UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL ATOMIC ENERGY AGENCY
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

H4.SMR/994-4

SPRING COLLEGES IN COMPUTATIONAL PHYSICS

19 May - 27 June 1997

ORDINARY DIFFERENTIAL EQUATIONS

C. REBBI
Boston University
Department of Physics
590 Commonwealth Ave.
Boston, Massachusetts 01125
U.S.A.

Preliminary notes from a course on Computational Physics - copyright by Claudio Rebbi, 1995 - for distribution to participants only.

2.3. Methods of integration

Our task is to solve the system of equations,

$$\dot{x}_k \equiv \frac{dx_k}{dt} = f_k\left(x_{k'}, t\right) , \quad k, k' = 1, 2, \ldots, M \tag{2.3.1}$$

by approximating the values that the variables, x_k, take for a discrete set of values, t_n, of the independent variable t. In most applications the t_n will be equally spaced, with spacing δ. Thus we will consider the sequence, $t = t_0$, $t_1 = t_0 + \delta$, $t_2 = t_0 + 2\delta, \ldots$, T. One can also generalize the methods we discuss here to the case where the spacing between successive times changes in the course of the evolution. We will discuss this in more detail later, but the necessity for this becomes obvious when one considers problems where characteristics of the object whose evolution we are following, such as its speed, change substantially as the system evolves. A typical case might be that of a gravitational system with orbits that are highly eccentric. In such cases, the speed of one of the gravitating objects may change by a large factor. In cases such as these, the accuracy of the integration of the evolution equations would vary as a function of time if the time discretization is held constant. A possible solution to this problem, which we will discuss at greater length later, is to use a time step with a magnitude which can change according to the value of some quantity which is being monitored.

We will also denote the values of the variables x_k by $x_{k,n}$, i.e. $x_{k,n} \equiv x_k(t_n)$. $x_{k,0}$ will be the given initial data. Likewise, $f_k(x_{i,n}, t_n)$ will be denoted by $f_{k,n}$ and, for economy of notation, we will refer to this term as the *force* term. Of course, $f_{k,n}$ does not necessarily coincide with the components of the machanical force. In most of the equations that we will use, the indices labelling the different variables will be irrelevant, and will thus often not be written, so that, for example, we will often write x_n as shorthand for $x_i(t_n)$.

All of the numerical methods described here are based on a suitable approximation to the derivatives $\dot{x}_{k,n}$, followed by an iterative solution of the algebraic equations that follow. When approximating a derivative, however, because of the finite size of the time step, the question immediately arises as to where in the interval should one calculate the force term, f. Specifically, if we want to evolve the system from $x = x_i$ to $x = x_i + \Delta t$, we could consider several possibilities; we could use f at x_i, or at $x_i + \Delta t$, or somewhere in between. Furthermore, we could even imagine methods which could improve these estimates by using information which is outside of this integration interval. As we shall see, these various possibilities give rise to many different methods for integrating the evolution equations.

2.3.1. The Euler method

The simplest approximation for the time derivatives is to use the forward difference approximation,

$$\dot{x}_n = \frac{x_{n+1} - x_n}{\delta} + \mathcal{O}(\delta) = f(x_n, t_n) , \tag{2.3.2}$$

to obtain,

$$x_{n+1} = x_n + f(x_n, t_n)\, \delta + \mathcal{O}(\delta^2) . \tag{2.3.3}$$

This equation, neglecting the unknown $\mathcal{O}(\delta^2)$, can be used iteratively to find x_n starting from x_0. The error in the individual steps, or *local error*, is $\mathcal{O}(\delta^2)$. However, going from the initial t_0 to a final $t = t_0 + T$ requires a number of steps, $N = T/\delta$. Since the local error in general will accumulate, we can expect a *global error* of order $N\delta^2 = \mathcal{O}(\delta)$ for fixed T. While this goes to zero for $\delta \to 0$, the approximation is rather poor; reducing the error in this procedure by a factor of two requires doubling the amount of computing. As you might expect, there are better ways of getting there. Also, as we shall see below, the inevitable violation of the conservation laws brought about by the discretization (well, not quite inevitable, as we will explain later) is particularly serious for this method. As we shall soon see, there is generally a conflict between accuracy and stability on the one hand, and computational efficiency on the other; the Euler method is surely the simplest from a computational point of view, but let us not lose sight of the fact that what we are interested in is to study physical phenomena, which this method is generally not adequate for. On its own, this method should essentially never be used.

2.3.2. Two–step method

We can obtain a better approximation by carrying out the expansion in Eq. (2.3.3) to a higher order in δ. It is convenient to use the notation f_n to denote $f(x_n, t_n)$. Expanding x_{n+1} about x_n to second order,

$$x_{n+1} = x_n + \dot{x}_n\,\delta + \frac{1}{2}\,\ddot{x}_n\,\delta^2 + \mathcal{O}\left(\delta^3\right) \tag{2.3.4}$$

and, using $\dot{x} = f$,

$$x_{n+1} = x_n + f_n\,\delta + \frac{1}{2}\dot{f}_n\,\delta^2 + \mathcal{O}\left(\delta^3\right). \tag{2.3.5}$$

We can now approximate $\dot{f}_n$ with its backward difference approximation

$$\dot{f}_n = \left(f_n - f_{n-1}\right)/\delta + \mathcal{O}\left(\delta\right)$$

to obtain,

$$x_{n+1} = x_n + \left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right)\delta + \mathcal{O}\left(\delta^3\right). \tag{2.3.6}$$

Iteration of this equation gives origin to the *two–step method* for integrating the system of differential equations. The global error in the integration over a finite evolution time T is now $\mathcal{O}(\delta^2)$, a substantial improvement over the Euler method.

Notice that the calculation of x_{n+1} relies on the values of the variables x at the two previous steps t_n and t_{n-1}. In general these are known, they are just the two previous values generated by the iteration procedure, which will be kept in memory (and, just as a side remark, it will be convenient to keep in memory also the value of f_{n-1} previously calculated, rather than repeat its calculation at step n). But there will be no previous value, x_{n-1}, at the initial step for $n = 0$. This means that this method, as well as many of the methods which we will subsequently encounter, must be initialized. We must devise some different procedure to calculate x_1, and this should be done with an accuracy compatible with the accuracy that we can expect for the subsequent steps. For instance, we could use the Euler method, but with a much smaller interval δ', to evolve the equations up to $t = t_1 = t_0 + \delta$. Or we could use a power series expansion of the equations (see also the next section) for $t = t_0$ to calculate x at $t = t_1$.

2.3.3. Taylor series method

In those cases where $\partial f/\partial x$ (by which we really mean all M^2 partial derivatives, $\partial f_i/\partial x_j$) and $\partial f/\partial t$ can be computed analytically in an efficient way, a useful method of integration, potentially of high accuracy, can be derived.

Writing,

$$x_{n+1} = x_n + f_n\delta + \dot{f}_n\frac{\delta^2}{2} + \mathcal{O}(\delta^3) \tag{2.3.7}$$

with

$$\dot{f}_n \equiv \frac{df}{dt} = \frac{\partial f}{\partial x}\dot{x}_n + \frac{\partial f}{\partial t}$$

with all derivatives evaluated at $t = t_n$.

This leads to,

$$x_{n+1} = x_n + f_n\delta + \left(\frac{\partial f}{\partial x}(x_n, t_n)\, f_n + \frac{\partial f}{\partial t}(x_n, t_n)\right)\frac{\delta^2}{2} + \mathcal{O}(\delta^3) \tag{2.3.8}$$

The above procedure can be carried out to higher order in δ to produce, correspondingly, evolution equations for x which are of higher order in δ. However, in all but the simplest cases, an explicit evaluation of the analytic expressions for the partial derivatives of f is cumbersome and computationally impractical. It is therefore usually more practical to simply find suitable approximations to the time derivatives. As a practical example we will now derive an immediate generalization of the two–step method we discussed above to a fourth–order method.

2.3.4. Adams–Bashforth four–step method

To derive this method, begin by expanding x_{n+1} about x_n to fourth order in δ, and using $\dot{x} = f$, to get,

$$x_{n+1} = x_n + f_n\delta + \dot{f}_n\frac{\delta^2}{2} + \ddot{f}_n\frac{\delta^3}{6} + \dddot{f}_n\frac{\delta^4}{24} + \mathcal{O}(\delta^5)\ . \tag{2.3.9}$$

We must now find approximations to the various time derivatives of f which will respect the overall $\mathcal{O}(\delta^5)$ correction to this equation; for example, we need $\ddot{f}$ accurate to $\mathcal{O}(\delta)$, but $\dot{f}$ accurate to $\mathcal{O}(\delta^2)$.

One way to do this is to construct the interpolating polynomial for f which passes through f_{n-3}, f_{n-2}, f_{n-1}, and f_n,

$$\begin{aligned} f(t) = {} & \frac{(t+\delta)(t+2\delta)(t+3\delta)}{6\delta^3} f_n - \frac{t(t+2\delta)(t+3\delta)}{2\delta^3} f_{n-1} \\ & + \frac{t(t+\delta)(t+3\delta)}{2\delta^3} f_{n-2} - \frac{t(t+\delta)(t+2\delta)}{6\delta^2} f_{n-3} \\ & + \mathcal{O}(\delta^4) \end{aligned} \tag{2.3.10}$$

If we now expand f for small $t \approx \delta$ about t_n,

$$f = f_n + \dot{f}_n t + \frac{\ddot{f}_n}{2} t^2 + \frac{\dddot{f}_n}{6} t^3 + \mathcal{O}(\delta^4) , \tag{2.3.11}$$

comparing this expansion with Eq. (2.3.10) gives us the time derivatives of f at t_n as the coefficients of a given power of t. Substituting the resulting expressions into Eq. (2.3.9) leads to,

$$x_{n+1} = x_n + (55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) \frac{\delta}{24} + \mathcal{O}(\delta^5) \tag{2.3.12}$$

Just as for the two–step method, Eq. (2.3.6), the four–step method is not self–starting, since x_{n+1} requires information which is three steps in the past. As we mentioned above, these methods require the use of a different, self–starting method to get to the point where the evolution formulas can be used.

Up to this point, the methods we have discussed evaluate the force term, f, in the evolution equation at either the beginning of the interval (the Euler method), or at the beginning of the interval, but with some correction given by the values of the derivatives of f there (the Taylor series method), or with corrections given by information on the force term from points preceding the integration interval (the multistep methods). Is it possible to use information which lies between the endpoints of the integration interval? There are, in fact, several ways of doing this. One of the most common, and certainly one of the most popular, is the basis for the Runge–Kutta methods, which we now describe.

2.3.5. Runge–Kutta methods

Consider using the central difference approximation to $\dot{x}$ at the midpoint of the interval, which we denote by, $x_{n+\frac{1}{2}}$, with step size $\delta/2$. That is,

$$x_{n+1} = x_n + \delta f\left(x_{n+\frac{1}{2}}, t_n + \frac{1}{2}\delta\right) + \mathcal{O}(\delta^3) . \tag{2.3.13}$$

The problem with this expression is that we don't know $x_{n+\frac{1}{2}}$, so we can't evaluate f. One way of finding this value approximately, if to use the Euler method to write

$$x_{n+\frac{1}{2}} = x_n + f_n \frac{\delta}{2} \tag{2.3.14}$$

or

$$f\left(x_{n+\frac{1}{2}}, t_n + \frac{1}{2}\delta\right) = f\left(x_n + f_n\frac{\delta}{2}, t_n + \frac{1}{2}\delta\right) + \mathcal{O}(\delta^3) . \tag{2.3.15}$$

We can think of this method (sometimes called the Euler–Cauchy method) as a two–step algorithm: First we evolve x from x_n to the midpoint of the interval, Eq. (2.3.14), and then we use the information at this point to evolve the system to x_{n+1}, Eq. (2.3.13).

2.3.6. Implicit methods

All the methods we have presented so far can develop instabilities if δ is taken too large. A class of methods that are intrinsically stable can be found, and, although these are generally more demanding from a computational point of view than the explicit methods we have looked at, they are sometimes desirable because of their stability properties. The terminology needs some clarification. Explicit methods are those in which all information required to compute x_{n+1} from x_n is contained explicitly in the recursion relation. By contrast, implicit methods contain some of this information implicitly in the force term. The simplest example of an implicit method is the *Backward Euler* method, where one evaluates the force term at the end of the interval,

$$x_{n+1} = x_n + \delta f\left(x_{n+1}, t_{n+1}\right) . \tag{2.3.26}$$

Another implicit method results when one uses as force term the average of the values f takes at the beginning and end of the interval, and $\dot{x}$ is approximated to $\mathcal{O}(\delta)$ by the central difference approximation, but using stepsize $\delta/2$,

$$x_{n+1} = x_n + \frac{\delta}{2}\left[f\left(x_n, t_n\right) + f\left(x_{n+1}, t_{n+1}\right)\right] . \tag{2.3.27}$$

To use implicit methods like the above, one needs to solve a recursion relation which is generally nonlinear. This is best done using iterative methods. One first obtains some estimate of x_{n+1}. Denote this first value by $x_{n+1}^{(0)}$. The recursion is then solved by using $x_{n+1}^{(0)}$ in the right hand side of a recursion such as Eq. (2.3.27), to obtain an improved value, $x_{n+1}^{(1)}$. Ths procedure is repeated until some convergence criterion is met. Better methods of solving the recursion exist; we use this one only as an illustration. The main advantage of implicit methods, and the only reason to use them alone as integrators, is that they are unconditionally stable. We will come back to this point later on when we study the stability of the various methods we have discussed. Used in conjunction with other integrators, however, implicit methods can be quite useful, as we now discuss.

2.3.7. Predictor–Corrector methods

One way of starting the recursion necessary to use an implicit method could be to obtain an approximation to x_{n+1} using one of the explicit methods we have discussed above. One can then improve this value by using an implicit method of the same order. One of the advantages of these methods is that they allow a continuous monitoring of the accuracy of the integration procedure. Let us mention at this point an issue which is probably on your mind. What do we mean by saying that the implicit method will *improve* the result of an explicit method of the same order? What we mean is that, with the explicit method one evolves the initial data from information at the beginning of the integration interval. The implicit methods, on the other hand, may include information from anywhere in the interval, including both endpoints. (The Runge–Kutta method is really an implicit method which we have

managed to make explicit by a series of partial steps). It is sensible to think that increasing the amount of information should improve the solution. This is always true. In general, and this is a case in point, there is a clear difference between the *order* of an approximation, and its *accuracy*. This is very important: high order doesn't always imply high accuracy.

Some combinations of predictor and corrector algorithms are preferred over others. The most popular combination is to use one of the multistep methods, particularly the fourth–order Adams-Bashforth method, Eq. (2.3.12), as predictor, and an implicit method of the same order, as corrector. The implicit method of choice in this case is called the *Adams–Moulton* four–step method. The procedure is then to first,

PREDICT:

$$x_{n+1} = x_n + (55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})\,\frac{\delta}{24} + \mathcal{O}\left(\delta^5\right)\,, \tag{2.3.28}$$

followed by,

CORRECT:

$$x_{n+1} = x_n + (9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})\,\frac{\delta}{24} + \mathcal{O}\left(\delta^5\right)\,. \tag{2.3.29}$$

2.3.8. Hamiltonian systems

Some special methods exist to integrate the equations of motion for separable Hamiltonian systems, that is, for systems whose dynamics can be derived from a Hamiltonian function of the form

$$\mathcal{H} = \mathcal{F}(\mathbf{q}) + \mathcal{G}(\mathbf{p}) \tag{2.3.30}$$

using Hamilton's equations,

$$\begin{aligned} \frac{\mathrm{d}p_i}{\mathrm{d}t} &= -\frac{\partial \mathcal{H}}{\partial q_i}\,, \\ \frac{\mathrm{d}q_i}{\mathrm{d}t} &= +\frac{\partial \mathcal{H}}{\partial p_i}\,. \end{aligned} \tag{2.3.31}$$

for the coordinates, q_i, and the momenta, p_i, which describe the system. A number of properties of these systems will serve to both constrain, as well as to suggest, possible integration algorithms.

For future reference, note that Eqs. (2.3.31) can be succinctly written as

$$\dot{\mathbf{z}} = \mathbf{J} \cdot \nabla_{\mathbf{z}} \mathcal{H}\,, \tag{2.3.32}$$

with,

$$\mathbf{z} \equiv \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix} \qquad \text{and} \qquad \mathbf{J} \equiv \begin{pmatrix} 0 & -\mathbf{I} \\ \mathbf{I} & 0 \end{pmatrix}, \tag{2.3.33}$$

where, in d dimensions, $\mathbf{I}$ is the $d \times d$–dimensional unit matrix.

A widely used method to study the evolution of Hamiltonian systems such as the above is called the *Leapfrog* method. The leapfrog method is applicable in a larger class of problems, of course; the fact that we have used the variables q and p, suggesting coordinates and momenta, doesn't have any profound meaning; the only restriction is on the form of the equations.

Define p at half–intervals,

$$p_{n+1/2} \equiv p\left((n+1/2)\,\delta\right). \tag{2.3.34}$$

Then the algorithm is defined by the recursion,

$$x_{n+1} = x_n + \mathcal{G}'\left(p_{n+1/2}\right)\delta + \mathcal{O}\left(\delta^3\right), \tag{2.3.35}$$

$$p_{n+3/2} = p_{n+1/2} + \mathcal{F}'\left(x_{n+1}\right)\delta + \mathcal{O}\left(\delta^3\right). \tag{2.3.36}$$

It is interesting to note that the step from n to $n+1$ can also be written as the succession of two symmetric and reversible half–steps. To see this, define

$$\begin{aligned} p_n &= p_{n-1/2} + \mathcal{F}'\left(x_n\right)\frac{\delta}{2} \\ &= p_{n+1/2} - \mathcal{F}'\left(x_n\right)\frac{\delta}{2}, \end{aligned} \tag{2.3.37}$$

for $n = 0$ one can define $p_{1/2} \equiv p_0 + \mathcal{F}'(x_0)\delta/2$ as the initialization step. The evolution now proceeds as follows. We evolve,

$$\left(x_n,\, p_n\right) \quad \rightarrow \quad \left(x_{n+1},\, p_{n+1}\right), \tag{2.3.38}$$

in three steps:

$$\begin{aligned} x_n,\, p_n \quad &\rightarrow \quad x_n,\, p_{n+1/2} = p_n + \mathcal{F}'\left(x_n\right)\frac{\delta}{2} \\ x_n,\, p_{n+1/2} \quad &\rightarrow \quad x_{n+1} = x_n + \mathcal{G}'\left(p_{n+1/2}\right)\delta,\, p_{n+1/2} \\ x_{n+1},\, p_{n+1/2} \quad &\rightarrow \quad x_{n+1},\, p_{n+1} = p_{n+1/2} + \mathcal{F}'\left(x_{n+1}\right)\frac{\delta}{2}, \end{aligned} \tag{2.3.39}$$

In practice, to evolve x, the intermediate values of p at the interval endpoints, p_n, and p_{n+1} are not really needed, since the two successive steps,

$$\begin{aligned} p_{n+1/2} \quad &\rightarrow \quad p_{n+1} = p_{n+1/2} + \mathcal{F}'\left(x_{n+1}\right)\frac{\delta}{2} \\ p_{n+1} \quad &\rightarrow \quad p_{n+3/2} = p_{n+1} + \mathcal{F}'\left(x_{n+1}\right)\frac{\delta}{2}, \end{aligned} \tag{2.3.40}$$

could be combined into a single (leapfrog) step,

$$p_{n+1/2} \quad \rightarrow \quad p_{n+3/2} = p_{n+1/2} + \mathcal{F}'\left(x_{n+1}\right)\delta. \tag{2.3.41}$$

However, the first representation is useful for two reasons. First, it is adequate for the first ($x_0 p_0 \rightarrow x_0 p_{1/2}$) step, and the last ($x_n p_{n-1/2} \rightarrow x_n p_n$) step; second, because of its conceptual clarity in showing what are the most important features of this algorithm, namely that it is reversible, that is,

$$\begin{array}{lll} \text{if} & x_n,\, p_n & \rightarrow \quad x_{n+1},\, p_{n+1} \\ \text{then} & x_{n+1},\, -p_{n+1} & \rightarrow \quad x_n,\, -p_n\,. \end{array}$$

and because the three transformations in Eq. (2.3.39) are canonical in the sense of classical mechanics, and therefore preserve the volume element $dx\,dp$ (which really refers to the M-dimensional phase-space volume $dx_1 \cdots dx_{\frac{M}{2}}\, dp_1 \cdots dp_{\frac{M}{2}}$).

Another popular method, very commonly used in N-body simulations is called the *Verlet* algorithm. It is applicable in the case where the equations we wish to solve are of the form,

$$\dot{x} = p\,, \tag{2.3.42}$$

and

$$\dot{p} = f(x)\,, \tag{2.3.43}$$

that is, the special case where $\mathcal{G}(\mathbf{p})$ in Eq. (2.3.30) is quadratic, where the system is equivalent to,

$$\ddot{x} = f(x)\,. \tag{2.3.44}$$

From

$$\begin{aligned} x_{n-1} &= x_n - p_n\delta + \frac{1}{2}\dot{p}_n\delta^2 + \mathcal{O}(\delta^3) \\ x_{n+1} &= x_n + p_n\delta + \frac{1}{2}\dot{p}_n\delta^2 + \mathcal{O}(\delta^3)\,, \end{aligned} \tag{2.3.45}$$

one can cancel both the linear and cubic terms in δ to get,

$$x_{n+1} = 2x_n - x_{n-1} + f(x_n)\,\delta^2 + \mathcal{O}(\delta^4)\,, \tag{2.3.46}$$

and

$$p_n = \frac{(x_{n+1} - x_{n-1})}{2\delta} \tag{2.3.47}$$

Notice that these recursions are, in fact, simply the resulting recursions for the direct implementation of Eq. (2.3.44), using,

$$\ddot{x}_n = \frac{x_{n+1} + x_{n-1} - 2x_n}{\delta^2} + \mathcal{O}(\delta^2) . \tag{2.3.48}$$

In a moment we will derive a different algorithm, of $\mathcal{O}(\delta^6)$, to solve second–order equations such as Eq. (2.3.44) directly, but first we discuss the *Velocity Verlet* method, which is the version of the Verlet method used in practice.

Considering still the restricted form of the equations, Eq. (2.3.42) and Eq. (2.3.43), the Velocity Verlet method evolves x and p according to

$$x_{n+1} = x_n + p_n\delta + \frac{1}{2}f(x_n)\delta^2 + \mathcal{O}(\delta^3) , \tag{2.3.49}$$

followed by

$$p_{n+1} = p_n + \frac{1}{2}[f(x_n) + f(x_{n+1})]\delta + \mathcal{O}(\delta^3) . \tag{2.3.50}$$

However, notice that, if we redefine

$$p_n + \frac{1}{2}f(x_n)\delta \equiv p_{n+1/2} , \tag{2.3.51}$$

then the above equations reduce to

$$x_{n+1} = x_n + p_{n+1/2}\delta , \tag{2.3.52}$$

and

$$p_{n+3/2} \equiv p_{n+1} + \frac{1}{2}f(x_{n+1})\delta = p_{n+1/2} + f(x_{n+1})\delta , \tag{2.3.53}$$

which is just the leapfrog method we have already derived.

The last method we present in this subsection, called the *Numerov–Cowling* algorithm, is a high–order integrator for the second–order equation, $\ddot{x} = f(x,t)$.

The algorithm is most easily derived by considering the Taylor series expansions for $x(t+\delta)$ and $x(t-\delta)$ to $\mathcal{O}(\delta^4)$. If one adds the two expansions, one finds that

$$\frac{x_{n+1} + x_{n-1} - 2x_n}{\delta^2} = f_n + \dddot{x}\frac{\delta^2}{12} + \mathcal{O}(\delta^4) . \tag{2.3.54}$$

But

$$\dddot{x} = \ddot{f} = \frac{f_{n+1} + f_{n-1} - 2f_n}{\delta^2} , \tag{2.3.55}$$

therefore,

$$x_{n+1} - 2x_n + x_{n-1} = f_n\delta^2 + \frac{f_{n+1} + f_{n-1} - 2f_n}{12}\delta^2 + \mathcal{O}(\delta^6) , \tag{2.3.56}$$

leaving, finally,

$$x_{n+1} - \frac{f_{n+1}}{12}\delta^2 = 2x_n - x_{n-1} + \frac{5}{6}f_n\delta^2 + \frac{1}{12}f_{n-1}\delta^2 + \mathcal{O}(\delta^6) . \tag{2.3.57}$$

In general, this is very accurate, implicit, algorithm. Notice, however, that it can be made explicit if the equation is linear. That is, if

$$f(x,t) = K(t)\,x\,, \tag{2.3.58}$$

then the left hand side of Eq. (2.3.57) reduces to

$$\left(1 - \frac{K(t_{n+1})\,\delta^2}{12}\right) x_{n+1}\,, \tag{2.3.59}$$

and the recursion becomes explicit.

Later on we will use the Numerov–Cowling method to solve the Schrödinger equation,

$$-\frac{\hbar^2}{2m}\psi''(x) + V(x)\,\psi(x) = E\psi(x)\,, \tag{2.3.60}$$

which is just of the required form for this algorithm, with the function K in Eq. (2.3.58) given by

$$\frac{2m}{\hbar^2}\,(V(x) - E)\,. \tag{2.3.61}$$

2.4. Accuracy considerations

We have derived several methods which, though different in other details, can be classified according to their order, that is, according to the power of the time step to which they are accurate. What if we want higher accuracy still? Generally, as we have stressed, simply increasing the order of an algorithm doesn't always pay off. Furthermore, increasing the order of an algorithm always increases the computational cost as well. As we shall see, there exists a very powerful procedure which increases the accuracy of any given method, at a very small computational cost. This method, called *Richardson's* extrapolation, can also be used to improve the accuracy of quadrature methods (where it is called *Romberg* integration). The idea behind the method is very simple.

Suppose we have computed the solution to an equation, $x_\delta(t)$, for a given stepsize δ, which is accurate to order δ^p. Assuming that the *exact* solution of the continuum problem, which we shall call, $X(t)$, has an expansion in δ, so that,

$$X(t) - x_\delta(t) = c_1\delta^{p+1} + c_2\delta^{p+2} + \cdots\,, \tag{2.4.1}$$

where the coefficients, c_k are independent of δ, but otherwise depend on x and its derivatives evaluated at some point. Assume now that we use an integration method of order p, which is consistent with the above expansion, to advance the solution from t to $t+\delta$. We use this method twice, once to go from t to $t+\delta$ using a stepsize δ, and once using two steps of stepsize $\delta/2$. Since the c_k are independent of δ, the

two approximate solutions at $t+\delta$ are related to the exact solution, $X(t+\delta)$ by the following expressions,

$$\begin{aligned} X(t+\delta) - x(t;\delta) &= \omega\delta^{p+1} + \epsilon_T(\delta) , \\ X(t+\delta) - x\left(t;\frac{\delta}{2}\right) &= 2\omega\left(\frac{\delta}{2}\right)^{p+1} + 2\epsilon_T\left(\frac{\delta}{2}\right) \end{aligned} \tag{2.4.2}$$

where $\epsilon_T(\delta)$ stands for the total truncation error above $\mathcal{O}(\delta^{p+1})$, and where the extra factor of 2 in the second expression above comes from the fact that it is composed of two steps of size $\delta/2$. It follows from the above that the combination,

$$\tilde{x} = \frac{1}{2^p - 1}\left[2^p x\left(t;\frac{\delta}{2}\right) - x(t;\delta)\right] \tag{2.4.3}$$

cancels the leading error at $t+\delta$, which was $\mathcal{O}(\delta^{p+1})$, and further reduces the truncation error by a factor $\sim 2^{-p}$. Furthermore, this process can obviously be iterated (by successive refinement of the interval) to the numerical resolution of the computer in a few steps.

It should be clear that an entirely analogous procedure can be used to improve the results of a numerical quadrature algorithm.

Notice, however, that if we have warned you about interpolation, we will warn you doubly about extrapolation. It is trivial to find examples where the extrapolation we have described above will fail. The most obvious case is where the error term is not, in fact, of the form shown in Eq. (2.4.1). Careful use of these techniques, though, leads to very powerful tools.

Another aid in determining the accuracy of a solution to an initial value problem is to integrate the equations from $t = t_0$ up to some time $t = T$, and then to reverse the procedure by integrating the equations back to t_0. The accuracy with which this process reproduces the initial data is a very good indicator of the accuracy of the integrating method.

Finally, it is often useful, and sometimes essential, to compute the time–evolution of quantities which are constants of motion for the continuum problem, and monitor the extent to which the discretization procedure violates the conservation laws. It is possible, in fact, to modify a given integration method, so that it is forced to satisfy one or more conservation laws exactly, as we will show later.

2.5. Stability considerations

Up to this point, our main preocupation has been with deriving algorithms to integrate differential equations which are both efficient from a computational point of view, as well as numerically accurate locally. In this section we will explore a related issue which is sometimes of equal importance. The point we wish to explore here is how large can the discretization stepsize be made such that one can still expect reasonable results or whether the results for a given δ are stable after many iterations of a given algorithm.

Suppose we have an exact solution, $X(t)$, to the equation

$$\dot{X}(t) = F(X(t), t) . \tag{2.5.1}$$

We would like to explore the stability of the solution under small perturbations. Write

$$x(t) = X(t) + s(t) \text{ , where } \quad s(t) \ll 1 , \tag{2.5.2}$$

and expand Eq. (2.5.1),

$$\begin{aligned} \dot{X}(t) + \dot{s}(t) &= F(X+s, t) \\ &= F(X,t) + \frac{\partial F}{\partial X}(X,t)\, s(t) . \end{aligned} \tag{2.5.3}$$

It follows that $s(t)$ satisfies the linear equation,

$$\dot{s}(t) = \frac{\partial F}{\partial X}(X,t)\, s(t) . \tag{2.5.4}$$

Similarly, in the discretized case, where $s(t_n) \equiv s_n$, will satisfy a linear recursion relation,

$$s_{n+1} = \mathcal{A} s_n . \tag{2.5.5}$$

For a system of N coupled equations, s is an N-dimensional vector, and $\mathcal{A}$ is an $N \times N$ matrix. For a general problem the form of the matrix, $\mathcal{A}$, can depend on the time step, so that the solution of Eq. (2.5.5) will be given by,

$$s_n = \mathcal{A}_n \mathcal{A}_{n-1} \cdots \mathcal{A}_1 s_0 . \tag{2.5.6}$$

It is interesting and instructive to consider the case of a linear equation, where $\mathcal{A}$ is fixed, in which case the solution of Eq. (2.5.5) is given by,

$$s_n = (\mathcal{A})^n \, s_0 , \tag{2.5.7}$$

and $|s|$ will remain bounded if, and only if, all eigenvalues of $\mathcal{A}$ are less or equal to one. If any of the eigenvalues of $\mathcal{A}$ exceeds unity, the error in the approximate solution, x_n will be amplified by successive integration steps. There is more to stability than this, but this will be sufficient for our present purposes. Our interest, then, is to determine the conditions under which the methods we have described are stable in the above sense, namely that the method does not amplify the error term associated with a given approximation. These properties will, in general, be specific not only to the

integration algorithm, but also to the particular differential equation this algorithm is applied to.

As a first example, we will consider the dynamical equations for a one–dimensional harmonic oscillator,

$$\begin{aligned} \dot{x} &= p \\ \dot{p} &= -x\,, \end{aligned} \tag{2.5.8}$$

where we have rescaled p and x so that the oscillator frequency is unity. Using this system, we will explore the stability of the Euler, leapfrog, and multistep algorithms we have derived.

2.5.1. Euler method

The method can be written as

$$\begin{aligned} x_{n+1} &= x_n + p_n\delta\,, \\ p_{n+1} &= p_n - x_n\delta\,. \end{aligned} \tag{2.5.9}$$

In terms of the two–component vector,

$$\mathbf{y} \equiv \begin{pmatrix} x \\ p \end{pmatrix}\,, \tag{2.5.10}$$

the above recurrence can be written as,

$$\mathbf{y}_{n+1} \equiv \begin{pmatrix} x_{n+1} \\ p_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & \delta \\ -\delta & 1 \end{pmatrix} \begin{pmatrix} x_n \\ p_n \end{pmatrix} \equiv \mathcal{A}\mathbf{y}_n\,. \tag{2.5.11}$$

We can easily find the eigenvalues of $\mathcal{A}$ by solving,

$$\begin{vmatrix} 1-\lambda & \delta \\ -\delta & 1-\lambda \end{vmatrix} = 0\,, \tag{2.5.12}$$

which gives,

$$(1-\lambda)^2 + \delta^2 = 0\,, \tag{2.5.13}$$

or,

$$\lambda = 1 \pm i\,\delta\,. \tag{2.5.14}$$

This implies that, as expected, the recurrence reproduces the continuum solution, $\exp(\pm i\,t)$, to order δ. However, because $|\lambda| > 1$, the approximation will deteriorate with time. Furthermore, notice that $|\lambda| = \sqrt{(1+\delta^2)} \approx 1 + \delta^2/2$, so that the energy, which is proportional to $|\mathbf{y}|^2 = x^2 + p^2$, *increases* linearly with the duration T of the evolution with a coefficient proportional to δ. As announced before, the Euler algorithm is one to use only with extreme care.

2.5.2. Leapfrog method

The algorithm is given by,

$$\begin{aligned} x_{n+1} &= x_n + p_{n+1/2}\delta\,, \\ p_{n+3/2} &= p_{n+1/2} - x_{n+1}\delta\,, \end{aligned} \tag{2.5.15}$$

which can be written as

$$\begin{pmatrix} 1 & 0 \\ \delta & 1 \end{pmatrix} \begin{pmatrix} x_{n+1} \\ p_{n+3/2} \end{pmatrix} = \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_n \\ p_{n+1/2} \end{pmatrix}, \tag{2.5.16}$$

or, defining,

$$\mathbf{y_n} \equiv \begin{pmatrix} x_n \\ p_{n+1/2} \end{pmatrix},$$

we can write,

$$\begin{pmatrix} 1 & 0 \\ \delta & 1 \end{pmatrix} \mathbf{y}_{n+1} = \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix} \mathbf{y}_n\,. \tag{2.5.17}$$

This can be easily solved to give,

$$\begin{aligned} \mathbf{y}_{n+1} &= \begin{pmatrix} 1 & 0 \\ -\delta & 1 \end{pmatrix} \begin{pmatrix} 1 & \delta \\ 0 & 1 \end{pmatrix} \mathbf{y}_n\,, \\ &= \begin{pmatrix} 1 & \delta \\ -\delta & 1-\delta^2 \end{pmatrix} \mathbf{y}_n\,. \end{aligned} \tag{2.5.18}$$

The eigenvalues of the evolution matrix are then given by the solution of

$$\begin{aligned} \begin{vmatrix} 1-\lambda & \delta \\ -\delta & 1-\delta^2-\lambda \end{vmatrix} &= 0\,, \\ 1-(2-\delta^2)\,\lambda + \lambda^2 &= 0\,, \end{aligned} \tag{2.5.19}$$

leading finally to,

$$\begin{aligned} \lambda &= 1 - \frac{\delta^2}{2} \pm \sqrt{-\delta^2 + \frac{\delta^4}{4}} \\ &= 1 \pm i\,\delta - \frac{\delta^2}{2} + \cdots \end{aligned} \tag{2.5.20}$$

which agrees, as expected, with the continuum evolution result, $\exp(\pm i\,t)$, including the $\mathcal{O}(\delta^2)$ term. Notice also that, if $\delta^2 > \delta^4/4$, that is, if $\delta < 2$, the term in the square root above becomes negative, and one finds,

$$|\lambda|^2 = 1\,. \tag{2.5.21}$$

This implies that, for any $\delta < 2$, the leapfrog method is stable for this set of equations; it also implies that this method conserves energy exactly, for the special case of the harmonic oscillator (up to roundoff errors, of course) for $\delta < 2$.

2.5.3. Two–step method

The evolution equations for the method are given by,

$$\begin{aligned} x_{n+1} &= x_n + \frac{3\delta}{2} p_n - \frac{\delta}{2} p_{n-1} \\ p_{n+1} &= p_n - \frac{3\delta}{2} x_n + \frac{\delta}{2} x_{n-1} \,. \end{aligned} \tag{2.5.22}$$

These can be written compactly for the four–component vector,

$$\mathbf{y}_n \equiv \begin{pmatrix} x_n \\ p_n \\ x_{n-1} \\ p_{n-1} \end{pmatrix} ,$$

leading to,

$$\mathbf{y}_{n+1} = \begin{pmatrix} 1 & \frac{3\delta}{2} & 0 & -\frac{\delta}{2} \\ -\frac{3\delta}{2} & 1 & \frac{\delta}{2} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{y}_n \tag{2.5.23}$$

The eigenvalues are given by,

$$\begin{aligned} \lambda_{1,2} &= 1 \pm i\,\delta - \frac{\delta^2}{2} + \cdots \\ \lambda_{3,4} &= \pm i \frac{\delta}{2} + \cdots \end{aligned} \tag{2.5.24}$$

therefore,

$$\begin{aligned} |\lambda_{1,2}| &= 1 + \mathcal{O}\left(\delta^4\right) + \cdots , \\ |\lambda_{3,4}| &= \frac{\delta}{2} + \cdots . \end{aligned} \tag{2.5.25}$$

Although two of the eigenvalues correspond to the correct solution of the equations to order δ^2, the other two eigenvalues correspond to other solutions of the recurrence equations (called *parasitic* solutions) are not solutions of the diferential equations at all. This is a general problem with multistep methods; although the differential equations have a unique solution for a given initial state (one arbitrary constant for each one of the differential equations), the recurrence relation for an M-step method has a general solution which is a linear combination of M independent solutions. Only one of these solutions, generally called the *principal* solution, corresponds to a solution of the differential equations. The remaining $M-1$ parasitic solutions can, in general, dominate over the principal solution (especially for large M), and render the method useless. Notice that the growth in the number of parasitic solutions in these methods is an independent source of instability, quite unrelated to the instability which we observed for the Euler method. The desired solution, corresponding to $\lambda_{1,2}$, is in itself unstable, with $|\lambda_{1,2}| > 1$, so that the global error in this method grows with time, as in the Euler method, but with a smaller coefficient (for $\delta < 1$).

2.5.4. Pitfalls

Stability considerations are usually of paramount importance, and ignoring this issue can result in catastrophe. As a further example of what an ill-chosen method can do, consider one which might seem attractive from the point of view of both computational efficiency, as well as being of $\mathcal{O}(\delta^2)$. The method is based on the central-difference approximation. Despite its superficial attractions, the approximation, as we shall soon see, is not as good as it appears to be.

Consider solving $\dot{x} = -x$ using the central-difference approximation to the first derivative,

$$\frac{1}{2\delta}(x_{n+1} - x_{n-1}) = -x_n\,, \tag{2.5.26}$$

corresponding to the matrix evolution,

$$\begin{pmatrix} x_{n+1} \\ x_n \end{pmatrix} = \begin{pmatrix} -2\delta & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_n \\ x_{n-1} \end{pmatrix}. \tag{2.5.27}$$

The eigenvalues are given by,

$$\lambda_{1,2} = -\delta \pm \sqrt{\delta^2 + 1}\,. \tag{2.5.28}$$

Expanding the term in the square root we see that one of the eigenvalues has what seems to be the correct expansion of the solution of $\dot{x} = -x$ to $\mathcal{O}(\delta^2)$,

$$\lambda_1 = 1 - \delta + \frac{\delta^2}{2} + \cdots\,. \tag{2.5.29}$$

However, there is also an *unstable* eigenvalue,

$$\lambda_2 = -\left(1 + \delta + \frac{\delta^2}{2} + \cdots\right) < -1 \qquad \text{for all } \delta > 0\,, \tag{2.5.30}$$

which is the expansion of the spurious solution, $-e^{+\delta}$ to $\mathcal{O}(\delta^2)$. Because $|\lambda_2| > 1$ for any δ, this algorithm is intrisically unstable.

2.5.5. Implicit methods

As a final example, we will investigate the stability of an implicit method (for the test case $\dot{x} = -x$). Consider the method,

$$x_{n+1} = x_n - \frac{\delta}{2}\left(x_{n+1} + x_n\right), \tag{2.5.31}$$

which can be solved for x_{n+1},

$$x_{n+1} = \frac{1 - \frac{\delta}{2}}{1 + \frac{\delta}{2}} x_n . \tag{2.5.32}$$

If we expand this solution for small δ we see that, as expected, it agrees with the exact solution to $\mathcal{O}(\delta^2)$. Moreover, though, the algorithm is stable for *any* value of δ. This property, alluded to before, is what makes implicit algorithms most useful. This situation is to be contrasted with the explicit algorithms, where, as we have seen, large values of δ will in general make the iteration unstable. For example, the application of the simple Runge–Kutta method, Eq. (2.3.24), to the same equation, $\dot{x} = -x$, leads to,

$$x_{n+1} = \left(1 - \delta + \frac{\delta^2}{2}\right) x_n , \tag{2.5.33}$$

which will grow exponentially when $\delta > 2$. Of course, you think that this is irrelevant, since δ is supposed to be a small number. This objection is valid in the case of ordinary differential equations. However, as we shall see later on, essentially the same stability analysis, when applied to the case of partial differential equations. In this case, the results are similar but the quantity which must be kept small to avoid instability is not simply δ, the discretization step, but rather $\lambda\delta$, where λ denote the eigenvalues of some discretized differential operator. Therefore, even if δ is small, the quantity controlling the stability of a recurrence, $\lambda\delta$, can easily exceed the stability bounds. We will return to this point when we study numerical methods for the solution of partial differential equations.

In the case of the solution of partial differential equations, the extra computational effort involved in using these methods is sometimes a small price to pay if by using them one can solve systems for which explicit methods are unstable.

Chapter 3.

Boundary value problems

In contrast with the case we discussed in Chapter 2, where we were interested in the time evolution of initial data as an example of initial value problems, in this Chapter we will explore a different class of problems whose solution requires initial data to be given at two or more points. As the main physical example of this type of problem, we will solve the time–independent Schrödinger equation for a bound state of heavy (non–relativistic) particles.

3.1. The Schrödinger equation

Our present understanding of the physics of the strong interactions between elementary particles is embodied in a quantum field theory called Quantum Chromodynamics (QCD). According to this theory, the particles which feel the strong force, called *hadrons*, are described in terms of elementary constituents called *quarks* and *gluons*. Examples of hadrons are the *baryons*, such as the proton (p) and the neutron (n), and the *mesons*, such as the pion (π), K mesons, and the rho meson (ρ).

Quarks and gluons are described by quantized, relativistic fields and hadrons emerge as bound states of quarks, kept together by the forces which result out of the highly non–linear dynamics of the gluonic field. In general, the problem of describing the interactions of quarks and gluons is quite complex and, if approached from a computational point of view, requires considering systems with an extremely large number of degrees of freedom. Very substantial simplifications however occur if one consider the case of heavy quarks, quarks whose mass (converted to energy units according to $E = mc^2$) is much larger then the typical energy scale of gluonic interactions. Then the motion of the quarks becomes non–relativistic and the complicated

interactions due to the gluonic fields can be accounted for by an effective potential between quarks. To a very good first approximation, this effective potential is given by the superposition of a short-range Coulombic potential, $-\frac{\alpha}{r}$, and a long-range, confining, linearly increasing potential τr.

The masses and other properties of mesones consisting of a bound state of a heavy quark and a heavy antiquark can thus be found by solving the Schrödinger equation that describes the $\overline{q}q$ bound states in the presence of effective Coulomb plus linear potential.

In particle physics one customarily adopts units where c, the speed of light, and $\hbar$, the reduced Planck constant, are set to unity. In these units, lengths, times, masses, momenta, energies are all expressed in terms of (suitable powers) of the unit for any of these quantities. We will use an energy unit, the *GeV* (10^9 electron volts) as a fundamental unit; with $c = 1$ mass and momentum are also given in units of energy, and we will measure these in GeV as well. In these units, where $c = \hbar = 1$, the unit of length is the inverse of that of energy. In *fermis* (fm), with $1\,\text{fm} = 10^{-15}$ m, $(\text{GeV})^{-1} = 0.197327053(59)\,\text{fm} \approx 0.2\,\text{fm}$.

The masses of the quarks are not uniquely defined, since quarks do not exist as free particles. However one can assign values of mass to the various quarks (which are definition dependent) what are called Lagrangian masses (that is, the masses that appear in the fundamental Lagrangian describing the theory). From phenomenology, these masses have the approximate values given in table below.

Symbol	Name	Charge	Mass (GeV)
u	*up*	$+\frac{2}{3}e$	0.005
d	*down*	$-\frac{1}{3}e$	0.01
s	*strange*	$-\frac{1}{3}e$	0.15
c	*charm*	$+\frac{2}{3}e$	1.5
b	*bottom*	$-\frac{1}{3}e$	5.0

Table 3.1.1: Mass and electric charge of the quarks

The mases and quark content of a few hadrons are given in the following table,

The typical range of energies due to gluonic interactions is of the oder of a few hundred MeV, that is, in the 0.1–1 GeV domain. Thus a non–relativistic approximation may work resonably well for the bound states of c and $\overline{c}$, and should work very well for bound states of b and $\overline{b}$. Among the bound states of the heavy quarks, the two most prominent are the J/Ψ and Υ mesons, whose properties are summarized in the table below,

The Schrödinger equation for the bound states of c and $\overline{c}$ or of b and $\overline{b}$ is

$$\left[-\frac{\hbar^2\nabla^2}{2m_R} + V(r)\right]\Psi = E\Psi \tag{3.1.1}$$

Symbol	Name	Quark content	Mass (GeV)
p	*proton*	uud	0.9383
n	*neutron*	udd	0.9396
Λ	*lambda*	uds	1.1156
$\pi^{\pm}$	*charged pions*	$u\bar{d} + \bar{u}d$	0.1396
π^0	*neutral pion*	$\bar{u}u + \bar{d}d$	0.1350
η	*eta*	$\bar{s}s + \bar{u}u + \bar{d}d$	0.5488
$K^{\pm}$	*charged kaons*	$u\bar{s} + \bar{u}s$	0.4936

Table 3.1.2: Mass and quark content of a few hadrons

Symbol	Quark content	Mass (GeV)
J/Ψ	uud	3.0969
Υ	udd	9.4603

Table 3.1.3: Mass and quark content of two heavy–quark bound states

where m_R is the reduced mass of the system. With a bound state of two identical quarks $m_R = \frac{m_q}{2}$. The effective potential we want to use is therefore (including the rest energy of the quarks, $2m_q$),

$$V(r) = 2m_q - \frac{\alpha}{r} + \tau r \,. \tag{3.1.2}$$

The coefficient in the Coulomb term in the potential, α, depends on the range of the interaction (the average separation between q and $\bar{q}$) This comes about from the fact that the above is not the exact interaction potential, but only an approximation. α is dimensionless and its typical values are in the range $0.2 - -0.5$.

The coefficient of the linear term in the potential, τ, is called the string tension and can be determined from the phenomenology of light mesons (from the relation between mass and angular momentum). τ is measured in energy/length. With [length] = [energy]$^{-1}$, τ is thus measured in units of [energy]^2. A good phenomenological value for the string tension is,

$$\tau \approx (420\text{MeV})^2 = 0.1764\ \text{GeV}^2 \approx 0.88\text{GeV/fm}\,.$$

The equation we want to solve is thus (remember $\hbar = 1$),

$$\left[-\frac{\nabla^2}{m_q} + \left(2m_q - \frac{\alpha}{r} + \tau r\right)\right]\Psi = E\,\Psi\,. \tag{3.1.3}$$

The eigenvalues of the above equation, E, will give the rest energies, that is, the masses, of the $q\bar{q}$ bound states.

The angular part of the wave function $\Psi(r,\vartheta,\varphi)$ can be factored out by writing

$$\Psi(r,\vartheta,\varphi) = \frac{U(r)}{r} Y_{lm}(\vartheta,\varphi) \ , \tag{3.1.4}$$

where Y_{lm} is a spherical harmonic with angular momentum l.

In the angular representation, Eq. (3.1.3) thus reduces to

$$-\frac{1}{m_q}\frac{d^2U(r)}{dr^2} + \frac{l(l+1)}{m_q r^2}U(r) + \left(2m_q - \frac{\alpha}{r} + \tau r\right)U(r) = EU(r) \ . \tag{3.1.5}$$

with the radial wave function, $U(r)$, satisfying some definite boundary conditions at $r = 0$ and $r = \infty$, which we now deduce.

For $r = 0$ the dominant terms in the equation are

$$-\frac{1}{m_q}\frac{d^2U}{dr^2}$$

and

$$\frac{l(l+1)U}{m_q r^2} \ .$$

If we set

$$-\frac{d^2U}{dr^2} + \frac{l(l+1)U}{r^2} = 0$$

we get,

$$U \propto r^{l+1} \quad \text{or} \quad U \propto r^{-l} \ .$$

The second choice produces a singular wave function Ψ and we conclude that the boundary condition for $r = 0$ is that U must vanish at the origin and, indeed, behave as r^{l+1} in the neighborhood of this point.

For $r \to \infty$ the term $\tau r U(r)$ dominates and the equation reduces to

$$-\frac{1}{m_q}\frac{d^2U}{dr^2} + \tau r U + \ldots = 0 \ ,$$

where the ellipsis stand for other subdominant terms. The equation

$$U'' - (\text{const}) \times rU = 0 \ ,$$

cannot be solved in terms of elementary transcendental functions (its solution can be expressed in terms of Airy's functions), but, if we try an Ansatz $U = e^{-cr^{3/2}}$, we get

$$U'' = \frac{9c^2}{4} r e^{-cr^{3/2}} - \frac{3c}{4} r^{-1/2} e^{-cr^{3/2}} \ .$$

The second term is subdominant, and we see that the solution will be of the form

$$U \propto e^{-cr^{3/2}} ,$$

plus terms which are subdominant for $r \to \infty$ provided that $\frac{9c^2}{4m_q} = \tau$. The leading behavior for large r is thus given by,

$$U(r) \propto_{r\to\infty} \exp\left(\mp\frac{2}{3}\sqrt{m_p\tau}\, r^{3/2}\right) ,$$

and, of course, only the upper, negative sign leads to an acceptable solution.

Finally, therefore, the equation we must solve to obtain the masses of the heavy–quark bound states takes the form,

$$-\frac{1}{m_q}\frac{d^2U}{dr^2} + \frac{l(l+1)}{m_q r^2}U + \left(2m_q - \frac{\alpha}{r} + \tau r\right)U = EU \tag{3.1.6}$$

subject to the boundary conditions $U(0) = U(\infty) = 0$. We will use this equation as an example to study the general case of computational methods for the solution of boundary value ordinary differential equations.

3.2. Methods of solution

We will consider equations of the general form,

$$y'' = f(y, y', x) , \tag{3.2.1}$$

where primes indicate differentiation with respext to the independent variable (x). In this equation, and for most of what follows, $y(x)$ can refer to several dependent variables, $y_i(x)$. There is no point studying equations with derivatives of order higher than second; first, equations of higher order are not all that common, but, more importantly, it is always possible to write higher–order equations as coupled systems of lower order. For example, an equation such as, $y''' = f(y, y'', x)$, can also be written as,

$$\begin{aligned} y_1'' &= y_2 \\ y_2'' &= f(y_1, y_2, x) . \end{aligned} \tag{3.2.2}$$

We will assume that x varies in the interval (a, b), with either limit possibly infinite. Boundary conditions will be expressed by constraints of the form,

$$\begin{aligned} g_1(y, y') &= 0 \quad \text{at} \quad x = a , \\ g_2(y, y') &= 0 \quad \text{at} \quad x = b . \end{aligned} \tag{3.2.3}$$

Typical examples

linear, non-homogeneous:

$$y''(x) = \rho(x), \qquad y(a) = y(b) = 0,$$

$$\text{or } y'(a) = y'(b) = 0;$$

linear, homogeneous:

$$-y''(x) + U(x)\,y(x) = Ey(x), \quad y(a) = y(b) = 0;$$

non-linear:

$$y''(x) - \sin y(x) = 0, \quad y(-\infty) = 0, \quad y(\infty) = \pi.$$

3.3. The shooting method

Supply additional information at e.g. $x = a$ sufficient to integrate the equation forward from a to b. Vary then the parameters until the condition at b is satisfied.

Classes of equations:

(i) general, non-linear

$$y'' = f(y, y'x).$$

The constraints are, for example, $y(a) = y_1$, $y(b) = y_2$. Introduce $y'(a) = c$ as parameter and integrate then the equation forward from a to b starting from $y(a) = y_1$ $y'(a) = c$. The value of y at b, $y(b)$ will depend on c: $y(b) \equiv y(b, c)$. Vary then c until $y(b, c) = y_2$.

With b fixed $y(b, c)$ is a function of c which we will denote by $y_b(c)$. We can write a subroutine that, given c, returns $y_b(c)$. Solving the equation becomes then equivalent to finding the zeroes of the function

$$y_b(c) - y_2.$$

In principle the procedure can be followed also if there are several dependent variables $y_1(x)$, $y_2(x)\ldots$ If, for instance, the boundary conditions are $y_1(a) = y_{1,1}$ $y_2(a) = y_{2,1}\ldots$ and $y_1(b) = y_{1,2}$ $y_2(b) = y_{2,2}\ldots$, we can assign to $y_1'(a)$, $y_2'(a)\ldots$ values c_1, $c_2\ldots$ and integrate the equations to find the values at b, denoted by $y_{i,b}$, which will depend on all the c_i

$$y_{i,b} \equiv y_{i,b}(c_1, c_2, \ldots).$$

We would then vary c_1 $c_2\ldots$ trying to find a simultaneous set of zeroes of all the functions

$$y_{1,b}(c_1 c_2 \ldots) - y_{1,2},$$

$$y_{2,b}(c_1 c_2 \ldots) - y_{2,2},$$

$$y_{3,b}(c_1 c_2 \ldots) - y_{3,2},$$

$$\ldots$$

The procedure becomes, however, much more complex, because the problem of finding the simultaneous solution of several non-linear equations is a rather difficult one.

(ii) Linear, non-homogeneous equations:

$$y'' = v_1(x)\, y + v_2(x)\, y' + v_3(x).$$

Again, we will consider, for definiteness, boundary conditions $y(a) = y_1$, $y(b) = y_2$.

Once more, we can assume a value c for $y'(a)$ and integrate the equation, adjusting c so that the other boundary condition is satisfied. (Of course, if the boundary condition at a were on y', e.g. $y'(a) = y_1$, we would take $y(a) = c$ as parameter. This remark applies also to the case considered in (i)).

While the method is the same as in (i), the linearity of the equation induces a simplification.

Indeed, imagine that starting from $y'(a) = c_1$, we obtain $y(b) = d_1$ and that, starting from $y'(a) = c_2$, we obtain $y(b) = d_2$.

If we now start from $y'(a) = c$ we will obtain

$$y(b) = d_1 + \frac{c - c_1}{c_2 - c_1}(d_2 - d_1)$$

because $y(b)$ is a *linear function* of $y'(a)$. Thus, we only need to integrate the equation forward for two different values of $y'(a)$ and we can then solve for the value of $y'(a) = c$ which satisfies the boundary condition $y(b) = y_2$.

This does generalize easily to the case of n variables $y_1(x) \ldots \quad y_n(x)$. We can integrate the equations with $n + 1$ independent sets of initial values and solve the resulting linear system of n equations in the n unknown coefficients $c_i \equiv y_i'(a)$.

Example: $n = 2$.

i) Integrate forward with $y_1'(a) = y_2'(a) = 0$ to obtain $y_1(b) = d_1^{(0)}$, $y_2(b) = d_2^{(0)}$.

ii) Integrate forward with $y_1'(a) = 1$, $y_2'(a) = 0$ to obtain $y_1(b) = d_1^{(1)}$, $y_2(b) = d_2^{(1)}$.

iii) Integrate forward with $y_1'(a) = 0$, $y_2'(a) = 1$ to obtain $y_1(b) = d_1^{(2)}$, $y_2(b) = d_2^{(2)}$.

The initial conditions $y_1'(a) = c_1$, $y_2'(a) = c_2$ will now produce

$$y_1(b) = d_1^{(0)} + \left(d_1^{(1)} - d_1^{(0)}\right) c_1 + \left(d_1^{(2)} - d_1^{(0)}\right) c_2,$$

$$y_2(b) = d_2^{(0)} + \left(d_2^{(1)} - d_2^{(0)}\right) c_1 + \left(d_2^{(2)} - d_2^{(0)}\right) c_2.$$

Equating the r.h.s. to the values that $y_1(b)$ and $y_2(b)$ must assume to satisfy the boundary conditions we obtain a system of 2 linear equations in 2 unknowns which we can solve for c_1 and c_2.

(iii) Linear, homogeneous equations:

$$y'' = v_1(x)\, y + v_2(x)\, y',$$

and boundary conditions that are also homogeneous (e.g. $y(a) = y(b) = 0$, or $y'(a) = y'(b) = 0$, or $c_1 y(a) + c_2 y'(a) = 0$ and $d_1 y(b) + d_2 y'(b) = 0$; but not boundary conditions such as $y(a) = c_1 \neq 0$. With this boundary condition the equation is no longer a homogeneous equation).

Generally $v_1(x)$ and/or $v_2(x)$ will contain some parameter that can be varied and non-trivial (i.e. non zero) solutions will exist only in correspondence of special values for this parameter.

Typical example is the eigenvalue equation

$$-y'' = Ey - v(x)\, y,$$

with $y(a) = y(b) = 0$.

With linear, homogeneous equation varying $y'(a)$ does not help, because the whole solution changes by the same factor. Rather, we start with a fixed $y'(a)$, e.g. $y'(a) = 1$, and integrate the equation for different values of the parameter(s) in $v_1(x)$ and $v_2(x)$ until we find a value such that the boundary condition at b is satisfied. Thus, in the eigenvalue equation

$$-y'' = Ey - v(x)\, y,$$

if we start from $y(a) = 0$, $y'(a) = 1$ and integrate, the value of y at b, y_b is a non-linear function of E: $y_b = y_b(E)$. The eigenvalues are then given by the zeroes of $y_b(E)$.

In all of the above equations one does not have, of course, to integrate forward from $x = a$. One can equally well integrate backward from $x = b$ down to $x = a$ or forward from $x = a$ to some $x = x_m$ and simultaneously backward from $x = b$ to the same value $x = x_m$. These alternate integration procedures may help one avoid instabilities in the integration of the equations (see later).

If we integrate the equation from $x = a$ to $x = x_m$ and from $x = b$ to $x = x_m$ in general we will have to specify one additional parameter, e.g. $y'(a) = c_1$, at a, and another parameter, e.g. $y'(b) = c_2$, at b. We shell then demand that the solutions $y^-(x)$ and $y^+(x)$, obtained integrating from $x = a$ and $x = b$ respectively, agree together with their derivatives at x_m:

$$y^-(x_m) = y^+(x_m),$$

$$y^{-\prime}(x_m) = y^{+\prime}(x_m).$$

This gives origin to two equations for the two parameters c_1 and c_2.

However, if the equation is linear and homogeneous, one can start with $y'(a) = 1$ and $y'(b) = 1$ and simply demand that

$$\frac{y^{-\prime}}{y^-} = \frac{y^{+\prime}}{y^+} \quad \text{at} \quad x = x_m\ ,$$

i.e., that the Wronskian $W = \det \begin{vmatrix} y^{-\prime} & y^{+\prime} \\ y^- & y^+ \end{vmatrix}$ vanishes at $x = x_m$.

This determinant will depend on the eigenvalue parameter (i.e. E) and thus one recovers one (non-linear) equation, whose zeroes produce the eigenvalues.

3.4. Instabilities in the solution procedure

With $b = \infty$ (or even at finite a or b) typically it will happen that the solution diverges rapidly for general values of either the initial parameter or the eigenvalue parameter. For example, the solution of

$$-y'' = Ey - v(x)\,y$$

for $v(x) > E$ is a linear combination of an exponentially decreasing solution and an exponentially increasing one, given, to leading order, by

$$y \sim \alpha_1 e^{-\int^x \sqrt{v(x')-E}dx'} + \alpha_2 e^{+\int^x \sqrt{v(x')-E}dx'}\,.$$

α_1 and α_2 will be functions of E and the eigenvalues are precisely those values of E that cause α_2 to vanish.

However, even if E is an exact eigenvalue, because of the numerical approximation in the integration procedure or because of round-off errors the divergent solution will eventually dominate. Moreover, if the integraton is carried out to very large values of x, the sensitivity of the solution to the changes in E will be so big that searching for a zero in the coefficient of the exponentially increasing term may be problematic.

All of this can also happen at finite x (so that changing the range by a change of independent variable or integrating down, for instance, from ∞ to 0 rather then from 0 to ∞, does not help). For example, the equation

$$\frac{d^2y}{dx^2} = \frac{l(l+1)}{x^2}y(x)$$

has a solution given by

$$y = \alpha_1 x^{l+1} + \alpha_2 x^{-l}$$

and the second term diverges for $x \to 0$.

In presence of such singular behaviours we can:

i) demand that $y = 0$ at a large but finite $x = x_{max}$. Then we can increase $x = x_{max}$ while tuning the eigenvalue parameter (however, the above mentioned sensitivity of $y(x)$ on E for very large x will, in general, still prevent us from considering exceedingly large $x = x_{max}$).

If we know the asymptotic behaviour of the correct solution, for instance that it must behave as e^{-cx}, we can take this to our advantage demanding that

$$y'(x_{max}) = -cy(x_{max})\,,$$

rather then $y(x_{max}) = 0$. This will produce good results with much lower values of x_{max}.

ii) Rescale the solution by the expected divergent asymptotic behaviour (this is particularly easy with linear homogeneous equations solved by iteration

$$y(x+dx) = f_1(x)\,y(x) + f_2(x)\,y(x-dx).$$

All we need to do is to replace the current values for $y(x)$ and $y(x-dx)$ with the rescaled ones).

iii) Integrate down from $x = x_{max}$ to some intermidiate x_m (in this integration the "correct" solution will now dominate) and similarly integrate up from $x = x_{min}$ to x_m and demand

$$\det \begin{vmatrix} y^{-\prime} & y^{+\prime} \\ y^{-} & y^{+} \end{vmatrix} = 0.$$

3.5. Alternative methods

Consider all the values $y(a)$ $y(a+dx)$ $y(a+2dx)$ ... $y(b)$ as unknown and solve the system of equations that follow from the discretization of the differential equation.

Solving is easy for a linear system, because of the special form of the matrix of coefficients. E.g., the equation $-y'' = \rho(x)$ with $y(a) = y(b) = 0$ becomes

$$\frac{2y(a+dx) - y(a+2dx) - y(a)}{(dx)^2} = \rho(a+dx)$$

$$\frac{2y(a+2dx) - y(a+3dx) - y(a+dx)}{(dx)^2} = \rho(a+2dx)$$

...

or, denoting $y(a+idx)$ by y_i $(i = 1 \ldots n-1)$,

$$2y_1 - y_2 = (dx)^2 \rho_1$$

$$2y_2 - y_3 - y_1 = (dx)^2 \rho_2$$

...

$$2y_{n-2} - y_{n-1} - y_{n-3} = (dx)^2 \rho_{n-2}$$

$$2y_{n-1} - y_{n-2} = (dx)^2 \rho_{n-1}$$

or, in matrix notation

$$\begin{vmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{vmatrix} \begin{vmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{vmatrix} = \begin{vmatrix} (dx)^2 \rho_1 \\ (dx)^2 \rho_2 \\ (dx)^2 \rho_3 \\ \vdots \\ (dx)^2 \rho_{n-2} \\ (dx)^2 \rho_{n-1} \end{vmatrix}$$

This system can be solved easily by assigning a value c for y_1, and going forward through the first $n-2$ equations to find $y_2 \ \ldots \ y_{n-2} \ y_{n-1}$. The last equation $-y_{n-2} + 2y_{n-1} = (dx)^2 \rho_{n-1}$ will not be satisfied for an arbitrary choice of c. However, the whole expression $-y_{n-2} + 2y_{n-1} - (dx)^2 \rho_{n-1}$ is a linear function of c and so we can easily determine the value of c that will make it equal to 0.

In practice, this coincides with the shooting method, and the corresponding alternatives in the solution procedure (such as solving forward from y_1 and backward from y_{n-1}, imposing consistency in the middle) that help avoid instabilities can also be used.

A direct solution of the system by a general purpose routine can also be done, although this is less economical.

A direct solution of the discretized set of equations in the non-linear case is much more problematic, unless the equations can be derived from a minimizaton principle.

For instance, the equation

$$-y''(x) + \lambda y^3(x) = 0\,, \qquad y(a) = y(b) = \text{ given}$$

follows from the requirement that

$$I = \int_a^b \{\frac{1}{2}[y'(x)]^2 + \frac{\lambda}{4}[y(x)]^4\} dx\,, \quad y(a) = y(b) = \text{ given}$$

by minimum. Similarly, the discretized version

$$-\frac{y(x+dx) + y(x-dx) - 2y(x)}{(dx)^2} + \lambda y^3(x) = 0$$

follows from the demand that

$$I' = \sum_x \{\frac{1}{2}\left[\frac{y(x+dx) - y(x)}{dx}\right]^2 + \frac{1}{4}\lambda y^4(x)\}$$

be minimum.

Then one can use powerful methods of minimization for I considered as a function of the variables $y(a+dx) \ \ldots \ \ y(b-dx)$.

Example: relaxation method for the solution of

$$-y''(x) + v(x)\,y(x) = Ey(x)\,.$$

We want the lowest eigenvalue(s) and corresponding eigenfunctions.

We use continuum notation for simplicity, but the method works also for a discretized system (indeed, the numerical procedure is only defined for a discretized system).

Let $E_1 < E_2 < E_3 \ldots$ be the eigenvalues and $y_1(x)$, $y_2(x)$, $y_3(x) \ldots$ the corresponding normalized solutions.

We can expand any $y(x)$ as

$$y(x) = a_1 y_1(x) + a_2 y_2(x) + \ldots$$

Consider now the equation

$$\frac{dy}{dt} = -\left(-y'' + v(x)\, y\right)$$

for a solution $y(x,t)$ of x and of a fictitious time variable t.

If we expand

$$y(x,t) = a_1(t)\, y_1(x) + a_2(t)\, y_2(x) + \ldots$$

and substitute into the equation we see that the coefficients $a_1(t)$, $a_2(t)$, ... satisfy

$$\frac{da_1(t)}{dt} = -E_1 a_1(t) \ ,$$

$$\frac{da_2(t)}{dt} = -E_2 a_2(t) \ ,$$

$$\ldots$$

Thus $a_i(t) = a_i(0)e^{-E_i t}$.

For large t, therefore, $y(x,t)$ will be dominated by the first term

$$y(x,t) \underset{t \to \infty}{\longrightarrow} a_1(t)\, y_1(x) \, .$$

In this way, by evolving the original equation in time (relaxation) we can obtain the eigenfunction $y_1(x)$ corresponding to the lowest eigenvalue and, from the rate of decay, also the eigenvalue E (NB: if $y(x,t)$ decreases too fast, we may rescale it so as to keep its normalization constant at each iteration, or every few time steps. If the rescaling factor, for a time interval Δt, is r, for large t $r \to e^{E_1 \Delta t}$, which can used to read off E_1).

Using the fact that the eigenfunctions $y_1(x)$ $y_2(x)$ ... are orthogonal (with respect to some suitable definition of inner product, frequently just

$$\langle y_1, y_2 \rangle = \int dx y_1(x)\, y_2(x) \sum_i y_1(x_i)\, y_2(x_i)$$

one can use the relaxation method to find a few of the lowest eigenvalues, not just the one corresponding to the lowest E.

Minimization principles can also be used to find a "good" discretization of a continuum equation, where a priori the discretization is ambigous or not obvious.

For instance, the equation

$$-\frac{d^2y}{dx^2} - \frac{1}{r}\frac{dy}{dr} + v(r)\,y = 0$$

follows from the minimization of

$$\int \{\frac{r}{2}\left(\frac{dy}{dr}\right)^2 + r\frac{v(r)}{2}y^2\}dr\,.$$

This suggest that we consider

$$I = \sum_i \{\left(\frac{r_i + r_{i+1}}{4}\right)\left(\frac{y_{i+1} - y_i}{dr}\right)^2 + \frac{r_i v(r_i)}{2}y_i^2\}dr$$

and minimize with respect to y_i.

This gives

$$\begin{aligned}&\left(\frac{r_i + r_{i+1}}{2}\right)\left(\frac{y_i - y_{i+1}}{dr^2}\right)\\ &+\left(\frac{r_i + r_{i-1}}{2}\right)\left(\frac{y_i - y_{i-1}}{dr^2}\right)\text{ (from the term with } y_i \text{ and } y_{i-1})\\ &+r_i v(r_i)\,y_i = 0\,.\end{aligned}$$

With equally spaced r_i this can also be rewriten as

$$\frac{2y_i - y_{i+1} - y_{i-1}}{dr^2} - \frac{1}{2r_i}\frac{y_{i+1} - y_{i-1}}{dr} + v(r_i)\,y_i = 0\,,$$

but, for some reasons, we might have wanted to r_i's which are not equally spaced and the discretization would then not have been so obvious. Similarly, if the equation had been of the form

$$-u(r)\frac{d^2y}{dr^2} - u'(r)\frac{dy}{dr} + v(r)\,y = 0\,.$$