PHY 325 Notes

Computational Physics

Jaeden Bardati

Last modified February 17, 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.

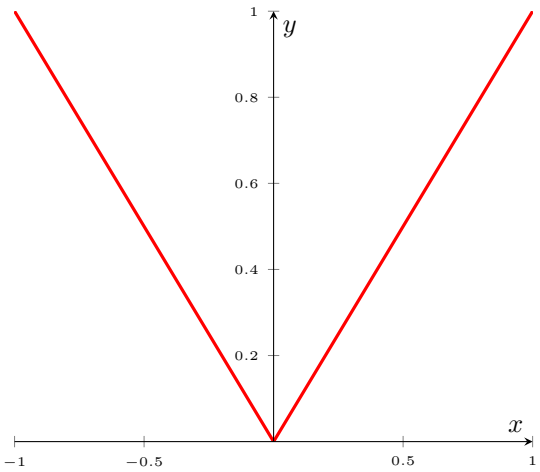

Figure 1: Absolute value function

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

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