Astronomy 8824: Numerical Methods Notes 2
Ordinary Differential Equations

Reading: Numerical Recipes, chapter on Integration of Ordinary Differential Equations (which is ch. 15, 16, or 17 depending on edition). Concentrate on the first three sections. Also look briefly at the first sections of the following chapter on Two Point Boundary Value Problems.

For orbit integrations, §3.4 of the current edition of Binney & Tremaine (Galactic Dynamics) is a good discussion, especially on the topics of symplectic integration schemes and individual particle timesteps.

We'll discuss some general aspects of integrating ODEs, while the example in the problem set will be a relatively simple case of integrating orbits in a gravitational potential, which is often relevant to astronomers!

A Side Note: Systems of Linear Equations

Reference: Numerical Recipes, chapter 2.

The system of equations

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ...a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ...a_{2n}x_n &= b_2 \\
&.... \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + ...a_{nn}x_n &= b_n
\end{aligned}$$

can be written in matrix form

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}.$$

If the equations are linearly independent, then the matrix **A** is non-singular and invertible.

The solution to the equations is:

$$\mathbf{x} = \mathbf{A^{-1}} \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{A^{-1}} \cdot \mathbf{b}.$$

The computational task of matrix inversion is discussed in Chapter 2 of NR.

For small, non-singular matrices it is straightforward, allowing a straightforward solution to a system of equations.

For example, in iPython try

```
import numpy as np
```

```
a=np.array([[1,9,7],[2,4,12],[3,7,3]])
```

```
b=np.array([9,-3,7])

ainv=np.linalg.inv(a)

np.dot(a,ainv)

x=np.dot(ainv,b)

np.dot(a,x)
```

The problem becomes more challenging when

• The solutions have to be done many, many times, in which case it's important to be efficient.

• The equations are degenerate or nearly so, in which case one needs *singular value decomposition.*

• The number of matrix elements is very large. If many of the matrix elements are zero, then one can use *sparse matrix methods.*

Initial Value Problems

Higher order differential equations can be reduced to systems of first-order differential equations.

For example

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x)$$

can be rewritten as

$$\frac{dy}{dx} = z(x)$$
$$\frac{dz}{dx} = r(x) - q(x)z(x),$$

two first-order equations that can be integrated simultaneously.

The generic problem to solve is therefore a system of coupled first-order ODEs,

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, y_2, ..., y_N), \qquad i = 1, ..., N\ .$$

For the case of an orbit in a gravitational potential we have

$$\frac{d^2\mathbf{x}}{dt^2} = -\vec{\nabla}\Phi(\mathbf{x})$$

which can be written

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}$$
$$\frac{d\mathbf{v}}{dt} = -\vec{\nabla}\Phi(\mathbf{x})\ .$$

This can be written in the $dy_i(x)/dx$ form above, though for orbits it is most straightforward to just think in terms of $\mathbf{x}$ and $\mathbf{v}$.

An initial value problem is one in which all values of y_i are specified at the initial value of x.

For example, one might specify $\mathbf{x}$ and $\mathbf{v}$ at $t = 0$ and integrate forward in time.

Runge-Kutta

We'll now think about integrating a single ODE for variable $y(x)$.

The simplest way to integrate an ODE, like the simplest way to do a numerical integral, is Euler's method:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

to advance from x_n to $x_{n+1} = x_n + h$.

As with numerical integrals, this is a bad method; one can get higher accuracy and greater stability with only slightly more work.

Mid-point method (2nd order Runge-Kutta)

Use derivative at x_n to advance to $x_{n+1/2}$. Then use the derivative at $x_{n+1/2}$ to advance from x_n to x_{n+1}.

$$\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + h/2, y_n + k_1/2) \\
y_{n+1} &= y_n + k_2
\end{aligned}$$

with an error $\mathcal{O}(h^3)$ per step.

Since the number of steps is $\propto 1/h$, the error in the integration should scale as h^2.

4th order Runge-Kutta

One can evaluate at more intermediate points and fit a higher order function.

This involves more evaluations per step, but it should allow one to take larger steps.

The most commonly used scheme is 4th order, which seems to be a good compromise between these two:

$$\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + h/2, y_n + k_1/2) \\
k_3 &= hf(x_n + h/2, y_n + k_2/2) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + k_1/6 + k_2/3 + k_3/3 + k_4/6 + \mathcal{O}(h^5) \ .
\end{aligned}$$

(See NR, eq. 17.1.3.)

Adaptive Step Size

If computing time is not an issue, you can just integrate your ODE multiple times with steadily decreasing h and check for convergence to your desired accuracy.

However, the step size required for an accurate integration may vary by by a large factor through the domain.

For example, integrating a highly elliptical orbit requires much smaller timesteps near peri-center than through most of the orbit, so using a fixed timestep may be wasteful.

You would also like your code to give you an estimate of the accuracy of its answer.

NR discusses strategies for doing this within Runge-Kutta.

The most straightforward idea is to advance from x to $x + H$, where H is macroscopic jump, using small steps h. Keep halving h (doubling the number of steps) until the desired accuracy is achieved.

Start with this value of h for the next H-step. If it's more accurate than needed, increase h; if it's less accurate than needed, decrease h.

There is a somewhat more efficient way to achieve this effect, described in NR and implemented in their routines.

This approach is quite general.

For a specific problem, you may have physical intuition about what should guide the size of steps.

For example, the accuracy of integration may depend on the ratio of the timestep to the local dynamical time $(G\rho)^{-1/2}$ for a gravitational calculation or sound-crossing time L/c_s for a hydrodynamic calculation.

In such situations, you may be able to choose a constant scaling factor that multiplies some function of local conditions, rather than using step-doubling or other Runge-Kutta strategies.

Bulirsch-Stoer

This is analogous to Romberg integration for numerical integrals.

Take a sequence of different numbers of steps, with decreasing h.

Using the results for this sequence, extrapolate to the result for $h = 0$.

NR is enthusiastic about this method for cases that require greater efficiency or higher accuracy than Runge-Kutta.

However, it's more susceptible to problems if the integration isn't smooth everywhere.

Leapfrog Integration

Leapfrog is a 2nd-order integration scheme for gravitational problems.

From initial conditions $\mathbf{x}_0$, $\mathbf{v}_0$, take a half-step to get $\mathbf{v}_{1/2}$.

Then "leapfrog" the positions over the velocities, then the velocities over the positions, and so forth.

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 + \mathbf{v}_{1/2}\Delta t \\
\mathbf{v}_{3/2} &= \mathbf{v}_{1/2} - \vec{\nabla}\Phi(\mathbf{x}_1)\Delta t \\
\mathbf{x}_2 &= \mathbf{x}_1 + \mathbf{v}_{3/2}\Delta t \\
&\text{etc.}
\end{aligned}$$

I am almost sure that this is equivalent to 2nd-order Runge-Kutta but for the specific case of equations where $\dot{x}$ depends only on v and $\dot{v}$ depends only on x, which makes it possible to do fewer evaluations.

Important: You will usually want to output both positions and velocities. If your arrays store positions on the whole steps and velocities on the half-steps, then you must advance the velocities half-a-step before output (but be careful to get them back on the half-step before continuing the integration). You also need to synchronize positions and velocities before computing energy conservation checks.

Stiff Equations and Implicit Integration

Sometimes you want to integrate a system where there are two or more very different timescales.

If you integrate with timesteps short compared to the shorter timescale, your calculation may never finish.

A common example is a hydrodynamic system with cooling. You are typically interested in phenomena happening on the sound-crossing timescale L/c_s, where L is the scale of the system.

But in a dense region, the cooling timescale may become orders-of-magnitude shorter than the sound-crossing timescale.

If you have to do the short timescale integration accurately to get the long timescale behavior to be accurate, then your stuck. But sometimes the short timescale behavior doesn't matter too much (e.g., it matters that the gas gets cold, but it doesn't matter exactly what its temperature is), and you just need your integration to be stable.

In this case, an implicit scheme for the short timescale behavior may be useful. Here is an example from NR:

Consider

$$y' = -cy \ , c > 0 \ .$$

The explicit Euler scheme is

$$y_{n+1} = y_n + hy'_n = (1 - ch)y_n \ .$$

If $h > 2/c$, then the method is unstable, with $|y_n| \to \infty$ as $n \to \infty$.

One can avoid this behavior by substituting y'_{n+1} for y'_n, obtaining the implicit equation

$$y_{n+1} = y_n + hy'_{n+1}.$$

In this particular case, the implicit equation is trivially solved

$$y_{n+1} = \frac{y_n}{1 + ch} \ ,$$

which gives a stable result even for large h.

More generally, the implicit equations may be solved by matrix inversion (if they're linear) or by some form of iteration (which we'll get to in two weeks) if they're non-linear.

If you have a stiff set of equations, then sometimes the physics of the problem will suggest an alternative solution.

For example, when timescales for ionization are short compared to other timescales in the system, then relative abundances may always be close to ionization equilibrium, where recombination rates balance ionization rates.

This gives a set of equations that can be solved as a function of local conditions (e.g., density, temperature), and perhaps stored in a lookup table.

If you're interested in evolution of a hierarchical triple star system where the outer orbital period is much longer than the inner orbital period, it may be adequate to treat the inner binary as a ring with mass spread over the orbit according to the amount of time the stars spend there.

Two Point Boundary Value Problems

Sometimes your boundary conditions aren't all specified at the same point.

For example, you might know an initial position but be trying to achieve a specific final velocity.

Stellar structure is a classic example: some boundary conditions are known at the center of the star, but others are known at the surface.

Two general strategies for this kind of problem are the *shooting method* and *relaxation methods.*

In the simplest version of the shooting method, you take a guess at the initial boundary values and see where you end up.

Then you change an initial boundary value and see how that changes where you end up.

You can now do some form of iteration (as discussed in two weeks) to try to zero in on the correct initial boundary condition to satisfy your final boundary condition.

It's like firing your artillery shell and continuously adjusting the angle of your gun until you hit your desired target.

Sometimes it may be very difficult to get to the right final boundary condition.

In such cases it may be better to integrate from both sides and try to meet in the middle, e.g., from the center of the star and the surface of the star.

In relaxation methods, you start with an approximate guess at the full solution.

You then use finite differences to calculate the errors in this guess at every point in the domain.

You then try to adjust your solution to eliminate these errors. If your guess is close, then you may be able to reduce the problem of eliminating the errors to a problem of solving simultaneous linear equations.

Relaxation methods are most useful when you have a very good guess to start with.

For example, you may want to solve for a sequence of main sequence stellar models with increasing mass. The solution at mass M, perhaps scaled in some way, then becomes a good starting guess for the solution at mass $M + \epsilon$.

One could do the evolution of a star of fixed M in a similar way, with the composition changing slightly from one time to the next.