Random Processes

(1) Need to generate pseudo random numbers

Plenty of algorithms out there that will do this for us

One of the most famous is called a Linear Congruential Sequence (LCS)
Famous - yes, Also simple albeit flawed

How does an LCS work

Consider the following equation

$\tilde{x} = (ax + c) \% m$ $\int$ LCS

where a, c, x, m are integers

Need to pick a seed ie x_0
a, c and m are constants

The sequence is completely deterministic

The algorithm is extremely simple and fine for simple problems

In practice people use whats called the mersenne twister algorithm (I think the basic python implementation uses MT)

<u>Python</u>

```
def LCS(N, a, c, m):

    x_array = []
    x = 687

    for i in range(N):
        x = (a*x + c) % m
        x_array.append(x)

    return x_array
```

Non-Uniform Random Numbers

Last week we discussed generating uniform variate random numbers
→ numbers in the interval $[0, 1]$
but in general we often want to generate non-uniform random numbers
For example random numbers that follow some distribution $p(x)$

eg $p(x) = Ae^{-x}$

Suppose you have a source of random floating point numbers z drawn from a distribution with probability density $q(z)$

Suppose also that you have a function $x = x(z)$

Then, by definition, when z is a random number so is $x(z)$ but with a different probability distribution $p(x)$

Our goal then is to choose a function $x(z)$ so that x has the distribution we want

So in this case the probability of generating a value of x between x and x+dx is by definition equal to the probability of generating a value of z in the corresponding z interval is

$$\underbrace{p(x)dx = g(z)dz}\quad (1)$$

But z can be uniform variate and $g(z) = 1$ between $z = 0$ and 1

Transformation Method

Use uniform variate random numbers to generate non-uniform random numbers

Integrating equation (1) gives us

$$\int_{-\infty}^{z} g(z)\,dz = z \qquad \left(\begin{array}{c}\text{remember } z \text{ is} \\ \text{uniform variate}\end{array}\right)$$

$$\therefore \quad z = \int_{-\infty}^{x(z)} p(x)\,dx \qquad (2)$$

So now if we can do this integral then we are in great shape. Unfortunately we can rarely do this integral though there are certain circumstances where it does work and we call the method the transformation method

Suppose we want to generate random real numbers x in the interval from zero to infinity with the exponential probability distribution

$$p(x) = \mu e^{-\mu x} \qquad (3)$$

Let's now take the equation (3) and sub into equation (2)

$$z = \int_{-\infty}^{x(z)} \mu e^{-\mu x}\, dx$$

$$= \frac{\mu e^{-\mu x}}{-\mu} \Bigg|_{-\infty}^{x(z)} = 1 - e^{-\mu x}$$

so $\quad z = 1 - e^{-\mu x} \qquad (4)$

$$x = -\frac{1}{\mu} \ln(1-z)$$

Gaussian Random Numbers

A common problem in physics is the question of distributions that are gaussian

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

Applying equation (1) (i.e the transformation method) we get

$$z = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{z} \exp\left(\frac{-x^2}{2\sigma^2}\right) dz$$

Problem no Analytic Solution

In this case we need to apply tricks that allow us to use the transformation method in a roundabout way

Imagine in this case we have two independent random numbers x, y both drawn from a gaussian distribution (say x and y) with the same standard deviation σ

The probability that the point with position vector (x, y) falls in some small element $dx\,dy$ is given by

$$p(x)\,dx\,p(y)\,dy = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) dx\,dy$$

Now for our second trick switch to polar coordinates (left as exercise)

$$p(r, \theta)\,dr\,d\theta = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right) r\,dr\,d\theta$$

$$= \frac{r}{\sigma^2} \exp\left(\frac{-r^2}{2\sigma^2}\right) dr \frac{d\theta}{2\pi}$$

$$= p(r)\,dr\,p(\theta)\,d\theta$$

Where $p(r) = \frac{r}{\sigma^2} \exp\left(\frac{-r^2}{2\sigma^2}\right)$

$$p(\theta) = \frac{1}{2\pi}$$

So now again we have constructed a transformation. Generating (r, θ) is equivalent to generating (x, y).

θ is trivial $\Rightarrow$ uniform variate between 0 and 2π

For r we must use a transformation method so

$$\frac{1}{\sigma^2}\int_{-\infty}^{r} \exp\left(\frac{-r^2}{2\sigma^2}\right) r\,dr$$

$$= 1 - \exp\left(\frac{-r^2}{2\sigma^2}\right) = z$$

or $r = \sqrt{-2\sigma^2 \ln(1-z)}$ (so we generate r from z

We still need x (and y) of course but

$x = r\cos\theta$
$y = r\sin\theta$

Monte Carlo Integration and Importance Sampling

Basic Approach

Suppose we want to evaluate the following integral

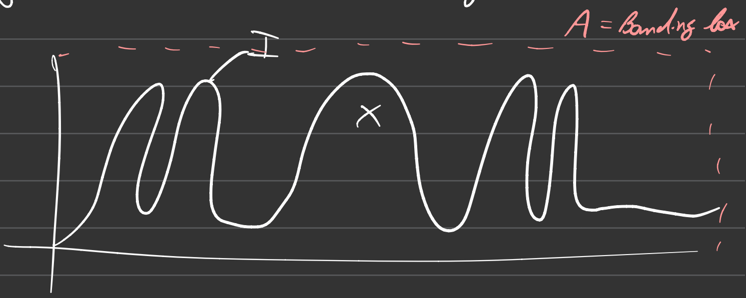
$$I = \int_0^2 \sin^2\left(\frac{1}{x(2-x)}\right)dx$$

So you can absolutely do this numerically but it is challenging. The reason being is that the integral varies wildly. MC integration can work better in these cases

How does MC integration work?

The basic idea is that we sample using random numbers, the area under the integrand. Say the bounding box is A, then the probability of the point falling under the integrand will be

$$p = \frac{I}{A}$$



So then the integral is simply

$$I = \frac{kA}{N} \quad \left(\begin{array}{l} N \text{ is the number of points} \\ \text{and } k \text{ is number of successes} \end{array} \right) \qquad (1)$$

But we can improve

Mean Value Method

Consider a general integration problem of the form

$$I = \int_a^b f(x)\,dx$$

The average value of $f(x)$ in the range (a, b) is by definition

$$\langle f \rangle = \frac{1}{b-a} \int_a^b f(x)\,dx$$

$$= \frac{I}{b-a}$$

or $I = (b-a)\langle f \rangle$

If we can estimate $\langle f \rangle$ accurately then we can estimate I

A simple way to estimate f then is to sample $f(x)$ using random numbers so

$$\langle f \rangle \approx N^{-1} \sum_{i=1}^{N} f(x_i) \quad \left(\begin{matrix}\text{where } x_i\text{'s are} \\ \text{uniform variates}\end{matrix}\right)$$

then
$$I = \frac{b-a}{N} \sum_{i=1}^{N} f(x_i) \qquad (2)$$

The next step is to build on the MV method by weighting the different values we choose

Consider the following

For any function $g(x)$ we can define a weighted average over the interval from a to b as

$$\langle g \rangle_w = \frac{\int_a^b w(x)\, g(x)\, dx}{\int_a^b w(x)\, dx} \qquad (3)$$

where $w(x)$ is some weight function

Now with this in mind consider again the 1-D integral

$$I = \int_a^b f(x)\,dx$$

Setting $y(x) = \frac{f(x)}{w(x)}$

in equation (3) we set

$$\left\langle \frac{f(x)}{w(x)} \right\rangle = \frac{\int_a^b w(x) \frac{f(x)}{w(x)} dx}{\int_a^b w(x)\,dx}$$

$$= \frac{\int_a^b f(x)\,dx}{\int_a^b w(x)\,dx}$$

$$= \frac{I}{\int_a^b w(x)\,dx}$$

$$I = \left\langle \frac{f(x)}{w(x)} \right\rangle_w \int_a^b w(x)\,dx \qquad (4)$$

So then how do we calculate

$$\int_a^b w(x)\,dx$$

we can start by defining a probability density

$$p(x) = \frac{w(x)}{\int_a^b w(x)\,dx} \qquad (5)$$

Let us sample N random points x_i non-uniformly with this density $p(x)$ and so for any function $g(x)$ we would get

$$\sum_{i=1}^{N} g(x_i) \approx \int_a^b N p(x)\, g(x)\,dx$$

or

$$\langle g \rangle_w = \frac{\int_a^b w(x)\, g(x)\,dx}{\int_a^b w(x)\,dx} \qquad \text{(equation (3) again)}$$

$$= \int_a^b p(x) y(x)\, dx \qquad \text{(definition of } p(x))$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} y(x) \qquad (6)$$

Combine (4) and (6) to get

$$I = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{w(x_i)} \int_a^b w(x)\, dx$$

This is importance sampling. If you set $w = 1$, then this is just the mean value method again.

Rootfinding

Systems of linear equations ($Ax = b$) are typically solved using algorithms such as

- Gaussian elimination
- LU decomposition (of A) $A = LU$ → upper triangular; ↳ lower triangular
- QR decomposition (of A)
  unitary matrix; radical part

Here we will study methods for solving non-linear systems. Namely, we look at methods for finding roots of functions

Note that if we have a method for finding x such that $f(x) = 0$, then we can find x' such that

$$f(x') = b$$

by finding roots of

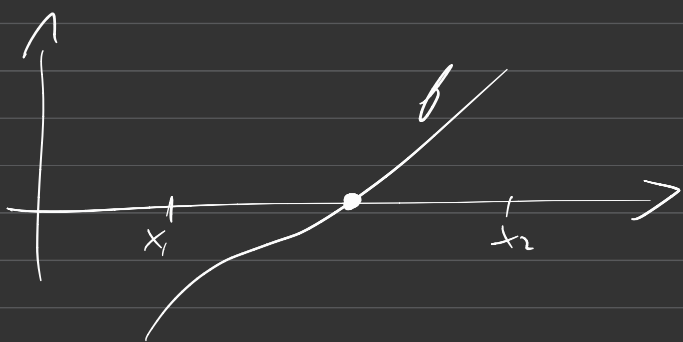
$$g(x) = f(x) - b$$

Bisection Method (1-dim case)

This method works by selecting an interval which contains a root of f and iteratively cutting the interval in half until it "zooms in" on the root

Steps

(1) Start with an interval pair of roots x_1 and x_2 which bracket a root of f and a target accuracy δ (minimum length of an interval)



Note that points x_1 and x_2 bracket a root if $f(x_1)$ and $f(x_2)$ have opposite sign

(2) Compute the midpoint x_m

$$x_m = \frac{x_1 + x_2}{2}$$

(3) Compute $f_m = f(x_m)$

(4) If $f_m = 0$ then stop and return x_m

(5) If f_m has the same sign as $f(x_1)$ then set $x_1 = x_m$ else $x_2 = x_m$

(6) If $|x_1 - x_2| > \delta$ then repeat steps (2) to (6) else stop and return

$$\frac{x_1 + x_2}{2}$$

Newton's Method

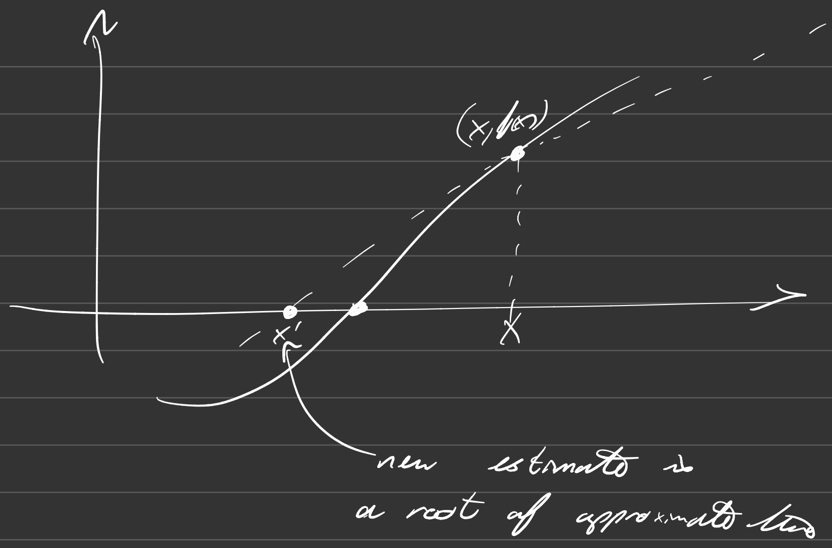
This method works by repeatedly approximating f as a straight line to improve current estimate of a root

Steps

(1) Begin with an initial estimate of the root x_0, a max number of iterations and a maximum tolerance

(2) Compute the new estimate as

$$x' = x - \frac{f(x)}{f'(x)}$$



(3) If the number of iterations $\geq$ max and

$|f(x)| > \varepsilon$

then go back to step (2)

(4) output x as answer

Secant Method

This method is essentially the same as newton's method, but the derivative of f is replaced with the numerical derivative of f (useful when f' is difficult to comput)

Recall $f'(x_2) \approx \frac{f(x_2) - f(x_1)}{x_2 - x_1}$

So the update step in the secant method looks like

$$x_3 = x_2 - f(x_2)\left(\frac{x_2 - x_1}{f(x_2) - f(x_1)}\right)$$

where x_1 and x_2 are previous estimate

Notice that this method begins with 2 values x_0 and x_1 unlike Newtons methods.

Maxima and Minima of functions

It's often the case of computer modelling that we want to find the minimum of some function $f(x)$

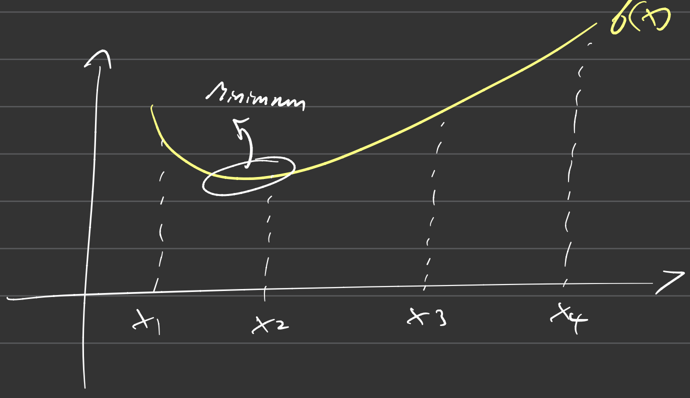
Generally speaking we can just take the partial derivative

so $\frac{\partial f(x)}{\partial x} = 0$

and find the minimum that way

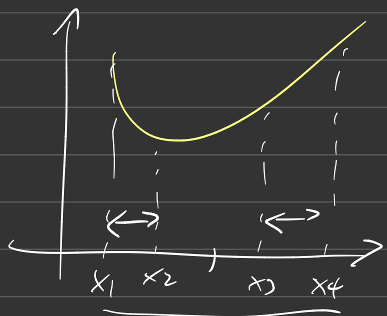
However, numerically this isn't allways possible or perhaps convenient. Sometimes we just have data points and no exact function, so sometimes we need another method

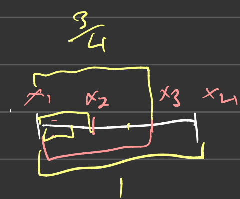
Golden Ratio search



(1) For given positions of the first and last points x_1 and x_4, the two interior points should be symmetrically distinct about the midpoint of the interval

ie $\quad x_2 - x_1 = x_4 - x_3 \qquad (1)$





$$\frac{1}{3/4} = \frac{4}{3} = z_B$$

(2) To fix x_2 and x_3 we need some additional information what we do is set up the procedure such that the ratio between the width of the bracketing intervals is the same before and after the next step

Before, $$z_B = \frac{x_4 - x_1}{x_3 - x_1} = \frac{x_2 - x_1 + x_3 - x_1}{x_3 - x_1} \qquad (2)$$

$$= \frac{x_3 - x_1 + x_2 - x_1}{x_3 - x_1} = \frac{x_2 - x_1}{x_3 - x_1} + 1 \qquad (3)$$

After, $$z_A = \frac{x_3 - x_1}{x_2 - x_1}$$

Equate the before and after steps st

$z_A = z_B$

$$\Rightarrow \boxed{\frac{x_3 - x_1}{x_2 - x_1} = \frac{x_2 - x_1}{x_3 - x_1} + 1} \quad (\text{subbing in } (3))$$

$$= \frac{1}{z} + 1$$

so $z = \frac{1}{z} + 1$

$$z = \frac{1 + \sqrt{5}}{2} \approx 1.618$$

This is the golden ratio and it is uniquely defines how we calculate x_2 and x_3 at each step

(1) Choose two initial points x_1 and x_4. Then calculate x_2 and x_3 according to the golden ratio. For this you need to calculate

$$x_2 = f(z, x_1, x_4)$$

$$x_3 = f(z, x_1, x_4)$$

(2) If $f(x_2) < f(x_3)$ then the minimum lies between x_1 and x_3

Set $x_4 = x_3$

$x_3 = x_2$

and calculate the new x_2 using the golden ratio x_1 remains the same

(3) Else $f(x_2) > f(x_3)$

In this case the minimum lies between x_2 and x_4

Then set

$x_1 = x_2$

$x_2 = x_3$

and calculate the new x_3 using the golden ratio, x_4 remains the same

(4) Continue this iteration until

$|x_4 - x_1| <$ Threshold

Once this threshold is reached the

minimum is

$$\frac{1}{2}(x_1 + x_4)$$

(5) Before attempting to calculate

$$x_2 = f(z, x_1, x_4)$$

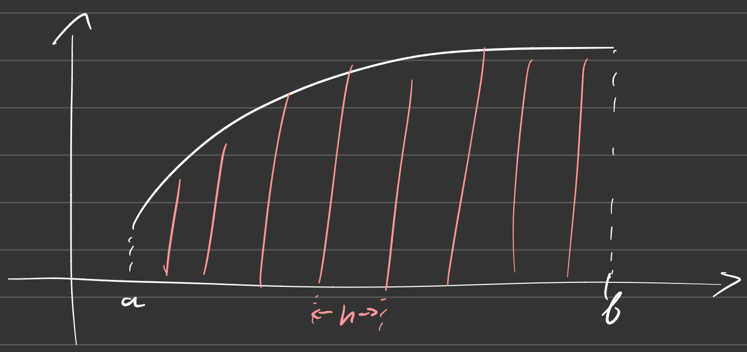
$$x_3 = f(z, x, x_4)$$

Numerical Integration

Trapezoidal Rule
Simpsons Rule
Gaussian quadrature
Infinite integrals

Trapezoidal Rule

(np.trapz)



For the trapezoidal rule the domain of integration is divided into segments. The width of each segment is given by

$$\frac{b-a}{N} = h$$

The area of each trapezoid is given by

$$A_k = \frac{1}{2} h \left[f(a+(k-1)h) + f(a+kh) \right]$$

and

$$I(a,b) = h \left[\frac{1}{2} f(a) + \frac{1}{2} f(b) + \sum_{k=1}^{N-1} f(a+kh) \right] \qquad (1)$$

<u>Example</u>

$$I = \int_0^2 (x^2 - 2x + 1)\, dx$$

$N = 10$
$a = 0$
$b = 2$

```
def f(x):
    return (x**4 - 2*x + 1)

h = (b-a)/N

s = 0.5*f(a) + 0.5*f(b)

for k in range(1, N)
    s += f(a+k*h)
```

Simpson's Rule

(scipy.integrate.simpson)

We can do better using Simpsons Rule which instead of using trapezoids uses quadratic functions to subdivide the area under the curve. This leads to the formula

$$I(a,b) = \frac{1}{3}h\Big[f(a) + f(b) + 4\sum_{\substack{k \text{ odd}\\ 1,\dots,N-1}} f(a+kh)$$
$$+ 2\sum_{\substack{k \text{ even}\\ 2,\dots,N-2}} f(a+kh)\Big]$$

Note the two different sums and also that N must be even

Errors and Adaptive Integration

The trapezoid rule is known to introduce errors of order h^2

Suppose that the true value of our integral is I and let us denote our first estimate using the trapezoid rule with N_1 steps by I_1

Then

$$I = I_1 + ch_1^2 \quad (c \text{ some constant})$$

Say we just increase N now

$$I = I_2 + ch_2^2$$

Combining the previous two equations we get that

$$I_1 + ch_1^2 = I_2 + ch_2^2$$

$$I_2 - I_1 = c(h_1^2 - h_2^2)$$

$$= 3ch_2^2 \quad \left(\text{assuming } h_2 = \frac{h_1}{2}\right)$$

or $\varepsilon = \frac{1}{3}(I_2 - I_1) = ch_2^2$

This error estimate sets a basis for adaptive integration

Consider the following algorithm

(1) Choose an initial number of steps N, and decide on the target accuracy for a given integral (eg 5 significant digits)

(2) Calculate the first approximation to the integral using N_1

(3) Double the number of steps and calculate the updated integral using the formula

$$I_i = \frac{1}{2} I_{i-1} + h_i \sum_{\substack{k \text{ odd} \\ 1,\ldots,N-1}} f(a + kh_i)$$

(4) Calculate the error using

$$\varepsilon = \frac{1}{3}(I_{i-1} - I_i)$$

(5) Continue until

$$\varepsilon < TARGET$$

Gaussian Quadrature Integration

So far we have talked about the Trapezoidal Rule and Simpson's rule. In both of these cases we have used fixed points to evaluate the integral

Gaussian Quadrature is significantly more powerful as the points it chooses are non-uniform and more tailored to the problem at hand

Gaussian integration uses so called Legendre Polynomials to construct the framework

Legendre Polynomials

$$\ell_k(x) = \prod_{\substack{m=1,\dots,N \\ m\neq k}} \frac{(x - x_m)}{(x_k - x_m)}$$

$\ell_k(x)$ is a polynomial of degree $N-1$. If we evaluate $\ell_k(x)$ at one of the sample points $x = x_m$ we get

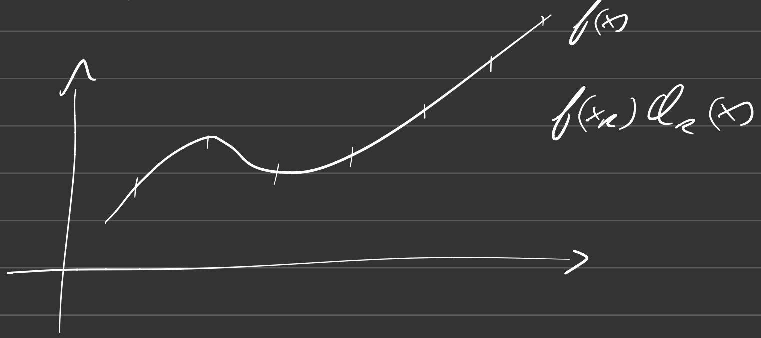
$$\left(\frac{x_m - x_1}{x_k - x_1}\right)\left(\frac{x_m - x_2}{x_k - x_2}\right)\cdots\left(\frac{x_m - x_n}{x_k - x_n}\right) = \begin{cases} 1 & \text{if } k = m \\ 0 & \text{if } k \neq m \end{cases}$$

$$l_k(x_m) = \delta_{km} \qquad \text{(Kronecker-delta function)}$$

So now lets consider the function

$$l(x) = \sum_{k=1}^{N} f(x_k)\, l_k(x)$$

This function, $l(x)$, is a linear combination (of degree $N-1$) that fits the function $f(x_k)$ at each sample point



$$\int_a^b f(x)\,dx = \int_a^b l(x)\,dx$$

$$= \int_a^b \sum_{k=1}^{N} f(x_k)\, l_k(x)\,dx$$

$$= \sum_{k=1}^{N} f(x_k) \int_a^b l_k(x)\,dx$$

$$= \sum_{k=1}^{N} f(x_k)\, w_k$$

these are called the weights

Remarkably using this technique we can integrate any function $f(x)$ by summing over $f(x_k)$ at a set of sample points and applying appropriate weight

The beauty of this method is that the weights do not depend on $f(x)$ and can be computed once for all $f(x)$!

Example

$$\int_0^2 (x^4 - 2x + 1)\,dx$$

```
from gaussxw import gaussxw

def f(x)
    return x^4 - 2x + 1

N = 3
a = 0
b = 2

# Calculate the sample points and weights
  and then apply them to
  our integral

x, w = gaussxw(N)
```

(these weights are only for the interval $[-1, 1]$)

```
xp = 0.5*(b-a)*x + 0.5*(b+a)
wp = 0.5*(b-a)*w
```

Preform integration

```
s = 0
for k in range(N):
    s += wp[k] * f(xp[k])
print("I = %g" % (s))
```

Infinite Integrals

Not surprisingly computers cannot integrate up to infinity. Instead what we do is we employ a change of variables so that the integral becomes tracktable

For an integral over the range from 0 to ∞ the standard change of variables is

$$z = \frac{x}{1+x} \quad \text{or equivalently}$$

$$x = \frac{z}{1-z}$$

then

$$dx = \frac{dz}{(1-z)^2}$$

$$\int_0^\infty f(x)\,dx = \int_0^1 \frac{1}{(1-z)^2} f\left(\frac{z}{1-z}\right) dz$$

Consider the following integral

$$I = \int_0^1 e^{-t^2}\,dt$$

We change the variable

$$x = \frac{z}{1-z} \qquad dx = \frac{dz}{(1-z)^2}$$

and the integral becomes

$$I = \int_0^1 \frac{\exp\left(\frac{-z^2}{(1-z)^2}\right)}{(1-z)^2}\,dz$$

ODE's

Runge Kutta 4th order Scheme

Ordinary differential equations take the form of an equation like

$$\frac{dx}{dt} = \frac{2x}{t} \qquad (1)$$

Solving ODE's (particularly second order and higher) is very common in computational physics. Many solutions exist and many libraries exist (eg odeint in python) to solve these equations for you

The most common/preferable approach is to use the so called RK4 scheme or Runge Kutta 4 scheme. RK4 can be used as part of odeint as can many other schemes. Here we will use the scheme explicitly

RK4 equations:

$$K_1 = hf(x, t)$$

$$K_2 = hf(x + \tfrac{1}{2}K_1, t + \tfrac{1}{2}h)$$

$$K_3 = hf(x + \tfrac{1}{2}K_2, t + \tfrac{1}{2}h)$$

$$K_4 = hf(x + K_3, t + h)$$

$$x(t+h) = x(t) + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

~~Example~~

Suppose we want to solve

$$\frac{dx}{dt} = -x^3 + \sin t$$

```
import numpy as np
import matplotlib.pyplot as plt

a = 0
b = 10
N = 10
h = (b-a)/N
```

```
tpoints = np.arrange(a, b, N)
xpoints = [ ]

x = 0.0

def f(x,t):
    return   -x^3 + sin(t)

for  t  in  tpoints:
    xpoints.append(x)

    K1 = h f(x,t)
    K2 = h f( x + 0.5*K1 ,  t + 0.5*h )
    K3 = h f(x + 0.5*K2 , t + 0.5*h)
    K4 = h f( x + K3 ,  t + h)

    x += (K1 + 2*K2 + 2*K3 + K4)/6

plt.plot( tpoints , xpoints
plt.xlable
plt.ylabel
plt.show()
```

ODE's, Solving for infinite Ranges

What if we want to study an ODE all the way out to infinity

This is solved in exactly the same way as with integration

We preform a change of variable

We define for the case $t \to \infty$,

$$u = \frac{t}{1+t}$$

or equivalently

$$t = \frac{u}{1-u}$$

and so as $t \to \infty$, $u \to 1$

Consider the following

$$\frac{dx}{dt} = f(x,t)$$

$$\frac{dx}{du}\frac{du}{dt} = f(x,t)$$

$$\frac{dx}{du} = \frac{dt}{du} f\left(x, \frac{u}{1-u}\right)$$

but $\frac{dt}{du} = \frac{1}{(1-u)^2}$

$$\Rightarrow \frac{dx}{du} = (1-u)^{-2} f\left(x, \frac{u}{1-u}\right)$$

If we define a new function $g(x,u)$ as

$$g(x,u) = (1-u)^{-2} f\left(x, \frac{u}{1-u}\right)$$

then $\frac{dx}{du} = g(x,y)$

and we solve this as before

ODEs with more than one Variable

Say we have the following

$$\frac{dx}{dt} = xy - x$$

$$\frac{dy}{dt} = y - xy + \sin^2 \omega t$$

The solution here is to write the ODEs in vector form. In this regard python is especially good.

so

$$\frac{dx}{dt} = f_x(x, y, t)$$

$$\frac{dy}{dt} = f_y(x, y, t)$$

then we write

$$\frac{dr}{dt} = f(r, t)$$

where $r = (x, y, \ldots\ldots)$

and f becomes a vector of functions and we solve as before

Second order ODEs

Second order ODEs are solved using the same technique as ODEs with multiple variables so we use vectors to solve the equations

Consider the following second order ODE

$$\frac{d^2x}{dt^2} = f\left(x, \frac{dx}{dt}, t\right)$$

eg

$$\frac{d^2x}{dt^2} = \frac{1}{x}\left(\frac{dx}{dt}\right)^2 + 2\frac{dx}{dt} - x^3 e^{-4t}$$

To turn this into a multivariate equation we define

$$y = \frac{dx}{dt}$$

$$\frac{d^2x}{dt^2} = \frac{dy}{dt}$$

(2-variable ODE
Solved using vectors

so $r[0] = x$
$r[1] = y$)

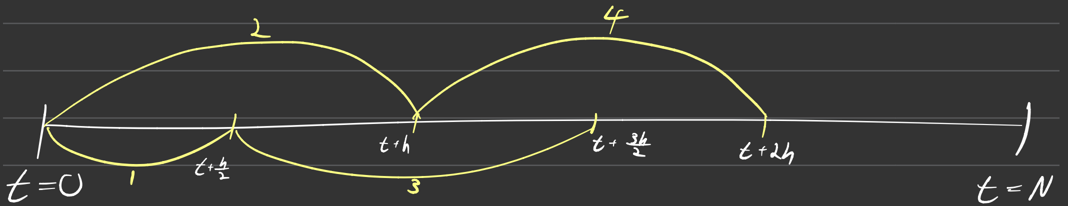
$$y = \frac{dx}{dt}$$

Leapfrog Integration

Consider a first order ODE

$$\frac{dx}{dt} = f(x,t) \qquad (1)$$

Break up interval into N stages



- The leapfrog method starts off with a $\frac{1}{2}$ step to the midpoint
- Next we have a full step to calculate $x(t+h)$ starting from $x(t)$
- Then for the next half step we calculate it from the midpoint value $x(t+\frac{h}{2})$

so $x(t+\frac{3h}{2}) = x(t+\frac{h}{2}) + hf(x(t+h), t+h)$

In this calculation $f(x(t+h), t+h)$ plays the role of the gradient at the midpoint between $t+\frac{h}{2}$ and $t+\frac{3h}{2}$

- Next we would have

$$x(t+2h) = x(t+h) + hf(x(t+\tfrac{3h}{2}), t+\tfrac{3h}{2}) \quad (4)$$

And we go on repeating this process as long as is required. Given values of $x(t)$ and $x(t+\frac{h}{2})$ we repeatedly apply the equations

$$x(t+h) = x(t) + hf(x(t+\tfrac{h}{2}), t+\tfrac{h}{2})$$

$$x(t+\tfrac{3h}{2}) = x(t+\tfrac{h}{2}) + hf(x(t+h), t+h)$$

Leapfrog is actually less accurate than RK4 in terms of cut and cut error propagation. So why ever use it?

The answer is that leapfrog is symplectic. This means that it is time symmetric. If you run leapfrog backwards you end up with identical equations (so $t \to -t$). Not true for RK4. RK4 is not ~~time~~ symmetric.

If you don't have time symmetry, then you don't conserve energy

Verlet (Extension of Leapfrog)

Generally applied to equations of motion. Consider Newton's equations

$$\frac{d^2x}{dt^2} = f(x,t)$$

eg $$\frac{d^2x}{dt^2} = -\frac{GM\vec{x}}{x^3}$$

(or the vector equivalent ($\vec{x} \to \vec{r}$) in more than 1-D)

We can convert this equation of motion into coupled first order equations

$$\frac{dx}{dt} = v \qquad (1)$$

$$\frac{dv}{dt} = f(x,t) \qquad (2)$$

If we want to apply the leapfrog method to these equations (ie (1) and (2)) the normal strategy would be to define a vector

$$\vec{r} = (\vec{x}, \vec{v})$$

and combine (1) and (2) into a single vector and solve, ie

$$\frac{d\vec{r}}{dt} = f(\vec{r}, t) \qquad (3)$$

Rather than going this route, however let us write out the leapfrog method in full as applied to (1) and (2)

If we are given the value of $\vec{v}$ at time $t + \frac{h}{2}$ and $\vec{x}$ at time t then applying LF we have that

$$x(t+h) = x(t) + hv\left(t + \frac{h}{2}\right) \qquad (4)$$

$$v\left(t + \frac{3h}{2}\right) = v\left(t + \frac{h}{2}\right) + hf(x(t+h), t+h) \qquad (5)$$

We have stuck to the original $\vec{x}$ and $\vec{v}$ (instead of $\vec{r}$) and derive a full solution to the problem by using just these two equations. Repeating these equations solves our problem an improvement over LF which would require $\vec{v}$ at both the full and half-points

It only works for specific types of equations where for example with Newton's equations the RHS of (1) depends only on $\vec{v}$ while the RHS of (2) depends only on $\vec{x}$

A slight issue arises when calculating total energies. In this case the velocity must be computed at the full step using an Euler step

Putting all of this together we can now write out ~~the~~ full procedure for Verlet:

Given the initial value of $\vec{x}$ and $\vec{v}$ at some time t

$$v\left(t+\tfrac{1}{2}h\right) = v(t) + \tfrac{1}{2}h f(x(t), t) \qquad (V1)$$

$$x(t+h) = x(t) + h v\left(t+\tfrac{h}{2}\right) \qquad (V2)$$

$$K = hf(x(t+h), t+h) \qquad (V3)$$

$$v\left(t+\frac{3h}{2}\right) = v\left(t+\frac{h}{2}\right) + K \qquad (\text{optional})$$

$$v\left(t+\frac{3h}{2}\right) = v\left(t+\frac{h}{2}\right) + K \qquad (V4)$$

Final Lecture

Leapfrog Integration

Given an ODE of the form

$$\frac{dx}{dt} = f(x, t)$$

(1) Calculate x at the $\frac{1}{2}$ step

$$x(t + \tfrac{h}{2}) = x(t) + \frac{h}{2} f(x, t)$$ (Euler Method)

(2) Now given the values of $x(t)$ and $x(t + \frac{h}{2})$ we repeatedly apply the leapfrog equation

$$x(t+h) = x(t) + h f\left(x(t + \tfrac{h}{2}), t + \tfrac{h}{2}\right)$$

$$x(t + \tfrac{3h}{2}) = x(t + \tfrac{h}{2}) + h f\left(x(t+h), t+h\right)$$

calculating values at both the full and half step

Verlet Integration

The verlet scheme is a derivative of leapfrog applied almost exclusively to equation of motion

Consider the following

$$\frac{d^2x}{dt^2} = f(x,t)$$

This can be written as

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = f(x,t)$$

For the special case of motion the verlet algorithm can be used

With applying leapfrog we would construct the vector

$$\vec{r} = (\vec{x}, \vec{v})$$

and solve at both the full step and half steps.

However, Verlet allows us to be efficient. Suppose we know x at t_0 and v at $t_0 + \frac{h}{2}$ then the verlet algorithm looks like

(1) $v(t + \frac{1}{2}h) = v(t) + \frac{1}{2} f(x, t)$

(2) $x(t + h) = x(t) + h\, v(t + \frac{h}{2})$

(3) $v(t + \frac{3h}{2}) = v(t + \frac{1}{2}h) + h f(x(t+h), t+h)$