derivative. Likewise, we can write the Taylor expansion for $f(x-h)$

$$f(x-h) = f(x) - \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f^{(3)}(x) + \dots \tag{43}$$

and obtain an error of the same order for the backward difference approximation.

Numerical differentiation V

But subtracting Eq. (43) from Eq. (41), we obtain the expression

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{3!} f^{(3)}(x) + \dots \tag{44}$$

Thus, the error in approximating $f'(x)$ with the central difference is smaller by one order in h than forward and backward differences.

In practice, achieving good accuracy in the numerical calculation of derivatives is difficult in comparison with numerical integration, for example.

The reason can be understood from the form of Eq. (38): It requires calculating the ratio of two differences: $f(x + h/2) - f(x - h/2)$ and $(x + h/2) - (x - h/2)$. As h becomes small, the differences will be several orders of magnitude smaller than the values of f or x.

Numerical differentiation VI

The number of significant figures in the differences will thus be much smaller than the theoretical number of significant figures corresponding to the machine precision.

In order to improve over the essentially linear approximation to the tangent of $f(x)$ at x, one can obtain five-point approximations for the first and second derivative using a Taylor series expansion for $f(x \pm 2h)$:

$$f'_k = \frac{f_{k-2} - 8f_{k-1} + 8f_{k+1} - f_{k+2}}{12h} + \frac{h^4}{30} f_k^{(5)} + \dots \tag{45}$$

$$f''_k = \frac{-f_{k-2} + 16f_{k-1} - 30f_k + 16f_{k+1} - f_{k+2}}{12h^2} + \frac{h^4}{90} f_k^{(6)} + \dots \tag{46}$$

The errors are two orders in h smaller than in the approximations of Eqs. (44) and (40).

Differential equations I

Types of differential equations

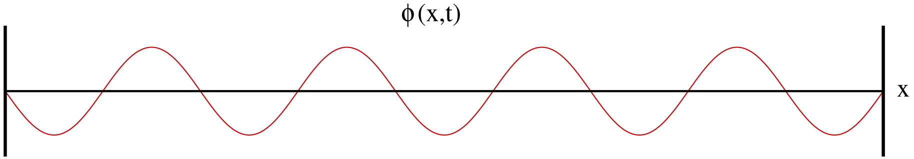
A differential equation relates a function, its derivatives and the independent variables. For example, the equation for simple harmonic motion

$$\frac{d^2\phi}{dt^2} + \omega_0^2\phi(t) = 0 \tag{47}$$

with $\omega_0 = k/m$ and string constant k, mass m is a linear second order differential equation as the highest order derivative is a second derivative, ϕ and its derivatives appear in the first order and there are no products between them. Nonlinear differential equations are more difficult to solve than linear ones.

If ϕ has more than one independent variable, partial derivatives enter into the equation as in the example of a vibrating string where the vertical displacement $\phi(x, t)$ is a function of time as well as location x.

Differential equations II



The equation of motion is a *partial differential equation* (PDE)

$$\frac{\partial^2\phi}{\partial x^2} - \frac{1}{v^2}\frac{\partial^2\phi}{\partial t^2} = 0 \tag{48}$$

with phase velocity v. In contrast, Eq. (47) is an *ordinary differential equation* (ODE).

Differential equations III

In general, a two-dimensional (*i.e.* two independent variable) second-order partial differential equation can be written in the form

$$p\frac{\partial^2\phi}{\partial x^2} + q\frac{\partial^2\phi}{\partial x\partial y} + r\frac{\partial^2\phi}{\partial y^2} + s\frac{\partial\phi}{\partial x} + t\frac{\partial\phi}{\partial y} + u\phi + v = 0 \tag{49}$$

where p, q, r, s, t, u and v may be functions of the independent variables x and y.

If they do not depend on dependent variable ϕ or its derivatives, it is a linear PDE, otherwise it would be a nonlinear PDE of higher order. The method of solution depends critically on the order of the equation and on whether it is linear or not.

If $q^2 < 4pr$, it is called an *elliptic* equation, if $q^2 = 4pr$ a *parabolic*, if $q^2 > 4pr$ a *hyperbolic* equation.

Differential equations IV

An example of an elliptic equation is the two-dimensional Poisson equation

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = -\rho(x,y) \tag{50}$$

The Poisson equation describes the field $\phi(x,y)$ of a charge distribution $\rho(x,y)$ in two dimensions. A standard example of a parabolic equation is the diffusion equation

$$\frac{\partial\phi}{\partial t} = -\frac{\partial}{\partial x}\left(D\frac{\partial\phi}{\partial x}\right) \tag{51}$$

where ϕ stands for the concentration of a certain kind of particle with diffusion coefficient D.

Differential equations V

Initial and boundary value problems

A classification of differential equations that is very important for their numerical solution is between initial and boundary value problems.

- The *initial value problem* propagates the solution forward from values given at the starting point.
- The *boundary value problem* has constraints that must be fulfilled both at the start and at the end of the interval.
- For PDEs, it may happen that the conditions for some independent variables are given as initial values, for others as boundary conditions. This would be an initial value boundary problem, a mixture of both types.

Differential equations VI

For example, consider the harmonic oscillator of Eq. (47). Let's assume we are interested in the solution in the interval $t = [t_0, t_N]$. As it is a second order differential equation, we need to supply two pieces of information before we can solve it. If we specify for example the value of $\phi(t)$ and its derivative at $t = t_0$, we have an initial value problem. If we have two values of $\phi(t)$ at $t = t_0$ and at $t = t_N$, we have a boundary value problem.

In general, the numerical solution of boundary value problems is more difficult than that of initial value problems because the relatively simple approaches of propagating a solution do not work as we do not have enough information at the starting point. Thus, numerical methods for both types are different.

Finite difference solution of differential equations I

Euler's method for initial value problems
The general philosophy of solving differential equations can be shown with Euler's method for the example of Eq. (47).

As in numerical integration, we have to discretize the interval $[t_0, t_N]$ on which we want to solve the equation by introducing a mesh. Our aim is then to find the values of $\phi(t)$ at discrete values $t = t_0, t_1, t_2, \dots, t_N$. The distance between two consecutive points (the step size) is

$$h_i = t_{i+1} - t_i \tag{52}$$

In the limit if all N subintervals $h_i \to 0$, we recover the continuous function of an analytical solution.

Finite difference solution of differential equations II

To solve for $\{\phi(t_1), \phi(t_2), \dots, \phi(t_{N-1})\}$, we can convert Eq. (47) into a set of algebraic equations by rewriting he second-order derivative using finite differences.
Following Eq. (40) we may write

$$\left.\frac{\partial^2\phi}{\partial t^2}\right|_{t=t_i} \longrightarrow \frac{\phi(t_{i+1}) - 2\phi(t_i) + \phi(t_{i-1})}{h^2} \tag{53}$$

Using a constant step size $h = t_{i+1} - t_i$ and the short hand $\phi_i = \phi(t_i)$, we arrive at the relation between the values of ϕ at three different times:

$$\phi_{i+1} - (2 - h^2\omega_0^2)\phi_i + \phi_{i-1} = 0 \tag{54}$$

This is called a *finite difference equation* (FDE) as it relates differences in the values of ϕ at nearby points.

Finite difference solution of differential equations III

To solve the initial value problem, we need two independent pieces of input on $\phi(t)$ at the starting time t_0. For example, the mass could be displaced at the start by one unit in positive direction:

$$\phi_0 \equiv \phi(t = t_0) = 1 \tag{55}$$

It is released at $t = t_0$ without any initial velocity:

$$\left.\frac{d\phi(t)}{dt}\right|_{t=t_0} = 0 \tag{56}$$

Using finite differences, these two initial conditions can be approximated by $\phi_0 \approx \phi_1 = 1$. That provides two of three ϕ values in Eq. (54) for $i = 1$, allowing us to start propagating the solution.

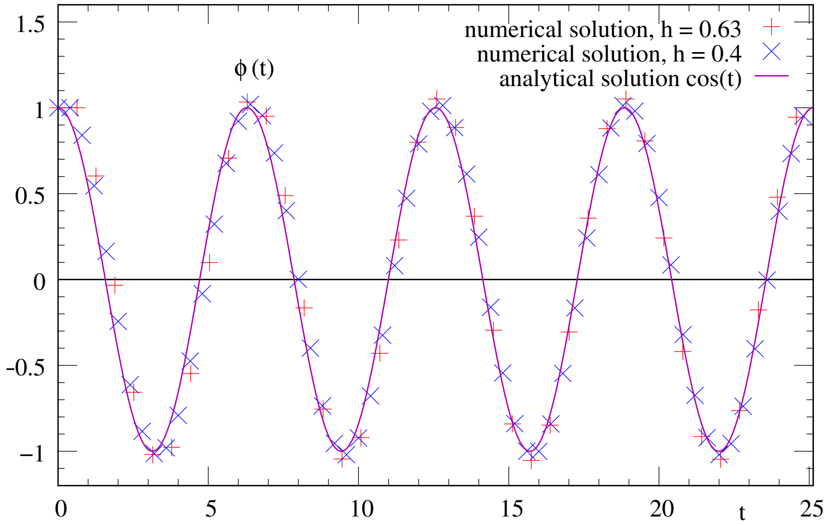
Finite difference solution of differential equations IV

In mathematica the implementation of Eq. (54) would look as simple as this:

```
step = 0.63; num = 100; omega0 = 1;
phi = Table[{i*step, 0}, {i, 0, num - 1}];
phi[[1, 2]] = 1;
phi[[2, 2]] = 1;
Do[phi[[i, 2]] = - phi[[i - 2, 2]]
   + (2.0 - step*step*omega0*omega0)*phi[[i - 1, 2]],
   {i, 3, num}]
Export["euler1.dat", phi, "Table"];
```

The following figure shows the result for $\omega_0 = 1$ and for two step sizes $h = 0.63$ and $h = 0.4$.

Finite difference solution of differential equations V

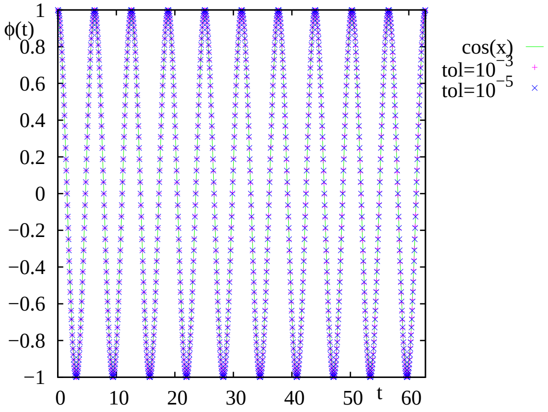


Finite difference solution of differential equations VI

The main source of error in the numerical solution of differential equations is the truncation error inherent in the finite difference approximation to the derivatives. *Multistep methods* (function evaluation at more than two points) allow higher order approximations to the derivative.

$\rightarrow$ **Runge-Kutta methods**

- Routine `d02pvc` from the NAG C library
- Solution of oscillator problem using `Nag_RK_4_5`, $t \in [0, 20\pi]$:



Finite element solution to partial differential equations I

In boundary value problems, finite element methods in which the space is divided into a number of small elements are more often used than finite difference methods. In each element, the solution is approximated by simple functions, characterized by a few parameters. Advantages are ease of treatment of cases with odd geometrical shapes and flexibility in adjusting the size of the elements to achieve fine subdivisions.
The basic principles of finite element methods can be demonstrated with a simple example, the first-order differential equation

$$\frac{d\phi(t)}{dt} + \lambda\phi(t) = 0 \tag{57}$$

describing exponential decay. For simplicity, we focus on $t = [0, 1]$ and take $\lambda = 1$. The one boundary condition of the first-order ODE can be taken to be $\phi(t = 0) = 1$.

Finite element solution to partial differential equations II

The analytical solution of Eq. (57) in this case is

$$\phi(t) = e^{-t} = 1 - t + \frac{t^2}{2!} - \frac{t^3}{3!} + \dots \tag{58}$$

For the numerical solution, we now try a power series with n terms:

$$\tilde{\phi}_n(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_n t^n \tag{59}$$

To satisfy the initial condition, we need $a_0 = 1$. We now discuss the case of $n = 2$ degrees of freedom:

$$\tilde{\phi}_2(t) = 1 + a_1 t + a_2 t^2 \tag{60}$$

This solution will differ from the exact solution Eq. (58), but our goal is to make it as good as possible on the limited domain $t = [0, 1]$.

Finite element solution to partial differential equations III

First, we substitute the trial solution Eq. (60) into the differential equation; we define a quantity called the **residual** measuring the degree to which the approximate solution fulfils the differential equation:

$$R(t; a1, a2) \equiv \frac{d\tilde{\phi}_2(t)}{dt} + \tilde{\phi}_2(t) = 1 + (1+t)a_1 + (2t + t^2)a_2 \qquad (61)$$

In the case we allow for n parameters, we can write (with $\mathbf{a} = \{a_1, a_2, ..., a_n\}$)

$$R(t; \mathbf{a}) \equiv \frac{d\tilde{\phi}_n(t)}{dt} + \tilde{\phi}_n(t) \qquad (62)$$

We now determine the value of **a** by minimizing the residual in the domain $t = [0, 1]$ with four methods.

Finite element solution to partial differential equations IV

Collocation method: We require the residual to vanish at n points $t_1, t_2, \ldots, t_n$ within the domain of interest:

$$R(t_i; \mathbf{a}) = 0 \qquad \text{for } i = 1, 2, \ldots, n \tag{63}$$

For the $n = 2$ approximation, we can take any two points within $[0, 1]$, for example $t_1 = 1/3$ and $t_2 = 2/3$. This leads to two equations:

$$R\left(t = \frac{1}{3}; \mathbf{a}\right) = 1 + \frac{4}{3}a_1 + \frac{7}{9}a_2 = 0 \tag{64}$$

$$R\left(t = \frac{2}{3}; \mathbf{a}\right) = 1 + \frac{5}{3}a_1 + \frac{16}{9}a_2 = 0 \tag{65}$$

Finite element solution to partial differential equations V

The roots of this set of equations are

$$a_1 = -\frac{27}{29} \qquad a_2 = \frac{9}{29} \tag{66}$$

leading to the solution

$$\phi_2^{\mathrm{col}}(t) = 1 - \frac{27}{29}t + \frac{9}{29}t^2 \tag{67}$$

Subdomain method: Instead of asking the residual to vanish in n points, we can also demand that it vanish on average in n subdomains (not necessarily nonoverlapping ones):

$$\frac{1}{\Delta t_i}\int_{\Delta t_i} dt\, R(t; \mathbf{a}) = 0 \tag{68}$$

Finite element solution to partial differential equations VI

For the two-parameter approximation we can simply choose two equal subdomains $\Delta t_1 = [0, 1/2]$ and $\Delta t_2 = [1/2, 1]$. The average residuals are:

$$\frac{1}{\Delta t_1} \int_0^{1/2} dt\, R(t; \mathbf{a}) = 2\left[\frac{1}{2} + \frac{5}{8} a_1 + \frac{7}{24} a_2\right] \tag{69}$$

$$\frac{1}{\Delta t_2} \int_{1/2}^{1} dt\, R(t; \mathbf{a}) = 2\left[\frac{1}{2} + \frac{7}{8} a_1 + \frac{25}{24} a_2\right] \tag{70}$$

Requiring both to vanish leads to

$$a_1 = -\frac{18}{19} \qquad a_2 = \frac{6}{19} \tag{71}$$

and thus

$$\phi_2^{\mathrm{sub}}(t) = 1 - \frac{18}{19} t + \frac{6}{19} t^2 \tag{72}$$

Finite element solution to partial differential equations VII

Least squares method: Using the condition of maximum likelihood to determine optimum values of **a**, we can demand:

$$\frac{\partial}{\partial a_i} \int_{t_b}^{t_d} dt\, [R(t;\mathbf{a})]^2 = 2 \int_{t_b}^{t_d} dt\, R(t;\mathbf{a}) \frac{\partial R(t;\mathbf{a})}{\partial a_i} \overset{!}{=} 0 \tag{73}$$

for all $i = 1, 2, ..., n$. In the $n = 2$ approximation, we find the solution

$$\phi_2^{\mathrm{LS}}(t) = 1 - \frac{576}{611} t + \frac{190}{611} t^2 \tag{74}$$

Finite element solution to partial differential equations VIII

All these methods and another one described below can be classified by considering that they are only different in the weights they apply on the residual:

	Method	Weighting function
Collocation	$R(t_i;\mathbf{a}) = 0$	$W_i(t) = \delta(t_i)$
Subdomain	$\frac{1}{\Delta t_i}\int_{\Delta t_i} R(t;\mathbf{a})$	$W_i(t) = \begin{cases} 1 \forall t \text{ in } \Delta t_i \\ 0 \text{ otherwise} \end{cases}$
Least squares	$\int_{t_b}^{t_d} dt\, R(t;\mathbf{a})\frac{\partial R(t;\mathbf{a})}{\partial a_i} = 0$	$\frac{\partial R(t;\mathbf{a})}{\partial a_i}$
Galerkin	$\int_{t_b}^{t_d} dt\, R(t;\mathbf{a})\psi_i(t) = 0$	$\psi_i(t)$

Finite element solution to partial differential equations IX

Galerkin method: The most widely used weight in finite element methods uses basis functions of the trial solution as weights. Expanding $\tilde{\phi}_n(t)$ in terms of $n+1$ linearly independent basis functions we can write

$$\tilde{\phi}_n(t) = \psi_0(t) + \sum_{i=1}^{n} a_i \psi_i(t) \tag{75}$$

Here, we use the first term to satisfy the boundary conditions so that it enters $\tilde{\phi}_n(t)$ without a parameter. A possible choice of basis functions is

$$\psi_0(t) = 1 \qquad \psi_1(t) = t \qquad \psi_2(t) = t^2 \tag{76}$$

Now we try to fulfil the condition

$$\int_{t_b}^{t_d} dt\, R(t; \mathbf{a}) \psi_i(t) = 0 \tag{77}$$

Finite element solution to partial differential equations X

In the $n = 2$ case the Galerkin equations are

$$\int_0^1 dt\, R(t, \mathbf{a})\, t = \frac{1}{2} + \frac{5}{6} a_1 + \frac{11}{12} a_2 = 0 \tag{78}$$

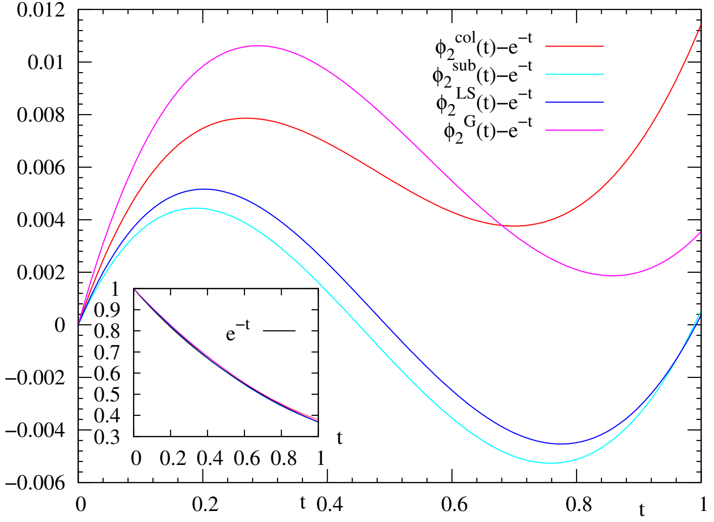
$$\int_0^1 dt\, R(t, \mathbf{a})\, t^2 = \frac{1}{3} + \frac{7}{12} a_1 + \frac{7}{10} a_2 = 0 \tag{79}$$

which we solve to get the solution

$$\phi_2^{\mathrm{G}}(t) = 1 - \frac{32}{35} t + \frac{2}{7} t^2 \tag{80}$$

In the domain $t = [0, 1]$, all four solutions represent the exact solution $\phi(t) = e^{-t}$ well (see inset of the following figure). Therefore, the figure shows the (small) deviations from the exact solution.

Finite element solution to partial differential equations XI



Eigenvalue problems I

Let us consider the time-independent Schrödinger equation

$$\hat{H}\psi_\alpha \equiv \left[-\frac{\hbar^2}{2\mu}\nabla^2 + V\right]\psi_\alpha = E_\alpha\psi_\alpha \tag{81}$$

The first term of the Hamiltonian $\hat{H}$ represents the kinetic energy, the second term V is the potential energy describing the interaction of different parts of the system. Solving this equation requires finding both eigenvalues E_α and eigenfunctions ψ_α. In principle, there are three ways of doing this:

1. Finding an analytic solution for simple forms of the potential V, like *e.g.* Hermite polynomials for the harmonic oscillator;
2. Treating the equation as a differential equation and solving it numerically (for example with the Numerov algorithm);
3. Introducing a basis set and applying matrix methods.

The last approach will be discussed today.

Eigenvalue problems II

First we have to find a complete set of **basis states** $\phi_1, \phi_2, \ldots, \phi_n$ so that any function can be expressed as a linear combination:

$$\psi_\alpha = \sum_{i=1}^{n} C_{\alpha\, i} \phi_i \tag{82}$$

with coefficients $C_{\alpha\, i}$ expressing the eigenvector ψ_α in terms of the ϕ_i. For convenience, we chose the basis states normalized and orthogonal:

$$\int d\tau\, \phi_i^* \phi_j \equiv \langle \phi_i | \phi_j \rangle = \delta_{ij} \tag{83}$$

where the integral runs over all independent variables.

Eigenvalue problems III

Choice of basis

As it is desirable to have as few basis states as possible, the basis should be chosen carefully on physical grounds. For example, in order to solve the anharmonic oscillator in one dimension for which the potential reads

$$V(x) = \frac{\mu}{2}\omega^2 x^2 + \epsilon\hbar\omega\left(\frac{\mu\omega}{\hbar}\right)^2 x^4 \qquad (84)$$

with typically small ϵ, it makes sense to choose the solution of the harmonic oscillator as a basis:

$$\phi_m(\rho) = \frac{1}{\sqrt{2^m m!\sqrt{\pi}}}\, e^{-\rho^2/2}\, H_m(\rho) \qquad (85)$$

Here, $H_m(\rho)$ are Hermite polynomials of degree m, and $\rho = x\sqrt{\mu\omega/\hbar}$.

Eigenvalue problems IV

The $\phi_m(\rho)$ are not eigenfunctions of the anharmonic oscillator because

$$\int_{-\infty}^{\infty} dx\, \phi_k(x)\, x^4\, \phi_l(x) \neq 0 \quad \text{for } |k-l| = 0, 2, 4, \ldots \tag{86}$$

But for small ϵ, each eigenfunction ψ_α is likely to be dominated by a single function ϕ_i. Then the main effect of the anharmonic term is to admix contributions from the basis functions $\phi_{i\pm 2}$. As contributions from $\phi_{i\pm 4}, \phi_{i\pm 6}, \ldots$ are likely to be smaller, we can choose the basis $\phi_{i-2}, \phi_i, \phi_{i+2}$ and approximate the solution with these. This reduces the anharmonic oscillator to a relatively simple problem: finding a linear combination of the three basis states that satisfies the Schrödinger equation.

Eigenvalue problems V

Once a basis set is chosen, the eigenvalue problem is reduced to finding expansion coefficients $C_{\alpha\, i}$. To see that, Eq. (81) is multiplied by ϕ_j^* and integrated over all independent variables:

$$\langle \phi_j | \hat{H} | \psi_\alpha \rangle = E_\alpha \langle \phi_j | \psi_\alpha \rangle \tag{87}$$

Now we use the expansion Eq. (82) and the orthogonality condition Eq. (83) to obtain the algebraic equation for the coefficients $C_{\alpha\, i}$:

$$\sum_{i=1}^{n} H_{ji} C_{\alpha\, i} = E_\alpha C_{\alpha\, j} \qquad \alpha = 1, 2, \ldots, n \tag{88}$$

where $H_{ji} \equiv \langle \phi_j | \hat{H} | \phi_i \rangle$.

Eigenvalue problems VI

In matrix notation, this is

$$\begin{pmatrix} H_{11} & H_{12} & \dots & H_{1n} \\ H_{21} & H_{22} & \dots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n1} & H_{n2} & \dots & H_{nn} \end{pmatrix} \begin{pmatrix} C_{\alpha\,1} \\ C_{\alpha\,2} \\ \vdots \\ C_{\alpha\,n} \end{pmatrix} = E_\alpha \begin{pmatrix} C_{\alpha\,1} \\ C_{\alpha\,2} \\ \vdots \\ C_{\alpha\,n} \end{pmatrix} \tag{89}$$

or bringing it into the form of linear equations for the $C_{\alpha\,i}$:

$$\begin{pmatrix} H_{11}-E_\alpha & H_{12} & \dots & H_{1n} \\ H_{21} & H_{22}-E_\alpha & \dots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n1} & H_{n2} & \dots & H_{nn}-E_\alpha \end{pmatrix} \begin{pmatrix} C_{\alpha\,1} \\ C_{\alpha\,2} \\ \vdots \\ C_{\alpha\,n} \end{pmatrix} = 0 \tag{90}$$

Eigenvalue problems VII

This equation only has a solution if

$$\det \begin{vmatrix} H_{11} - E_\alpha & H_{12} & \dots & H_{1n} \\ H_{21} & H_{22} - E_\alpha & \dots & H_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n1} & H_{n2} & \dots & H_{nn} - E_\alpha \end{vmatrix} = 0 \qquad (91)$$

Eq. (91) is called characteristic equation, and the eigenvalues are the roots of this equation. Once the roots of this polynomial in E_α are found, the coefficients $C_{\alpha\,1}, C_{\alpha\,2}, \dots, C_{\alpha\,n}$ can be found by solving the linear system of equations Eq. (90).

Eigenvalue problems VIII

Another way of approaching the problem posed by Eq. (89) is to find a transformation matrix **U** such that the similarity transformation

$$\mathbf{U}^{-1}\mathbf{H}\mathbf{U} = \mathbf{E} \tag{92}$$

results in a diagonal matrix **E**. The process of reducing a matrix **H** to a diagonal form is called **diagonalization**.
Another way of looking at the problem posed by Eq. (81) is by thinking in terms of a basis made from the eigenvectors $\{\psi_\alpha\}$.

Eigenvalue problems IX

In this basis the Hamiltonian matrix is diagonal, with the diagonal elements corresponding to the eigenvalues:

$$\langle \psi_\beta | \hat{H} | \psi_\alpha \rangle = \langle \psi_\beta | E_\alpha | \psi_\alpha \rangle = E_\alpha \delta_{\alpha\beta} \tag{93}$$

Thus, the transformation we are looking for is one that takes us from an arbitrary basis $\{\phi_i\}$ to the basis $\{\psi_\alpha\}$ made from the eigenvectors; in terms of matrices

$$\mathbf{U}\boldsymbol{\phi} = \boldsymbol{\psi} \tag{94}$$

with

$$\boldsymbol{\phi} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{pmatrix} \qquad \boldsymbol{\psi} = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix} \tag{95}$$

Eigenvalue problems X

From Eq. (94) one sees that the elements of **U** are the expansion coefficients of ψ_α in terms of ϕ_i:

$$\mathbf{U} = \begin{pmatrix} C_{1\,1} & C_{1\,2} & \dots & C_{1\,n} \\ C_{2\,1} & C_{2\,2} & \dots & C_{2\,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n\,1} & C_{n\,2} & \dots & C_{n\,n} \end{pmatrix} \tag{96}$$

In other words, **U** is made up out of the eigenvectors. It can be shown that **U** is unitary

$$\mathbf{U}^{\mathrm{T}} = \mathbf{U}^{-1} \tag{97}$$

and Eq. (94) is equivalent to Eq. (92).
There are many well-established methods for the computationally expensive task of diagonalizing a matrix.

Eigenvalue problems XI

- The simplest, the Jacobi method, is an iterative method for directly constructing a rotation matrix **U** that brings the matrix into diagonal form.
- Most efficient methods proceed in two steps, bringing a general matrix into tridiagonal or Hessenberg form in a finite number of steps and then diagonalizing the latter by iteration.
- In order to avoid unnecessary computation, diagonalization routines are usually specialized: They calculate some or all eigenvalues, no, few or all eigenvectors. They are for tridiagonal, for real symmetric, for real nonsymmetric, for complex Hermitian, for complex non-Hermitian matrices.

Some literature on basic numerical methods

Forman S. Acton, *Numerical methods that work*, Mathematical Society of America, Washington 1990.

Very detailed examples, a lot of critical commentary about thoughtless application of numerical methods, entertainingly written. State of the art of 1970, thus somewhat dated.

Samuel S. M. Wong, *Computational Methods in Physics and Engineering*, World Scientific, Singapore 1997.

Very detailed presentation with algorithms, program examples in Fortran.

Paul L. DeVries, *A First Course in Computational Physics*, John Wiley & Sons, New York 1994.

Accessible text with examples from physics, program pieces in Fortran.

Some literature on basic numerical methods

William H. Press *et al.*, *Numerical recipes in C, The Art of Scientific Computing*, Cambridge University Press, Cambridge 1992.

Exhausting presentation of methods and algorithm, with a library of useful C programs.

Alexander K. Hartmann, Heiko Rieger, *Optimization Algorithms in Physics*, Wiley-VCH, Berlin 2002.

Specialized text on optimization/minimization, with algorithms.

Jürgen Schnakenberg, *Algorithmen in der Quantentheorie und Statistischen Physik*, Verlag Zimmermann-Neufang, Ulmen 1995.

Well written text with theoretical physics applications.