PHY 325 Notes

Computational Physics

Jaeden Bardati

Last modified March 31, 2022

1 Numerical Evaluation of Derivatives

Let $y = f(x)$, then the **derivative** is defined as

$$\frac{\mathrm{d}f}{\mathrm{d}x} \equiv \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

Now consider $y = f(x_1, ..., x_n)$, then the **partial derivative** is defined as

$$\frac{\partial f}{\partial x_i} \equiv \lim_{h\to 0} \frac{f(x_1, ..., x_i + h, ..., x_n) - f(x_1, ..., x_n)}{h} \tag{2}$$

We can numerically compute derivatives in three main ways: forwards, backwards and with an average of the two.

The **forward** derivative is defined as

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x+h) - f(x)}{h} \tag{3}$$

The **backward** derivative is defined as

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x-h) + f(x)}{h} \tag{4}$$

The **central** derivative is defined as

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x+h) - f(x-h)}{2h} \tag{5}$$

The smaller n is, the more accurate this approximation is.

Note that, while the forward and backward derivatives take the same amount of time to compute, the central derivative has twice the number of calculations to do and so will take longer to run.

Furthermore, the central derivative can have divergent behaviour. To illustrate this, we can take the central derivative at the point (0, 0) on the absolute value function (see figure 1). Notice that the value of the derivative is 0 here, which could lead to undesired effects in the simulations of such systems.

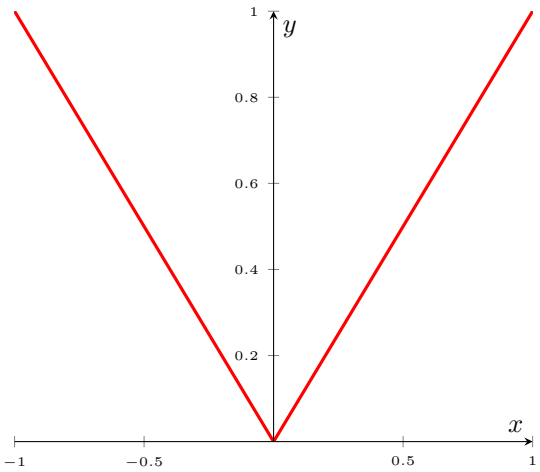

Figure 1: Absolute value function

2 Error in Derivative Calculation

When implementing a numerical derivative calculation, we would like to use a small value of h. But how small? One might think that we should try to make h as small as it can be. Of course, zero will not work since it would result in an error due to division by 0 (see equations 3, 4, and 5).

If we plot h by the error in the derivative, we will find that the derivative is inaccurate at both high and low values for h. There is a "sweet spot" in the middle (roughly around 10^{-8}) that minimizes the error. The error at low values of h is called **round-off error** and at high values of h is called **truncation error**.

2.1 Round-off Error

To understand this error, we must first ask: How are numbers represented by a computer? Specifically, we are interested in floats. The form is:

$$s \times M \times B^{e-E} \tag{6}$$

where s is the sign (0 if number is positive, 1 if negative), M is the Mantissa, B is the base, E is the bias, and e is the exponent.

This form is similar to scientific notation in principle. Namely, it is much more space-efficient to write 1.2×10^{-8} rather than 0.000000012. Note that here, 1.2 is the Mantissa, 10 is the base, the sign is 0, and $e - E = -8$.

For example, we will look at how 10.75 is stored. In binary, $(10)_{10} = (1010)_2$ and $(0.75)_{10} = (11)_2$, where the subscript indicates the base). So,

$$(10.75)_{10} = (1010.11)_2 = (1.01011)_2 \times 2^3$$

The bias term E is highly dependent on the particular machine you use. However, for a typical 32-bit computer, it is common to have a bias of $E = 2^{n-1} + 1$ with $n = 8$. This gives that $E = 129$. The base is, of course, $B = 2$.

Calculating e now, we know

$$e - E = 3 \implies e = 3 + E = 3 + 129 = 132$$

In binary, $(132)_{10} = (10000100)_2$.

Therefore, we would store the float value of 10.75 as

0	10000100	10101100000...000
s	$e - E$	M (23-bits)

In a 32-bit system, 1 bit is used to store the sign s, 8 bits for the exponent $e - E$, and 23 bits for the mantissa M. In a 64-bit system (double precision), 1 bit is used to store the sign s, 11 bits for the exponent $e - E$, and 52 bits for the mantissa M.

Since the mantissa is only a certain size, once a decimal value becomes lower than what can be stored in the mantissa, the value is lost. This is round-off error. The machine accuracy for a 32-bit system is around 10^{-8} and for a 64-bit system is 10^{-16}.

It is important when analyzing numerical systems to pick normalized units (close to unity). That is to say, choose units such that the numbers that are being used do not have this round-off error effect.

2.2 Truncation Error

Truncation error has to do with the sort of "rate of error decreasing with respect to h." Namely, if we Taylor expand $f(x+h)$, we obtain

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + ...$$

So, the calculation we use for the forward derivative (equation 3) is

$$\begin{aligned} & \frac{f(x+h)-f(x)}{h} = f'(x) + \frac{1}{2}hf''(x) + \frac{1}{6}h^2 f'''(x) + ... \\ \implies & \frac{f(x+h)-f(x)}{h} \approx f'(x) + O(h) \end{aligned}$$

We can see that the forward derivative is only accurate to the first order of h. This is why there is an increase in error when h is large. It is called **truncation error** since we truncate the infinite series when approximating.

3 Example: Projectile Motion

Consider a baseball thrown with air resistance. The equations of motion are

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \frac{1}{m}\vec{F}_{\text{air}}(v) - g\hat{y} \tag{7}$$

$$\frac{\mathrm{d}\vec{r}}{\mathrm{d}t} = \vec{v} \tag{8}$$

where $\vec{v}$ is the velocity, t is the time elapsed, m is the mass of the baseball, g is the gravitational acceleration, $\hat{y}$ is the upward direction, and $\vec{r}$ is the position vector. Note that the force of air friction here is

$$\vec{F}_{\text{air}}(v) = -\frac{1}{2}C_d\rho A|\vec{v}|\vec{v} \tag{9}$$

where C_d is the coefficient of air friction, ρ is the density of air, and A the area of the baseball (perpendicular to its direction of travel).

3.1 Euler Method

The Euler method is a numerical procedure for solve initial value problems of ordinary differential equations. Namely, if we have the form

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \vec{a}(\vec{r},\vec{v})$$

$$\frac{\mathrm{d}\vec{r}}{\mathrm{d}t} = \vec{v}$$

where $\vec{a}(\vec{r}, \vec{v})$ is the acceleration vector as a function of the position and velocity vectors.

Using the forward derivative where $\tau = h$ represents an increment in the time, then

$$\frac{\vec{v}(t+\tau) - \vec{v}(t)}{\tau} = \vec{a}(\vec{r}, \vec{v})$$
$$\frac{\vec{r}(t+\tau) - \vec{r}(t)}{\tau} = \vec{v}(t)$$

So,

$$\vec{v}(t+\tau) = \tau\vec{a}(\vec{r}, \vec{v}) + \vec{v}(t)$$
$$\vec{r}(t+\tau) = \tau\vec{v}(t) + \vec{r}(t)$$

We can write this as an iterative process

$$\vec{v}_{n+1} = \tau\vec{a}(\vec{r}_n, \vec{v}_n) + \vec{v}_n$$
$$\vec{r}_{n+1} = \tau\vec{v}_n + \vec{r}_n$$

This is what we use for the Euler method. Concretely, the steps for this method are

1. Specify the initial values $\vec{r}_1, \vec{v}_1$ at $t = 0$
2. Choose a time step τ
3. Calculate $\vec{a}$, given the current $\vec{r}$ and $\vec{v}$
4. Compute the new $\vec{v}_{i+1}$ and $\vec{r}_{i+1}$
5. Go to step 3

4 Example: Simple Pendulum

The equations of motion for the simple pendulum is

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L}\sin\theta \tag{10}$$

For small angles $\theta \ll 1$, we have

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L}\theta$$

The solutions of this approximation are

$$\theta(t) = C_1 \cos\left(\frac{2\pi t}{T_s} + C_2\right)$$

For arbitrary constants C_1 and C_2 determined by the initial conditions and where $T_s = \sqrt{\frac{L}{g}}$.

However, if we are interested in the behaviour of the pendulum in regions where the angle is not small, we must result to numerical approximation. If we wanted to use the Euler method here, we would split the second-order ODE into two first-order ODEs:

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = \alpha(\theta)$$
$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \omega$$

where $\alpha(\theta) = -\frac{g}{L} sin\Theta$. The Euler method would therefore be done using

$$\theta_{n+1} = \Theta_n + \tau\omega_n$$
$$\omega_{n+1} = \omega_n + \tau\alpha(\Theta_n)$$

However, because the Euler method is only accurate to the first-order, this implementation diverges to infinity in error very quickly. Instead, we must look to a new method.

4.1 Central Derivative Truncation Error

For the new scheme, we must understand the truncation error of the central derivative. We will start by Taylor expanding $f(t+\tau)$ and $f(t-\tau)$

$$f(t+\tau) = f(t) + \tau f'(t) + \frac{1}{2}\tau^2 f''(t) + \frac{1}{6}\tau^3 f'''(t) + ...$$
$$f(t-\tau) = f(t) - \tau f'(t) + \frac{1}{2}\tau^2 f''(t) - \frac{1}{6}\tau^3 f'''(t) + ...$$

So, using the formula for the central derivative (equation 5),

$$\frac{f(t+\tau) - f(t-\tau)}{2\tau} = f'(t) - \frac{1}{6}\tau^2 f'''(t) + ...$$
$$\implies \frac{f(t+\tau) - f(t-\tau)}{2\tau} \approx f'(t) + O(\tau^2)$$

The central derivative is therefore accurate to the second order of τ. Note that we can also find the second derivative by adding the Taylor expanded series instead of subtracting them.

4.2 Leap Frog Method

We will start with the general equations of motion where **the acceleration does not depend on velocity**. This is different from the Euler method, which allowed accelerations as a function of velocity.

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \vec{a}(\vec{r}(t))$$
$$\frac{\mathrm{d}\vec{r}}{\mathrm{d}t} = \vec{v}$$

where $\vec{a}(\vec{r}, \vec{v})$ is the acceleration vector as a function of the position and velocity vectors.

This time, we will evaluate using the central derivative instead of the forward one. We will evaluate the velocity at $t + \tau$ and $t - \tau$, and the position at $t + \tau$ and $t + 2\tau$,

$$\frac{\vec{v}(t+\tau) - \vec{v}(t-\tau)}{2\tau} + O(\tau^2) = \vec{a}(\vec{r}(t))$$
$$\frac{\vec{r}(t+2\tau) - \vec{r}(t)}{2\tau} + O(\tau^2) = \vec{v}(t+\tau)$$

We can rewrite this as

$$\frac{\vec{v}_{n+1} - \vec{v}_{n-1}}{2\tau} + O(\tau^2) = \vec{a}(\vec{r}_n)$$
$$\frac{\vec{r}_{n+2} - \vec{r}_n}{2\tau} + O(\tau^2) = \vec{v}_{n+1}$$

Therefore,

$$\vec{v}_{n+1} = \vec{v}_{n-1} + 2\tau\vec{a}(\vec{r}_n) + O(\tau^3)$$
$$\vec{r}_{n+1} = \vec{r}_n + 2\tau\vec{v}_{n+1} + O(\tau^3)$$

This is the **leap-frog method**. The solution is advanced in n steps of 2τ. Moreover, the position is evaluated at $\vec{r}_1$, $\vec{r}_3$, $\vec{r}_5$, etc., and the velocity is evaluated at $\vec{v}_2$, $\vec{v}_4$, $\vec{v}_6$, etc., hence the leap-frog.

4.3 Verlet Method

Now taking the 1st and 2nd derivatives, consider

$$\frac{\mathrm{d}\vec{r}}{\mathrm{d}t} = \vec{v}(t) \quad \frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} = \vec{a}(t)$$

Then,

$$\frac{\vec{r}_{n+1} - \vec{r}_{n-1}}{2\tau} + O(\tau^2) = \vec{v}_n$$
$$\frac{\vec{r}_{n+1} + \vec{r}_{n-1} - 2\vec{r}_n}{\tau^2} + O(\tau^2) = \vec{a}_n$$

Therefore,

$$\vec{v}_n = \frac{\vec{r}_{n+1} - \vec{r}_{n-1}}{2\tau} + O(\tau^2)$$
$$\vec{r}_{n+1} = 2\vec{r}_n - \vec{r}_{n-1} + \tau^2\vec{a}_n + O(\tau^4)$$

Therefore, if we do not need the velocity, we can have accuracy to the 4-th order.

Note that both the leap-frog and the verlet methods are not self-starting. In other words, you need to use another method to get the first step or two to work.

4.4 Euler-Cromer Method

We can make an improvement to the regular Euler method without doing too much. Namely, we first compute the velocity of the current iteration and then use the current velocity to find the current position. Namely, instead of

$$\vec{r}_{n+1} = \vec{r}_n + \tau\vec{v}_n$$
$$\vec{v}_{n+1} = \vec{v}_n + \tau\vec{a}(\vec{r}_n, \vec{v}_n)$$

which is the Euler method, we do

$$\vec{v}_{n+1} = \vec{v}_n + \tau\vec{a}(\vec{r}_n, \vec{v}_n)$$
$$\vec{r}_{n+1} = \vec{r}_n + \tau\vec{v}_{n+1}$$

This is the Euler-Cromer method.

5 Integration of Ordinary Differential Equations

Consider the general equation

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

ODEs can always be written as sets of first order differential equations. Here, we can do this by substitution,

$$\frac{\mathrm{d}y}{\mathrm{d}x} = z(x)$$
$$\frac{\mathrm{d}z}{\mathrm{d}x} = r(x) - q(x)z(x)$$

The general problem in ODEs is thus reduced to the study of N coupled first-order differential equations.

$$\frac{\mathrm{d}y_i}{\mathrm{d}x} = f_i(x, y_1, ..., y_N)$$

where $i = 1, ..., N$.

We always need boundary conditions. These can be in the form of

- Initial boundary conditions
  - All y_i are given at some starting rate of xs (start)
  - Need to find y_i at x_f (finish)
- Two point boundary value problem
  - Conditions are specified at more than one x
  - Typically at x_f, x_s

For example, when modeling the Sun, we set the boundary conditions of the temperature, luminosity, mass and pressure at the edge. Then, we integrate inwards to find those properties in the center.

5.1 General Strategy

The general stategy for attacking these types of problems is to rewrite the dy's and dx's as Δx's and Δy's, and then multiply by Δx. The literal interpretation is the Euler method. Though, in general one should not use the Euler method.

There are two very common general methods that one can try:

- Runge-Kutta
- Bulinsch-Stoer (extrapolation method)

5.2 Example: Kepler Orbit

Ou running example will be a Kepler problem of a small object in orbit around the Run (e.g. a comet). The gravitational force is

$$\vec{F} = \frac{-GmM}{|\vec{r}|}\vec{r}$$

where $\vec{r}$ is the position of the comet, m is the mass of the comet, M is the mass of the Sun, G is the gravitational constant.

The natural units for this problem are in AU, years and $M_\odot$. So,

$$GM = \frac{4\pi^2 \text{AU}^3}{\text{years}^2}$$

We want to trace something to make sure the behaviour is correct. We will track the total energy:

$$E = \frac{1}{2}mv^2 - \frac{GMm}{r}$$

We could now implement the Euler or Euler-Cromer methods if we want, however they will not be very accurate. Instead, we will look at a new, more accurate method.

5.3 Runge-Kutta Method

The formula for the Euler method is

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

which advances the solution from x_n to x_{n+1}.

- The Euler method is $\text{O}(n^2)$
- Only uses derivative information at the beginning of the interval
- Euler is not accurate and not stable

Consider instead the use of Euler to make a trial step to the mid-point and use the midpoint to advance the next step.

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned}$$

This is called **2nd order Runge-Kutta** or the **"midpoint" method**.

5.4 RK4 Method

The 4th order Runge-Kutta Method (RK4) is very commonly used. It goes as the following.

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

The idea is to take a few intermediate steps to determine the next y to return.

Now that we have a general method to solve this problems, we need to find a way of setting the size of h.

5.5 Adaptive RK4 Method

Let us use the example of an elliptical orbit. In the perihelion (closest to the center body), we would like h to be small so that it is more accurate, since the orbiting body moves fast there. However, we could allow h to be larger in aphelion (furthest to the center body), where the orbiting body moves slower.

The way we will do this is with adapting h on the fly. We will do this by comparing the relative error of a big step $y_b(t+h)$ to a couple small steps $y_s(t+h)$ (result from two steps of $t + \frac{1}{2}h$).

We will compare y_b with y_s to understand the *local truncation error*.If the error is tolerable, then the step is accepted, and a larger value of h is used for the next step.

How can we code this Adaptive RK4 method?

5.5.1 Code Outline

- Loop over maximum number of attempts to satisfy error bound (user set)
  - Take two small steps
  - Take one large step
  - Compute truncation error

  - Estimate h
  - If acceptable, return updated solution

- Calculate $\Delta_1 = y_s - y_b$
- Calculate $Delta_0 = \text{err} \times \frac{1}{2}(|y_s| + |y_b|)$
- Calculate $\Delta_{\text{ratio}} = \left|\frac{\Delta_1}{\Delta_0 + \text{eps}}\right|$
- Estimate new h value: $h_{\text{new}} = h(\Delta_{\text{ratio}})^{-\frac{1}{5}}$

We need to be careful with h_{new} since this is a linear interpolation. So, we will correct it

$$h_{\text{new}} = S_1 \times h_{\text{new}}$$

where $S_1 < 1$, and typically, $S_1 \approx 0.9$. Also,

$$h_{\text{new}} = \max(h_{\text{new}}, \frac{h}{S_2})$$
$$h_{\text{new}} = \max(h_{\text{new}}, S_2 h)$$

where $S_2 > 1$, and typically, $S_2 \approx 4$.

We then check if this actually worked. If $\Delta_{\text{ratio}} < 1$, then we are done. Otherwise, we use h_{new} as our h and try again.

5.6 Current Method

The currently accepted method is the **"embedded" solution**. The 5th order Runge Kutta (RK5) is

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + a_2 h, y_n + b_{21} k_1) \\ &... \\ k_6 &= hf(x_n + a_6 h, y_n + b_{61} k_1 + ... + b_{65} k_5)) \\ y_{n+1} &= y_n + c_1 k_1 + c_2 k_2 + ... + c_6 k_6 \end{aligned}$$

This is typically known as the **Runge-Kutta-Fehlberg methods**. The $a_2, a_3, ..., a_6$, $b_{21}, ..., b_{61}, ..., b_{65}$, and $c_1, ..., c_6$ are constants. There are tabular values of the coefficients, such as "cash-karp".

5.7 The Lorentz Model

The Lorentz model is

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \sigma(y-x)\\ \frac{\mathrm{d}y}{\mathrm{d}t} &= rx-y-xz\\ \frac{\mathrm{d}z}{\mathrm{d}t} &= xy-bz\end{aligned}$$

where r, σ, and b are constants.

This model demonstrates an example of chaos. That is to say, it is highly sensitive to a small change in the initial conditions if you iterate far enough in the future.

6 Solving Systems of Equations

Consider a general system of equations

$$\begin{gathered}a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1,\\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2,\\ \vdots\\ a_{M1}x_1 + a_{M2}x_2 + \ldots + a_{MN}x_N = b_N\end{gathered}$$

There are N unknowns: x_j for $j = 1, 2, ..., N$, related by M equations.

The coefficients a_{ij} with $i = 1, 2, ..., M$ and $j = 1, 2, ..., N$
The right had side quantities b_i for $i = 1, 2, ...M$.

If $M = N$,

- There is a good change of solving for x_j's.
- Unless one of the equaiton is linear, combination of the others
- Singular matrix

There are numerical considerations,

- Round off error may make the system singular
- Accumulated round-off error
  - important for large systems
  - the closer the matrix is to singular, the more problematic

For double precision, simple inversion with $N \leq 100$ is likely safe with any method.

There are many sophisticated packages that will detect and correct for singular issues (e.g. LSODE from ODEPACK). Some common software packages for this are

- LINPACK
  - Analyzes and solves linear least squares
  - Designed for supercomputers in the 70s and 80s
  - Largely superseded by LAPACK
- LAPACK
  - Solves systems of equations, eigenvalues problems and singular value problems
  - Also does LU decomposition, Cholesky, QR, SUD, etc.

Many compilers and software packages will include optimized versions of ODEPACK or LAPACK. These are all free and open-source.

When solving these equations, we should consider:

- Is the matrix only composed of positive numbers?
- Is the matrix equal to its own conjugate transpose?
- Is the matrix sparse (lots of zeros)?
- Is the matrix close to singular?

In these cases, there cna be significant increases in performance. Picking the right routine could make a algorithm that runs as $O(n^3)$ to $O(n \log n)$.

Our linear system can be written as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ & \dots\dots\dots\dots & & & \\ a_{M1} & a_{M2} & a_{M3} & \dots & a_{MN} \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ b_N \end{bmatrix}$$

This gives

$$A\boldsymbol{x} = \boldsymbol{b}$$

where we would like to solve for $\boldsymbol{x}$.

We have M rows and N columns.

If $M < N$, there is no solution.
If $M > N$, the system is over determined. This happens frequently. The typical case is wanting a solution to satisfying all equations (e.g. data fitting).

6.1 Gauss Elimination

Consider the system

$$\begin{aligned} x_1 + x_2 \ + x_3 &= 6 \\ -x_1 + 2x_2 \quad &= 3 \\ 2x_1 \quad + x_3 &= 5 \end{aligned}$$

Adding the first equation to the second and subtracting the first multiplied by 2 from the third gives

$$\begin{aligned} x_1 + x_2 \ + x_3 &= 6 \\ 3x_2 \quad + x_3 &= 9 \\ -\ 2x_2 + x_3 &= -7 \end{aligned}$$

Now, multiplying the second equation by $-\frac{2}{3}$ and subtracting the third gives

$$\begin{aligned} x_1 + x_2 \ + x_3 &= 6 \\ 3x_2 \quad + x_3 &= 9 \\ -\frac{1}{3}x_3 &= -1 \end{aligned}$$

This is known as forward elimination and then using backward substitution to solve for $x_1, x_2, ...$ This is a $O(n^3)$ routine.

Note that you should always use **pivoting**. To illustrate the need for this, consider the set of equations

$$\begin{aligned} \epsilon x_1 + x_2 + x_3 &= 5 \\ x_1 + x_2 \quad &= 3 \\ x_1 \quad + x_3 &= 4 \end{aligned}$$

In the limit as $\epsilon \to 0$, the solution is $x_1 = 1, x_2 = 2, x_3 = 3$.

Using the forward elimination,

$$\begin{aligned}
\epsilon x_1 + x_2 + x_3 &= 5 \\
(1 - \frac{1}{\epsilon})x_2 + \frac{1}{\epsilon}x_3 &= 3 - \frac{5}{\epsilon} \\
-\frac{1}{\epsilon}x_2 + (1 - \frac{1}{\epsilon})x_3 &= (4 - \frac{5}{\epsilon})
\end{aligned}$$

In the limit as $\epsilon \to 0$, we have a problem as the term $\frac{1}{\epsilon}$ blows up.

For example, $C - \frac{1}{\epsilon} \approx \frac{1}{\epsilon}$ for small ϵ. So our system of equations becomes

$$\begin{aligned}
\epsilon x_1 + x_2 \quad + x_3 &= 5 \\
-\frac{1}{\epsilon}x_2 - \frac{1}{\epsilon}x_3 &= \frac{5}{\epsilon} \\
-\frac{1}{\epsilon}x_2 - \frac{1}{\epsilon}x_3 &= \frac{5}{\epsilon}
\end{aligned}$$

The last two equations cause the singular condition to arise, even though the original matrix is not singular.

The solution is to interchange the order of the equations for forward elimination. This is called **pivoting**. For example, in the case before, simply changing the order of the equations gives

$$\begin{aligned}
x_1 + x_2 \quad &= 3 \\
\epsilon x_1 + x_2 + x_3 &= 5 \\
x_1 \quad + x_3 &= 4
\end{aligned}$$

Which yields

$$\begin{aligned}
x_1 + x_2 \quad &= 3 \\
+ (1 - \epsilon)x_2 + x_3 &= 5 - 3\epsilon \\
- x_2 \quad + x_3 &= 4 - 3\epsilon
\end{aligned}$$

Simply picking the largest element as the **pivot** is a good chance.

6.2 LU decomposition

Every matrix A can be decomposed into a lower and upper diagonal form.

$$A = L \cdot U$$

where L is the lower diagonal and U is the upper diagonal. For example,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ 0 & \beta_{22} & \beta_{23} & \beta_{24} \\ 0 & 0 & \beta_{33} & \beta_{34} \\ 0 & 0 & 0 & \beta_{44} \end{bmatrix}$$

If we can get this upper and lower form, then

$$A \cdot \boldsymbol{x} = (L \cdot U) \cdot \boldsymbol{x} = L \cdot (U \cdot \boldsymbol{x}) = \boldsymbol{b}$$

The we can solve the linear set

$$L \cdot \boldsymbol{y} = \boldsymbol{b}$$

solving for $\boldsymbol{y}$ through forward substitution. And then

$$U \cdot \boldsymbol{x} = \boldsymbol{y}$$

solving through back substitution.

This is useful for cases where the matrix is constant and only solving x for b is needed. Note that this can also be used to find the determinant of a matrix faster.

Actually getting the L and U matrices is the part that takes the most time. The decomposition is done by noting

If $i < j$,

$$a_{ij} = \alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + ... + \alpha_{ii}\beta_{ij} \tag{11}$$

If $i = j$,

$$a_{ii} = \alpha_{i1}\beta_{1i} + \alpha_{i2}\beta_{2i} + ... + \alpha_{ii}\beta_{ii} \tag{12}$$

If $i > j$,

$$a_{ij} = \alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + ... + \alpha_{ii}\beta_{ij} \tag{13}$$

To solve for $i = 1, 2, ..j$, use 11 and 12.

To solve for β_{ij}, use

$$\beta_{ij} = a_{ij} \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj}$$

And use 13 to solve

$$\alpha_{ij} = \frac{1}{\beta_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} \alpha_{ik}\beta_{kj} \right)$$

The determinant of the matrix is

$$\det(A) = \prod_{j=1}^{N} \beta_{jj}$$

6.3 Iterative Improvements

$$A \cdot \boldsymbol{a} = \boldsymbol{b}$$

We want to find $\boldsymbol{x}$ given $\boldsymbol{b}$ and A. We can find a slightly wrong solution $x + \delta x$.

If δb is the unknown error on $\boldsymbol{b}$,

$$A \cdot (x + \delta x) = b + \delta b \tag{14}$$

$$\implies A \cdot \delta x = \delta b$$

Using 14, to sub for δb. This gives

$$A \cdot \delta x = A \cdot (x + \delta x) \cdot b$$

Then solve for δx to get your new solution for x.

6.4 Single Value Decomposition

If A is a m by n matrix, U is m by n, ω is n by n, and V^T is n by n then

$$A = U \cdot \omega \cdot V^T$$

with $m > n$. Where

- U and V are orthogonal matrices
- ω is a diagonal

This is the ideal choice for an over-determined problems (such as least squares or curve fitting). The solution is

$$x = V \cdot \left[\text{diag} \left(\frac{1}{\omega_j} \right) \right] \cdot \left(U^T \cdot \boldsymbol{b} \right)$$

for $A \cdot \boldsymbol{x} = \boldsymbol{b}$.

7 Interpolation and Extrapolation

We may know the value of a function $f(x)$ at x_1, x_2, but we do not have an analytic function for $f(x)$. An example of this is data measurements or numerical calculations.

Given and ordered set of x_1, x_2, x_3, ..., x_n, we would like to know if $f(x)$ at arbitrary x.

If $x_1 \leq x$ and $x \leq x_n$, then we have **interpolation**.
If $x_1 > x$ or $x > x_n$, then we have **extrapolation**. This method requires caution.

We can use the functional forms:

- Polynomials
- Rational functions
- Trigonometric functions

The process of interpolation/extrapolation has two steps:

- Fit an interpolating function to the data
- Evaluate the function at arbitrary x

7.1 Polynomial Interpolation

For any two points there is a unique line. For any three points there is a unique quadratic, etc. In general, the solution of a polynomial can be written as

$$\begin{aligned} P(x) = & \frac{(x-x_2)(x-x_3)...(x-x_n)}{(x_1-x_2)(x_1-x_3)...(x_1-x_n)} y_1 \\ & + \frac{(x-x_1)(x-x_3)...(x-x_n)}{(x_1-x_1)(x_1-x_3)...(x_2-x_n)} y_2 + ... \\ & + \frac{(x-x_1)(x-x_2)...(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)...(x_n-x_{n-1})} y_n \end{aligned}$$

For example, this can be used to approximate the functions

$$f(x) \ = 2\sin(2x+3) + 2\sin(0.7x+1)$$

$$f(x) = |x|$$

$$f(x) = 3x^2 + \frac{1}{\pi^4}\log\left((\pi - x)^2\right) + 1$$

7.2 Cubic Spline Interpolation

Given $y_i = y(x_i)$, $i = 1, ..., N$.

For one interval between x_j and x_{j+1} we can linearly interpolate

$$y = Ay_i + By_j + 1$$

where

$$A = \frac{x_{j+1} - x}{x_{j+1} - x_j} \qquad\qquad B = 1 - A$$

The piece-wise description has zero second derivative inside each boundary and is undefined at intervals' boundaries.

If we know y_i at each x_i, we can add cubic polynomial such that y'' varies linearly from x_j to x_{j+1}. The provides a continuous second derivative. We can do this with

$$y = Ay_j + By_{j+1} + Cy''_j + Dy''_{j+1}$$

such that

$$C = \frac{1}{6}(A^3 - A)(x_{j+1} - x_j)^2 \qquad\qquad D = \frac{1}{6}(B^3 - B)(x_{j+1} - x_j)^2$$

Taking the derivative of y,

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{3A^2 - 1}{6}(x_{j+1} - x_j)y''_j + \frac{3B^2 - 1}{6}(x_{j+1} - x_j)y''_{j+1}$$

Taking another derivative gives

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} = Ay''_j + By''_j$$

In reality, we do not know y_j''. Instead, we can require that y' is continuous across the boundary.

This gives $N-2$ equations and N unknowns. This leaves us with two choices:

- Set y_1'' and y_2'' at the boundary. This is called the "Natural spline"
- Or define y_1'' and y_2'' yourself.

7.3 Bulirsch-Stoer Integration

This is a modified midpoint method for advancing a solution:

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = f(t, \vec{x})$$

the solution is advanced by

$$H = Nh$$

where N (equal sub-steps) is an even integer and H (full step) is constant, and h is small (small step). We try different values of N until we have some confidence of what the next value should be and then extrapolate to $n \to \infty$.

The power is that x(t+H) can be evaluated at different values of N that are divisibe by 2 (2, 4, 6, ...) and combined to get a final answer.

We estimate $\vec{x}(t+Nh)$ with N=2, 4, etc. and

$$\vec{x}_{t+H}(h) = a_0 + a_1 h + a_2 h^2 + ...a_{k-1} h^{k-1}$$

and extrapolate to $x_n \to \infty$.

We are fitting a polynomial of degree $k-1$ through points $h_1, h_2, ..., h_{k-1}$ with values $x_{t+H}(h_1), x_{t+H}(h_2)$, ...

The solution is exactly the polynomial interpolation we described earlier.

For some problems, Bulirch-Stoer is several times faster than RK4 for the same accuracy.

It is well suited for N-body/orbital integration because it can handle "close encounters."

8 Making your own N-body Integrator

The equations of motion are

$$\frac{\mathrm{d}^2\vec{x}}{\mathrm{d}t^2} = \sum_{j=1;i\neq j}^{N} \frac{Gm_j(\vec{x_i} - \vec{x_j})}{|\vec{x_i} - \vec{x_j}|^3}$$

for i = [1, N] bodies.

It is useful to consider the 2-body problem, as 3-body does not have an analytic solution.

$$\vec{F_1} = \frac{Gm_1m_2}{r^3}\vec{r} = m_1\ddot{\vec{r_1}}$$
$$\vec{F_2} = -\frac{Gm_1m_2}{r^3}\vec{r} = m_2\ddot{\vec{r_2}}$$

Note that

$$m_1\ddot{\vec{r_1}} + m_2\ddot{\vec{r_2}}$$

Integrating,

$$m_1\dot{\vec{r_1}} + m_2\dot{\vec{r_2}} = \vec{a}$$
$$m_1\vec{r_1} + m_2\vec{r_2} = \vec{a}t + \vec{b}$$

The center of mass is

$$\vec{R} = \frac{m_1\vec{r_1} + m_2\vec{r_2}}{m_1 + m_2}$$

Then,

$$\dot{\vec{R}} = \frac{\vec{a}}{m_1 + m_2}$$
$$\vec{R} = \frac{\vec{a}t + \vec{b}}{m_1 + m_2}$$

This means either $\vec{R}$ is stationary or moves in a straight line. Using $\ddot{(\vec{r})} = \ddot{(\vec{r})}_1 + \ddot{(\vec{r})}_2$, then

$$\frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} + \mu\frac{\vec{r}}{r^3}$$

Taking the vector product ($\times$) of $\vec{r}$ with the above equation,

$$\vec{r} \times \dot{\vec{r}} = \vec{h}$$

where $\vec{h}$ is some constant vector that is perpendicular to both $\vec{r}$ and $\dot{\vec{r}}$. It is related to the specific angular momentum.

Using polar coordinates,

$$\vec{r} = r\hat{r} \quad \dot{\vec{r}} = \dot{r}\hat{r} + r\dot{\theta}\hat{\theta} \quad \ddot{\vec{r}} = (\ddot{r} - r\theta^2)\hat{r} + (\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}t})\hat{\theta}$$

A change in the sweeping area A is

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \frac{1}{2}r^2\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{1}{2}h$$

Rewriting our equation of motion using polar coordinates,

$$\ddot{r} - r\dot{\theta}^2 = \frac{\mu}{r^2}$$

Letting $u = \frac{1}{r}$ and $h = r^2\dot{\theta}$, then

$$\ddot{u} + u = \frac{\mu}{h}$$

Thus, the ODE has the solution

$$u = \frac{\mu}{h}(1 + e\cos(\theta - \phi))$$

Switching back to r,

$$r = \frac{h^2}{\mu(1 + e\cos(\theta - \phi))}$$

Where e is the eccentricity, and ϕ is the phase.

8.1 Scale of the Problem

The gravitational cnonstants are $G = 6.28 \times 10^{-11}\mathrm{m}^3/\mathrm{kg}/\mathrm{s}^2$. We can set

$$G = 1\frac{[\mathrm{L}]^2}{[\mathrm{t}]^2[\mathrm{M}]}$$

We can use [t] = years, and [M] = $\mathrm{m}_\odot$. This means that [L] $5.0939\times10^1$1 m.

Kepler's Third Law states that

$$P^2 = \frac{4\pi^2}{GM}a^3$$

We can use this to find that 1 AU ≈ 0.2937 length units.

8.2 Choose your Integrator

The ones that are available are

- solve_ivp from scipy
- odeint from scipy
- custom

Ideally you want the third option.

8.3 Hamiltonian Construction

So far, we have looked at

$$\frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} + u\frac{\vec{r}}{r^3} = 0$$

where $u = Gm_1m_2$. We are using $\vec{r}$ and $\dot{\vec{r}}$ to describe the behaviour of the system. Instead we could use

$$\vec{r} = \vec{r}_x\hat{i} + \vec{r}_y\hat{j} + \vec{r}_z\hat{k}$$
$$\vec{p} = \vec{p}_x\hat{i} + \vec{p}_y\hat{j} + \vec{p}_z\hat{k}$$

where $\vec{p} = \frac{m_1m_2}{m_1+m_2}\vec{v}$.

We can then rewrite this as

$$\dot{\vec{r}} = \boldsymbol{\nabla}_p H \quad \dot{\vec{p}} = -\boldsymbol{\nabla}_r H$$

where

$$\boldsymbol{\nabla}_p = \hat{i}\frac{\partial}{\partial p_x} + \hat{j}\frac{\partial}{\partial p_y} + \hat{k}\frac{\partial}{\partial p_z}$$
$$\boldsymbol{\nabla}_r = \hat{i}\frac{\partial}{\partial r_x} + \hat{j}\frac{\partial}{\partial r_y} + \hat{k}\frac{\partial}{\partial r_z}$$

Here, the Hamiltonian is

$$H = \frac{p^2}{2\mu_r} - \frac{\mu\mu_r}{r}$$

with $\mu_r = \frac{m_1m_2}{m_1+m_2}$ and $\mu = Gm_1m_2$.

This gives us

$$\vec{r} = \frac{\vec{p}}{\mu_r} \quad \dot{\vec{p}} = \frac{-\mu\mu_r}{r^3}\vec{r}$$

For N-body, H is constant and is equal to the total energy of the system. The total energy of the system is

$$E = \frac{1}{2}m_1|\vec{v}_1|^2 + \frac{1}{2}m_2|\vec{v}_2|^2 - \frac{Gm_1m_2}{|\vec{r}_1 - \vec{r}_2|}$$

Using the reduced mass $\mu_r = \frac{m_1m_2}{m_1+m_2}$,

$$E = \frac{1}{2}\mu_r v^2 - \frac{GM\mu_r}{r} \quad p = mv$$

So,

$$E = \frac{p^2}{2\mu_r} - \frac{\mu\mu_r}{r}$$

This is our Hamiltonian. We can then use a symplectic integrator, where the energy must be conserved. In most cases that we can write the Hamiltonian explicitly like this, we will want to use it so keep energy conserved.

9 Smoothed Particle Hydrodynamics

There are two main types of SPH code

- Eulerian: Grid based approach
- Lagrangian: Particle based approach

The major difference between nbody and SPH is that gas has pressure. First, we will write down the system of equations

$$\frac{\mathrm{d}\dot{\vec{r}}}{\mathrm{d}t} = \vec{v}_i$$
$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = -\frac{1}{\rho_i}\vec{\boldsymbol{\nabla}}P - \boldsymbol{\nabla}\Phi$$

where ρ_i is the density, P is the pressure, and Φ is the gravitational potential.

We want to think of each particle as a smeared out distribution.

$$\rho_i(\vec{r}) = m_j W(|\vec{r} - \vec{r}_j|, h)$$

Where W is the smoothing kernel, $|\vec{r} - \vec{r}_j|$ is the distance between to the particle i, and h is the smoothing length.

The equation for a Gaussian kernel is

$$W(|\vec{r}-\vec{r}_j|,h) = \frac{1}{h^3\pi^{\frac{3}{2}}}\exp\left[-\left(\frac{|\vec{r}-\vec{r}_j|}{h}\right)^2\right]$$

To get the physical density at any point we use

$$\rho(\vec{r}) = \sum_{j=1}^{n}\rho_j(\vec{r})$$

The kernel must be defined such that

$$\int_0^\infty W(|\vec{r}-\vec{r}_j|,h)\ \mathrm{d}\vec{r} = 1$$

So that mass is conserved.

Any physical quantity (e.g. pressure, temperature, etc.) can be estimated with

$$A(r) = \sum_{j=1}^{N} m_j \frac{A_j}{\rho_j} W(|\vec{r}-\vec{r}_j|,h)$$

Derivatives come from the derivative of the Kernel itself.

$$\boldsymbol{\nabla}\vec{P}(\vec{r}) = \sum_{j=1}^{N} m_j \frac{P_j}{\rho_j}\boldsymbol{\nabla}W(|\vec{r}-\vec{r}_j|,h)$$

The ingredients for SPH are

- Initial conditions (velocities and positions)
- Potential (e.g. gravity)
- A physical law relating ρ_i and P_i
- A time integration scheme
- Method to determine $\frac{\boldsymbol{\nabla}P_i}{\rho_i}$
- Determining the smoothing length (h)

Your choice of δt should take into account the sound speed c_s.

$$c_s^2 = \frac{\mathrm{d}P}{\mathrm{d}\rho}$$

$$\implies \delta t < \frac{h}{c_s}$$

This is known as the Courant-Friedrichs-Levy (CFL) condition.

Let's consider a system of particles. The adiabatic gas law is:

$$P = k\rho^{\gamma}$$

We can set the total mass to something like the mass of jupiter with 400 particles.

We can also introduce viscosity into the equations:

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = -\frac{1}{\rho}\boldsymbol{\nabla}\vec{P} - \boldsymbol{\nabla}\Phi - \nu\vec{v}$$

where ν is the viscosity and $\vec{v}$ is the speed.

For "Jupiter", $k = 2.6{\times}10^5\ \mathrm{Nm}^4/\mathrm{kg}^2$ ($P \approx 6.5{\times}10^1 2N/m^2$ and $\rho \approx 5000kg/m^3$).

A dimensional analysis of k yields that

$$k = \frac{[\text{length}]^5}{[\text{time}]^2[\text{mass}]}$$

Let the mass be in Jupiter masses, the length unit be in jupiter radii, $k = 1$, this defines the time unit. It turns out that G is also close to 1.

9.1 Choosing a Kernel

We do not need to choose the Gaussian kernel, we can also consider a kernel with compact-support called a "spline kernel". For example,

$$W(r,h) = \frac{1}{\pi h^3}[1 - \frac{3}{2}(\frac{r}{h})^2 + \frac{3}{4}(\frac{r}{h})^3] \quad \text{for } 0 < \frac{r}{h} \leq 1$$

$$W(r,h) = \frac{1}{4\pi h^3}[2 - \frac{r}{h}]^3] \quad \text{for } 1 \leq \frac{r}{h} \leq 2$$

$$W(r,h) = 0 \quad \text{for } \frac{r}{h} \geq 2$$

9.2 Variable Smoothing Length

If the density range of your problem is large, than a uniform smoothing length will not work well. To make this variable we could use

$$\frac{\mathrm{d}h_i}{\mathrm{d}t} = -\frac{h_i}{3\rho_i}\frac{\mathrm{d}\rho_i}{\mathrm{d}t}s$$

To use a variable smoothing length, you need to symmetrize the smoothing interaction to insure that the force seen by particle i on j is the same as seen by j on i.

$$A_i = \frac{1}{2}\sum_j m_j \frac{A_j}{\rho_j}[W(r_{ij}, h_j) + W(r_{ij}, h_i)]$$

for any property of i A_i. So,

$$\rho_i = \frac{1}{2}\sum_j m_j [W(r_{ij}, h_j) + W(r_{ij}, h_i)]$$

This is taking a mean, but we can also use other methods.

9.3 Adding energy to SPH

If the gas is adiabatic, then it is wasy to track internal energy:

$$P = (\gamma - 1)\rho\epsilon$$

where ϵ is the energy per unit mass for an ideal gas. The ideal gas law is

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}t} = -(\frac{P}{R})\vec{\boldsymbol{\nabla}} \cdot \vec{v}$$

So with the kernel,

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}t} = \sum_{j=1}^{N} m_j (\frac{\sqrt{P_i P_j}}{\rho_i \rho_j})\vec{v}_{ij} \cdot \frac{1}{2}[\boldsymbol{\nabla}_i W(r_{ij}, h_i) + \boldsymbol{\nabla} W(r_{ij}, h_j)]$$

with $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$
and the geometric mean is

$$\boldsymbol{\nabla} P = 2\sqrt{P}\boldsymbol{\nabla}\sqrt{2}$$

If we do not use the variable smoothing length, we get

$$\frac{\vec{\boldsymbol{\nabla}} P_j}{\rho_i} = \sum_{j=1}^{N} m_j (\frac{\sqrt{P_i P_j}}{\rho_i \rho_j})[\boldsymbol{\nabla}_i W(r_{ij}, h_i) + \boldsymbol{\nabla} W(r_{ij}, h_j)]$$

9.4 Adding Heat Transfer

Consider conduction as a diffusive process.

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}t} = \frac{1}{\rho}\vec{\boldsymbol{\nabla}} \cdot (k\vec{\boldsymbol{\nabla}} T)$$

where k is the heat conduction coefficient.

For an ideal gas, $\epsilon = C_v T$ where C_v is the heat capacity and T is the temperature.

We need the second derivative

$$\frac{\mathrm{d}\epsilon_i}{\mathrm{d}t} = -\sum_{j=1}^{N} m_j \frac{(k_j + k_i)(T_i - T_j)(\vec{r}_{ij}\vec{\boldsymbol{\nabla}}_i W_{ij})}{\rho_i \rho_j |\vec{r}_{ij}|^2}$$

9.5 Navier-Stokes Equations

Gases and liquids are ensembles of atoms and molecules. Ideally we would want to know the position $\vec{x_i}$ and velocity $\vec{u_i}$ and force per unit mass $\vec{F_i}$ acting on the particle i.

$$\frac{\mathrm{d}\vec{x_i}}{\mathrm{d}t} = \vec{u_i} \quad \frac{\mathrm{d}\vec{u_i}}{\mathrm{d}t} = \vec{F_i}(x_j, u_j, t) \forall j$$

One mole of gas has $10^2 3$ particles.

To describe gases and liquids, we will use a statistical approach with continuous media which will be described by equation of hydrodynamics.

Let $\vec{x} = (x_1, x_2, x_3)$ be the position, $\vec{u} = (u_1, u_2, u_3)$ bet the velocity, $\vec{q} = (q_1, q_2, q_3)$ bet the momentum, and $\vec{F} = (F_1, F_2, F_3)$ is the force per unit mass.

We will introduce the distribution function.

$$dN = f(\vec{x}, \vec{u}, t) d\vec{x} d\vec{u}$$

where dN is the number of particles between $\vec{x}$ and $\vec{x} + d\vec{x}$ in position, and between $\vec{u}$ and $\vec{u} + d\vec{u}$ in velocity.

If we ignore collisions (for now), then if t changes by dt, then

$$\vec{q} \to \vec{q} + m\vec{F}dt$$
$$\vec{u} \to \vec{u} + \vec{F}dt$$
$$\vec{x} \to \vec{x} + \vec{u}dt$$

and

$$f(\vec{x} + \vec{u}dt, \vec{u} + \vec{F}dt, t + dt) - f(\vec{x}, \vec{u}, t) = 0$$

with collisions

$$f(\vec{x} + \vec{u}dt, \vec{u} + \vec{F}dt, t + dt) - f(\vec{x}, \vec{u}, t) = [\Delta f]_{\text{coll}}$$

Or

$$\frac{\partial f}{\partial t} + u_i \frac{\partial f}{\partial x_i} + F_i \frac{\partial f}{\partial u_I} = [\frac{\partial f}{\partial t}]_{\text{coll}}$$

where the i implies a summation over all indices. This is the Boltzmann equation. It describes the net number of particles that leave $d\vec{x}$, d$\vec{u}$ phase-space.

At a locatoin $\vec{x}$, $f(\vec{x}, \vec{u}, t)d\vec{u}$ will be the number of particles per unit volume in the interval $\vec{u}$ to $\vec{u} + d\vec{u}$.

$$\implies n(\vec{x}, t) = \int f(\vec{x}, \vec{u}, t)d\vec{u}$$

is the total number of particles per unit volume.

The mass density (or just "density") is

$$\rho(\vec{x,t}) = \int m f(\vec{x}, \vec{u}, t)d\vec{u}$$

The bulk velocity will be

$$\vec{v}(\vec{x}, t) \equiv \frac{1}{\rho} \int \vec{u} m f(\vec{x}, \vec{u}, t)d\vec{u}$$

If collisions are elastic and singular, then you get the Maxwell-Boltzmann Distribution:

$$f(\vec{x}, \vec{u}, t)d\vec{u} = n(\vec{x}, t) \left[(\frac{m}{2\pi k_B T(\vec{x}, t)})^{\frac{3}{2}} \exp\left(\frac{-m(\vec{u} - \vec{v})^2}{2k_B T(\vec{x}, t)} \right) du \right]$$

where k_B is the Boltzmann constant and T is the temperature. This is better suited for equilibrium and not shocks.

The fd$\vec{u}$ from the Maxwell-Boltzmann equation gives the number of particles per unit volume between $\vec{u}$ and $\vec{u} + d\vec{u}$ in an equilibrium state.

The specific internal energy ϵ (per unit mass) will be

$$\epsilon = \frac{3}{2} \frac{k_b}{m} T$$

where temperature is a measure of the kinetic energy associated with peculiar motions (motions relative to the bulk flow).

Note: Phase space can be defined using momentum $\vec{q}$ instead of velocity ($\vec{u}$). This is useful when dealing with massless particles.

The equations of hydrodynamics come from the moments of the Boltzmann Equation. For kth order moment ($k = 0, 1, 2$), we multiply by $U_k = (1, \vec{u}, u^2)$

and integrate over the whole velocity space. These are the mean, variance, etc.

If collisions converse the number of particles, mass momentum, and energy, then

$$\int [\frac{\partial f}{\partial t}]_{\text{coll}} d\vec{u} = 0$$

since the number of particles is conserved

$$\int [\frac{\partial f}{\partial t}]_{\text{coll}} u_i d\vec{u} = 0$$

(for $i = 1, 2, 3$) since the momentum is conserved
Multiplying the Boltzmann Equation by m and integrating,

$$m \int \frac{\partial f}{\partial t} d\vec{u} + \int u_i \frac{\partial f}{\partial x} d\vec{u} + mF_i \int \frac{\partial f}{\partial u_i} d\vec{u} = 0$$

To converve particles the $\int \frac{\partial f}{\partial u_i} d\vec{u}$ term goes to zero, and the collision cancel out (total is zero) since the number of particles is conserved.

This gives

$$\frac{\partial}{\partial t} \int m f f \vec{u} + \frac{\partial}{\partial x_i} \int u_i m f d\vec{u} = 0$$

or

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i} = 0$$

or

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \vec{v}) = 0$$

This is known as the **continuity equation**. It describes how a fluid conserves mass in its motion.

If we multiply the Boltzmann Equation by mu_i $(i = 1, 2, 3)$ and integrate, we find

$$\frac{\partial}{\partial t} + \frac{\partial}{\partial x_j} \int m u_i v_j f d\vec{u} - \rho F_i = 0$$

for each i and summing over $j = 1, 2, 3$.

The term

$$\int m u_i v_j f d\vec{u} = \int m v_i v_j f d\vec{u} + \int m \tilde{u_i} \tilde{u_j} f d\vec{u}$$

The first integral is the bulk terms and the second is the peculiar terms. The pressure is defined as

$$P_{ij} \equiv \int m\tilde{u_i}\tilde{u_j}$$

If the pressure is isotropic ($P_{ij} = P_{ji}$), then

$$P = \frac{1}{3}\int m{\tilde{u_i}}^2 f d\vec{u}$$

$$\implies \frac{\partial}{\partial t} + \frac{\partial}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \rho_i F_i$$

This is the **momentum equation**. Here, ρv_i is the momentum density and $\rho v_i v_j$ is the momentum flux. This can also be written as

$$\frac{\partial}{\partial t} + \boldsymbol{\nabla} \cdot \vec{\Pi} = \rho \vec{F}$$

where

$$\Pi_{ij} = \rho v_i v_j + P\delta_{ij}$$

is the momentum flux density tensor.

The internal energy can be derived from the momentum and continuity equations.

$$\frac{\partial e}{\partial t} + \frac{\partial}{\partial x_j} = -P\frac{\mathrm{d}v_{ij}}{\mathrm{d}x_i}$$

where e is the internal energy per unit volume.

The momentum and continuity equations along with the above equation for the internal energy are known as the Euler Equations.

We have $\rho, v_1, v_2, v_3, P, \epsilon$ plus an external force F_i (for example, gravity).

These are 5 equations with 6 unknowns so we are missing an equation. This equation is the equation of state.

To solve we need a relationship between P and ρ. For example,

$$P = \frac{2}{3}\rho\epsilon$$

In general the equation of state can be very complicated.

Euler's equations describe a collection of particles as a continuous motion. This approximation is valid if the free mean path between collision is less than the characteristic length scale of the problem

$$\lambda \ll l_c h$$

where λ is the free mean path and $l_c h$ is the characteristic length.

The number of particles here should be very large.

This means the distribution function should not vary over a length scale of $l_c h$.

Inter-particle forces should be short range. Therefore, there must be no interaction between particles separated by distances greater than $l_c h$. E.g. Gravity is from a source external to the fall medium.

Euler equations describe the state of the medium at a fixed location in space "x". This is called **"Eulerian"**.

$\frac{\partial}{\partial t}$ describes changes due to a "flow" of the medium past $\vec{x}$.

In the **Lagrangian** method, comoving spatial coordinates (r_1, r_2, r_3).

$\frac{\mathrm{d}}{\mathrm{d}t}$ describes changes occurring within the element as it changes its state and location.

Eulerian equations become

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \rho \frac{\mathrm{d}v_i}{\mathrm{d}x_j} = 0$$

$$\frac{\mathrm{d}v_j}{\mathrm{d}t} = -\frac{1}{\rho}\frac{\mathrm{d}P}{\mathrm{d}x_i} + F_i$$

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}t} = -\frac{P}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}t} = 0$$

The Euler Equations neglect the interchange of particles between adjacent fluid elements. When these effects are significant, we have to consider internal friction or **viscosity** of the fluid.

The momentum equation describes macroscopic transport of momentum through space and time due to external forces. In a viscous fluid, we need to consider the microscopic transport of momentum due to friction.

In a viscous fluid, we introduce a new momentum flux density.

$$\Pi_{ij} = \rho v_i v_j + P\delta_{ij} - \sigma_{ij}$$

where we introduce σ_{ij} as the viscous stress.

$$\sigma_{ij} = \nu(\frac{\mathrm{d}v_i}{\mathrm{d}x_j} + \frac{\mathrm{d}v_j}{\mathrm{d}x_i} - \frac{2}{3}\frac{\mathrm{d}v_k}{\mathrm{d}x_k}\delta_{ij}) + \eta\frac{\partial v_k}{\partial x_k}\delta_{ij}$$

ν and η are the shear and bulk viscosity coefficients.

The momentum equation becomes

$$\frac{\partial}{\partial t}(\rho v_i) + \frac{\partial}{\partial x_j}(\rho v_i v_j) = -\frac{\partial P}{\partial x_j} + \frac{\partial \sigma_{ij}}{\partial x_j} + \rho F_i \tag{15}$$

And,

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j} = -P\frac{\partial v_j}{\partial x_j} + \sigma_{ij}\frac{\partial v_j}{\partial x_k} \tag{16}$$

And the continuity equation does not change.

$$\frac{\partial P}{\partial t} + \frac{\partial}{\partial x_j}(\rho v_j) = 0 \tag{17}$$

These three equations (15, 16, and 17) are known as the Navier-Stokes equations.

9.6 Diffusion Equation

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(v_d\frac{\partial u}{\partial x}\right)$$

where u is some physical quantity and $v_d > 0$ is the diffusion coefficient.

If we consider v_d to be constant, then

$$\frac{\partial u}{\partial t} = v_d\frac{\partial^2 u}{\partial x^2}$$

We will start at $t = 0$, over the domain $0 \leq x \leq 1$ with $u(x, 0) = \delta_D(x - 0.5)$, where x is the position and δ_D is the Dirac delta function. The boundary conditions are $u(0, t) = u(1, t) = 0 \ \forall \ t$.

The analytical solution is

$$w(x,t) = 2\sum_{n=1}^{\infty}\sin\left(\frac{\pi n}{2}\right)\sin(\pi n x)\exp\left(-\pi^2 n^2 v_d t\right)$$

To solve this non-analytically, we will construct a grid.

The space derivative is

$$u_{j+1}^n = u_j^n + \left.\frac{\partial u}{\partial x_j}\right|_j \delta x + \frac{1}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_j (\delta x)^2 + O(\delta x)^3$$

The forward derivative here is

$$\left.\frac{\partial u}{\partial x}\right|_j = \frac{u_{j+1}^n - u_j^n}{\Delta x}$$

The backwards derivative here is

$$\left.\frac{\partial u}{\partial x}\right|_j = \frac{u_j^n - u_{j-1}^n}{\Delta x}$$

The centered derivative here is

$$\left.\frac{\partial u}{\partial x}\right|_j = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}$$

The second derivative can also be found similarly in the same grid space.

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\delta x)^2}$$

The equation we will use then is

$$u_j^{n+1} = u_j^n + v_d \frac{\Delta t}{\Delta x^2}(u_{j+1}^n + u_{j-1}^n - 2u_j^n)$$

9.7 Modeling a sound wave

Consider the equations of continuity and momentum for a 1D flow.

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0$$

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = -\frac{1}{\rho}\frac{\partial P}{\partial x}$$

Let's assume the fluid is **isothermal** in space and time. Therefore, the sound speed is

$$c_s^2 \equiv \frac{P}{\rho} \equiv \text{constant}$$

The initial conditions we will use are

$$\rho(x,0) = \rho_0 \qquad P(x,0) = P_0 \qquad v(x,0) = 0$$

The fluid is perturbed such that

$$\rho = \rho_0 + \rho_1 \qquad P = P_0 + P_1 \qquad v = v_1$$

where $\rho_!$ and P_1 are small compared to ρ_0 and P_0.

The equations then simplify to be

$$\frac{\partial \rho_1}{\partial t} - \rho_0 \frac{\partial v_1}{\partial x}$$

$$\frac{\partial v_1}{\partial t} = -\frac{1}{\rho_0}\frac{\partial \rho}{\partial x}$$

If we let $\phi = P_1$ and $\psi = v_1 c_s \rho_0$. This allows us to rewrite this equations with

$$\frac{\partial \psi}{\partial t} = -c_s \frac{\partial \phi}{\partial x} \tag{18}$$

$$\frac{\partial \phi}{\partial t} = -c_s \frac{\partial \psi}{\partial x} \tag{19}$$

Take now the partial derivative with respect to x of equation 18 and the partial derivative with respect to t of 19. Then, we obtain the wave equation

$$\frac{\partial^2 \phi}{\partial t^2} = c_s^2 \frac{\partial^2 \phi}{\partial x^2}$$

When implementing, we want to look at the quantity

$$S = \frac{c_s \Delta t}{\Delta x}$$

We would like $0 < S < 1$ and generally $S \approx 0.5$. And we would like Δt sufficiently small.

9.7.1 Staggered Grid

We will use a staggered grid with ϕ defined at $j + \frac{1}{2}$ and ψ at j.

This method is good for conservation of energy (similar to the leap frog method for ODEs).

We can now write our equations for as finite difference equations:

$$\frac{1}{\Delta t}(\psi_j^{n+\frac{1}{2}} - \psi_j^{n-\frac{1}{2}}) = -\frac{c_s}{\Delta x}(\phi_{j+\frac{1}{2}}^n - \phi_{j-\frac{1}{2}}^n)$$
$$\frac{1}{\Delta t}(\phi_{j+\frac{1}{2}}^{n+1} - \phi_{j+\frac{1}{2}}^n) = -\frac{c_s}{\Delta x}(\psi_{j+1}^{n+\frac{1}{2}} - \psi_j^{n+\frac{1}{2}})$$

We start with ϕ at t^n and ψ at $t^{n+\frac{1}{2}}$, then we get ϕ at t^{n+1} and proceed to get ψ at $t^{n+\frac{3}{2}}$.

For the wave propagation to be stable,

$$\Delta t \leq \frac{\Delta x}{v} \quad \text{or} \quad \Delta t \leq \frac{\Delta x}{v}$$

where v is the propagation speed of the wave. Here it is the speed of sound c_s. This is known as the CFL condition.

We can define

$$S \equiv \frac{c_s \Delta t}{\Delta x}$$

and note that if $S \leq 1$ then the CFL condition is met.

The "courant number" is C_0 where $0 \leq C_0 \leq 1$ and $\Delta t = C_0 \frac{\Delta x}{c_s}$. This courant factor is a sort of "safety factor". In general, you should choose $C_0 < 0.5$.

10 Curve Fitting

10.1 Modelling of Data

Given a set of observations one wants to summarize the data by fitting a model.

A model could be a simple function. For example, $y = mx + b$, where x and y are the independent and dependent variables, respectively, and where m and b are the model parameters.

To fit our model to observations, we need a "figure of merit" function that measures agreement between the model and observations. We need to consider

- Measurement error
- Accuracy of fitted model parameters
- Are there multiple solutions

10.2 Least Squares Estimator

We are fitting N data points $(x_i,\ y_i)$, where $i = 1, ..., N$, to a model with M adjustable parameters a_j, where $j = 1, ..., M$.

The simplest metric to minimize is

$$\sum_{i+1}^{N} [y_i - y(x_i, a_1, ..., a_m)]^2$$

This does not answer the question regarding probability. We can only answer this if we understand the uncertainty on y_i.

If we assume a Gaussian distribution and that the error on each y_i is independent then we can consider the Gaussian probability function

$$P_g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

where σ is the width (standard deviation) of the distribution and μ is the mean.

It is most common to see error bars represent 1σ.

The **Central Limit Theorem** states that the probability distribution of adding up a large number of small random deviates almost always converges to a Gaussian.

We can write down the probability of the dataset

$$P = \prod_{i=1}^{N} \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_i - y(x_i)}{\sigma_i}\right)^2\right)$$

where σ_i is the measured error of each measurement.

Maximizing P is equivalent to maximizing $\log P$ (natural logarithm).

$$\log P = \frac{1}{2}\left[\sum_{i=1}^{N}\left(\frac{y_i - y(x_i)}{\sigma_i}\right)^2 - \frac{1}{2}N\left(\log(2\pi) + \sum_{i=1}^{N}\log(\sigma_i)^2\right)\right]$$

Note that the last term is a constant.

We can now define

$$\chi^2 \equiv \sum_{i=1}^{N}\left(\frac{y_i - y(x_i)}{\sigma_i}\right)^2$$

This is known as *chi-square*.

We say that the "likelihood" L is

$$\log L = -\frac{1}{2}x^2 + C$$

Where C is a constant.

10.3 Straight Line Fitting

We would like to fit data to the model $y(x) = y(x_i, a, b) = a + bx$. This is commonly called "Linear Regression."

$$\chi^2(a,b) \equiv \sum_{i=1}^{N} \left(\frac{y_i - a - bx_i}{\sigma_i} \right)^2$$

We want to find a and b that minimize χ^2.

$$0 = \frac{\partial \chi^2}{\partial a} = -2 \sum_{i=1}^{N} \frac{y_i - a - bx_i}{\sigma_i^2}$$

$$0 = \frac{\partial \chi^2}{\partial b} = -2 \sum_{i=1}^{N} \frac{x_i(y_i - a - bx_i)}{\sigma_i^2}$$

If we let

$$S \equiv \sum_{i=1}^{N} \frac{1}{\sigma} \quad S_x \equiv \sum_{i=1}^{N} \frac{x_i}{\sigma} \quad S_y \equiv \sum_{i=1}^{N} \frac{y_i}{\sigma}$$

$$S_{xx} \equiv \sum_{i=1}^{N} \frac{x_i^2}{\sigma} \quad S_{xy} \equiv \sum_{i=1}^{N} \frac{x_i y_i}{\sigma}$$

Then our equations become

$$aS + bS_x = S_y \quad aS_x + bS_{xx} = S_{xy}$$

which has the solution

$$a = \frac{S_{xx}S_y - S_x S_{xy}}{SS_{xx} - (S_x)^2}$$

$$b = \frac{SS_{xy} - S_x S_y}{SS_{xx} - (S_x)^2}$$

The reduced chi square statistic is $\chi_2^2 = \frac{\chi^2}{N-1}$. A good fit would be $\chi_2^2 < 1$.

Now we need to estimate the uncertainty on the model parameters a and b.

If σ_i are independent then we can do a simple propagation of errors.

For any function f,

$$\sigma_f^2 = \sum_{i=1}^{N} \sigma_i^2 \left(\frac{\partial f}{\partial y_i} \right)^2$$

For a straight line,

$$\frac{\partial a}{\partial y_i} = \frac{S_{xx} - S_x S x_i}{\sigma_i^2 \Delta}$$
$$\frac{\partial b}{\partial y_i} = \frac{S x_i - S_x}{\sigma^2 \Delta}$$

where $\Delta = S S_{xx} - (S_x)^2$.
Summing for all i,

$$\sigma_a^2 = \frac{S_{xx}}{\Delta} \quad \sigma_b^2 = \frac{S}{\Delta}$$

will give the variances for a and b.

10.4 General Least Squares Fit

Instead of simple straight line, consider a polynomial of order $M - 1$

$$y(x) = a_1 + a_2 x + a_3 x^2 + ... + a_m x^{M-1}$$

This could easily be a series of sines or cosines or another other function. For example,

$$y(x) = a_1 + a_2 \sin(x) + a_3 \sin^2(x) + ... + a_m \sin^{M-1}(x)$$

What is important is that the function is linear with respect to the model parameters.

The general form is

$$y(x) = \sum_{k=1}^{M} a_k X_k(x)$$

where $X_i(x), ..., X_M(x)$ are arbitrary fixed functions called "basis functions".

$$\chi^2 = \sum_{i=1}^{N} \left[\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i^2} \right]^2$$

Let

$$A_{ij} \equiv \frac{X_j(x_i)}{\sigma_i}$$

This is the "design matrix." It is a N (data points) by M (model parameters) matrix. Traversing the rows, we have the data points and traversing the columns, we have the basis functions. In general, $N \geq M$ (more rows than columns). We will also define $b_i \equiv \frac{y_i}{\sigma_i}$, with $i = 1, ...n$, and model parameters

$a_k = a_1, ..., a_M$.

Starting with

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i, a_1, ..., a_M)}{\sigma_i^2} \right)^2$$

Taking the derivative with respect to a_k, for $k = 1, ..., M$,

$$0 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i)}{\sigma_i^2} \right) \left(\frac{\partial y(x_1, ..., a_k, ...)}{\partial a_k} \right)$$

$$0 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left(y_i - \sum_{k=1}^{M} a_k X_k(x_i) \right) X_k(x_i)$$

If

$$\alpha_{kj} = \sum_{i=1}^{N} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2}$$

$$\beta_{kj} = \sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2}$$

Note that $\alpha = A^T \cdot A$ and $\beta = A^T \cdot b$. Then we have

$$\sum_{j=1}^{M} \alpha_{kj} a_j = \beta_k$$

This is known as the "Normal" equations of the Least Squares Problem.

$$\implies \alpha \cdot \alpha = \beta$$

$$\implies (A^T \cdot A) \cdot a = A^T \cdot b$$

We can solve for a using matrix methods (such as Gauss-Jordan, LU Decomposition, or Singular Value Decomposition (SVD)), and $\sigma^2(a_j) = \alpha_{jj}^{-1}$.

10.5 Non-Linear Models

The least squares method is linear and so we can only solve for the best parameters for certain models.

We cannot, for instance, solve

$$f(x) = a_0 + a_1 \exp\left(-\frac{(a_2 - x^2)^2}{a_3}\right)$$

Non-linear problems require initial guesses for your fitted parameter.

The Levenberg-Marquadt method is done with
$\chi^2(a) \rightarrow \boldsymbol{\nabla}\chi(a)\lambda a' \rightarrow \chi^2(a')$
Where a' in the first part is a guess. Using the gradient to get closer to the desired value of the true best model parameter.

Let us say we have 3 data points, with 3 solutions. There are multiple solutions that are valid, so we are not looking for very precise values of this. For instance, we could have a sinusoidal wave model and a line model that map to the data points. Which one is best fit? It is the line model, because it has less model parameters (1 or 2) instead of the sinuosidal model (which has 3 parameters). This is occam's razor. It is tracked by the evidence term in Bayesian statistics, which is not so present in frequentist stats.

10.6 Bayesian Statistics

The formula for Bayesian stats is

$$P(n|d) = \frac{P(n)P(d|n)}{P(d)}$$

where $P(n|d)$ is the posterior, $P(n)$ is the prior, $P(d|n)$ is the likelihood, and $P(d)$ is the evidence.

The Bayesian Information Criteria (BIC) can be used to determine the better model. It is

$$\text{BIC} = \chi^2 + k\log(n)$$

where χ^2 is the chi square, k is the number of parameters and n is the number of data points.

In the case of the line and sinusodial models (2 and 3 parameters) with the same χ^2. We have a change in the BIC of

$$\Delta\text{BIC} = 3\log(3) - 2\log(3) = \log(3)$$

So we should pick the line model.

10.7 Monte-Carlo Simulations

We can use Monte-Carlo simulations to assess model parameters.

We can start with a set of true value for the parameters. We can then do repeated measurements using the model. For each of the measurements, we can do chi-square minimization for each of the to obtain a set of parameters for each of the measurements.
What are the uncertainties on the model parameters? One thing we could do is look at a histogram of the measurements of the model parameters?
If the histograms follow a Gaussian we can find a standard deviation. If not, then we have to be careful about it. For example, we can take the 68 percentile around the median, mean, or mode.

However, we need to tackle the scenario where we do not have many measurements.

Scenario 1: I have some measurements and I understand the measurement errors.

We have seen how to generate uniform random numbers. How do we generate a Gaussian random distribution? We could use the cumulative distribution function, or pick random numbers in a set volume and see if the value is inside or outside of the distribution function.

Then, we can generate trail sets by gaussian random picking for each of the data given the measurement uncertainties. And then create a histogram and find the 68 percentile as before.

Scenario 2: I have measurements but I do not know or understand the measurement errors.

A technique that is very good for this scenario is the Bootstrap method. It goes as follows. If I have m measurements,

- Generate a new dataset with m measurements using replacement
- This means that some measurements are duplicated and some are not included.

This is case replacement means that we could take a value out of the distribution and check the value, and then replace it and repeat. This is called sampling

with replacement.

The procedure is

1. Draw new sample with replacement (this means 1/e of the sample is duplicated)
2. Calculate χ^2
3. Repeat 1 and 2
4. Measure σ on your model

The assumption here is that the scatter of the underlying points are equal. The bootstrap method will not work when your model depends on time sampling intervals (very regular sampling). For instance, Fourier problems are not well suited.

10.8 Summary of Techniques

For estimating errors on fitted parameters when you know your measurement errors σ_i.

1. Generate synthetic dataset using σ_i
2. Find χ^2_{min} to get m' and b'
3. Repeat 1 and 2 to create a sample of m and b
4. Measure standard deviation

For estimating errors on fitted parameters when do not know your measurement errors σ_i.
For the bootstrap method, you create new sample by drawing from your data with replacement.

10.9 Markov-Chain-Monte-Carlo

Recall that the likelihood L is

$$\log L = -\frac{1}{2}\chi^2 + C$$

It is intractable to simply generate the chi-squared values for a bunch of model parameters (over all parameter space).

Instead, we would like to have a method to travel the parameter space. This is Markov-Chain-Monte-Carlo (MCMC).

We start at a certain spot in parameter space and then find the likelihood for that value and then we make a trail at a small step in a certain direction. If the likelihood of this trail value is higher than that of the current likelihood, then, based on a certain probability, we either accept or decline the jump.

10.9.1 Metropolis-Hastings Sampler

The algorithm is

1. Initialize a "chain" with $n = 1$, $x_n = x_1$
2. Generate a trial state x' using a transition function $g(x'|x_n)$
3. Calculate the likelihood for the trial state
4. Determine the acceptance probability $\alpha(x'|x)$
5. Draw a certain number $u = [0, 1]$
6. If $u \leq \alpha(x'|x)$ then $x_{n+1} = x'$, otherwise $x_{n+1} = x_n$.
7. $n = n + 1$
8. Go to step 2

where n is the chain element and x is the state (model parameters). For example, a straight line model has $x_1 = m_1, b_1$.

An example of a **transition function** is

$$g(x'_\mu|x_\mu) = \frac{1}{\sqrt{2\pi B_\mu}} \exp\left[-\frac{(x'_\mu - x_\mu)^2}{2\beta_\mu^2}\right]$$

with β is the "beta parameter", which we choose. It should be "close" to your expectation for the error in your model parameters. This is known as the Gibbs sampler.

The **acceptance probability** is

$$\alpha(x'|x) = \min\left|\frac{p(x'|d)}{p(x|d)}, 1\right|$$

where $p(x'|d)$ is the probability of the state x' given your dataset "d". For the moment, $p(x'|d) = L(x'|d)$l , where L is the likelihood.

We would like to choose the β parameter such that we get an acceptance of 20% to 30%.

MCMC requires that $q(x'|x) = q(x|x')$, so we must be careful when changing β dynamically. After the burn in phase, β must be constant for the chain generation.

After the simulation, we obtain a Markov chain, which is a series of states for the system (model parameters). This allows us to obtain information such as the average, standard deviation, or even correlation between the model parameters.

Returning to the chi square implementation,

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{f_i - d_i}{\sigma_i} \right)^2$$

where f_i is the model function, d_i is the data, and σ_i is the data uncertainty. Then the likelihood L is

$$\log L = -\frac{1}{2}\chi^2 + C$$

However, what if we are not certain about the values of σ_i. We could try adding a ξ model parameter such that

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{f_i - d_i}{\xi\sigma_i} \right)^2$$

Note that C has to do with the normalization, so it must be changed when changing the uncertainties like this.

If we expect to have multiple maximum likelihood locations in parameter space, we can simply run a lot of MCMC simulations with different initial parameters to explore parameter space properly.

If we have χ^2 contours where the parameters are highly correlated, then the MCMC will not have a clean distribution in the end. We would like to have our Markow-chain to be random and have no discernible features.

To evaluate whether a dataset has converged or not, we can use a sinusoidal parameter with a small periodicity.

We can use decorrelated MCMC (DeMCMC) to get past issues with model parameter correlation. This is done by first running an initial state using Metropolis-Hastings (M-H) and hten save the result in a buffer. Using a random

variable we will decide to either run M-H again, or do DeMCMC on a trial state. Afterwards, if we can tell that the Markov-chain converged, then we are done, otherwise, we save in the buffer and repeat again.

DeMCMC is done with

$$x' = x_n + \delta(x_{B_2} - x_{B_1})$$

where x_{B_1} and x_{B_2} are two random states taken from the buffer and δ is a factor similar in its role to β.

10.10 Introduction to Gaussian Processes

Thus far, we have considered noise to be independent. This is not always the case. A good textbook on this is found at http://www.gaussianprocess.org/gpml/chapters/.

The reduced chi-square is

$$\chi^2_R = \frac{\chi^2}{N-1}$$

If the error is representative of the scatter in the dataset, we would have $\chi^2_R \approx 1$. If we have a large chi^2_R adn we expect the model to be correct, we can set teh reduced chi-square to be 1, but this is dangerous.

Definition of Gaussian processes is __.

Instead of using chi-square, we can use the likelihoods

$$\log p(y|x,\sigma) = -\frac{1}{2}\sum_{i=1}^{N}\frac{y_i - f(x_i)}{\sigma_i^2} + \log 2\pi\sigma_n^2$$

This is identical as writing this in matrix form as

$$\log p(y|x,\sigma) = -\frac{1}{2}r^T C^{-1} - \frac{1}{2}\log\det C - \frac{N}{2}\log 2\pi$$

where C is a diagonal matrix containing the uncertainties, and $r = [y_1 - f(x_1), y_2 - f(x_2), ...]$.

We can then substitute C with a kernel K

$$K(x_i, x_j, \theta) = k_\theta(x_i, x_j) + \delta_{ij}\sigma i$$

This describes the covariance between data points. This gives us a multi-variate Gaussian. We can now model noise.

One example we can choose is the squared exponential model:

$$k_A = \sigma_A^2 \exp\left[\frac{(x_i - x_j)^2}{2l^2}\right]$$

for a length scale l which determines the wiggles in the function. This says that the correlation between any two points to decrease with time.

A linear model using Bayesian linear regression is

$$k_B(x_i, x_j) = \sigma_B^2 + \sigma_\nu^2(x_i - c)(x_j - c)$$

The extra parameters are called hyper-parameters, which are parameters for the model of the noise we have. We can take

$$k = k_A + k_B$$

From this, we can get a predictive mean and covariance.

Something that is important to do is to choose a correct kernel. If we make the kernel too complicated, we will over-fit the data. This needs to match what we expect for the correlation of the noise in the dataset.

Inverting matrices is a very intensive computational task, so we can use Cholesky decomposition for K since it is a Hermitian, positive-definite matrix,

$$K = LL^T$$

where L is a lower-triangular matrix. We also get the determinant for free using this decomposition also, since the determinant of K is the trace of L.

We can optimize these hyper-parameters using the Broyden-Fletcher-Goldfarb-Shanno algorithm, for example.

Something to consider is that we are still treating errors and uncertainties as Gaussian. This is convenient, but not necessarily true. Also note that the Kernel model should be motivated by something physical. We can also combine MCMC with these Gaussian processes to get distributions of parameters and hyper parameters.