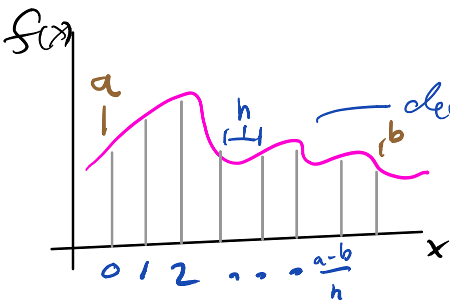
Integration

$$N(1) = \int_0^1 \dot{N}(t)\, dt$$



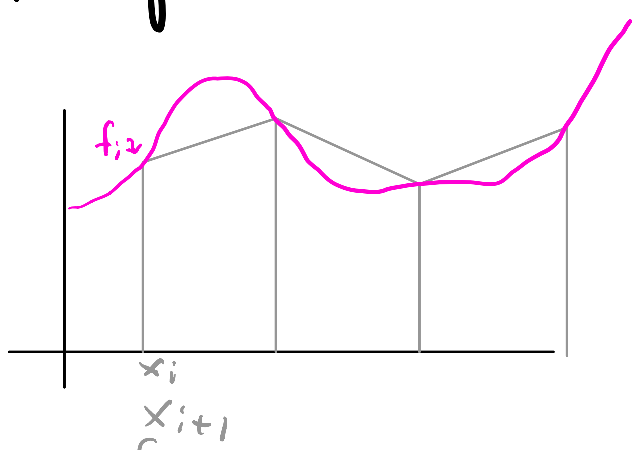
— divide into J bins

$$\int_a^b f(x)\, dx = \lim_{h \to \infty} h \sum_j^{\frac{a-b}{h}} f(x_j)$$

we shouldn't include a singularity:

$$\int_{-1}^{1} \cancel{|x|} f(x)\, dx = \int_{-1}^{0} f(-x)\, dx + \int_0^1 f(x)\, dx$$

Trapezoid Rule



- evenly spaced bins
- include endpoints

$$\int_{x_i}^{x_{i+1}} f(x)\, dx \simeq \frac{1}{2} h f_i + \frac{1}{2} h f_{i+1} = h\left(\frac{1}{2} f_i + \frac{1}{2} f_{i+1}\right)$$

Now sum all points:

$$\int_a^b f(x)\,dx = h\left(\tfrac{1}{2}f_1 + f_2 + \cdots + f_{N-1} + \tfrac{1}{2}f_N\right)$$

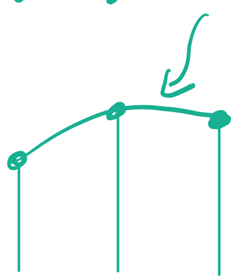
weights $w_i = h\left[\tfrac{1}{2}, 1, \ldots, 1, \tfrac{1}{2}\right]$

$$\frac{1}{2}\left[\sum_{i=1}^{N-1} f_i + \sum_{i=2}^{N} f_i\right]$$

Simpson's rule:

$$f(x) \simeq f'(x) = Ax^2 + Bx + C$$

$$\begin{array}{cccc} f_1 & f_2 & \cdots & f_{N-1} \\ + & + & & + \\ f_2 & f_3 & \cdots & f_N \end{array}$$



x_{i-1} x_i x_{i+1}

$x=-1$ 0 $x=+1$

First, what is the area?

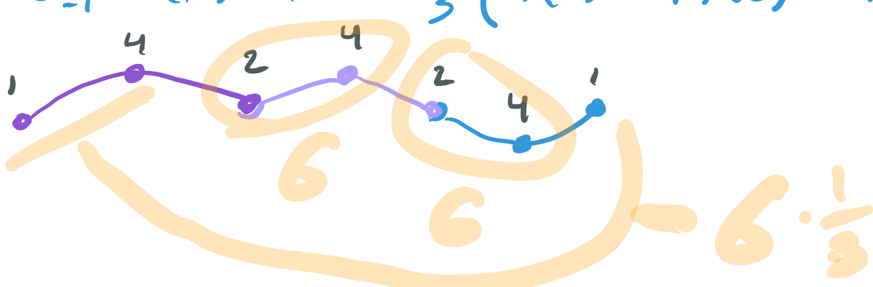
$$\int f'(x)\,dx = \tfrac{1}{3}Ax^3 + \tfrac{1}{2}Bx^2 + Cx$$

$$\int_{-1}^{1} f'(x)\,dx = \tfrac{2}{3}A + 2C$$

Given that
$$f'(-1) = A - B + C$$
$$f'(0) = C$$
$$f(1) = A + B - C$$

$$2A = f(1) + f(-1)$$
$$2B = f(1) - f(-1)$$

$$\int_{-1}^{1} f'(x)\,dx = \tfrac{1}{3}\left(f(-1) + 4f(0) + f(1)\right)$$



$$\sum_{i=1}^{N} w_i = (N-1)h$$

$$\begin{array}{c|c|c|c}
f_1 & f_3 & & f_{N-2} \\
+ & & & + \\
4f_2 & 4f_4 & \cdots & 4f_{N-1} \\
+ & & & + \\
f_3 & f_5 & & f_N
\end{array}$$

$$\sum_{i=1}^{\frac{N-1}{2}} f_{2i-1} + 4\sum_{i=1}^{\frac{N-1}{2}} f_{2i} + \sum_{i=1}^{\frac{N-1}{2}} f_{2i+1}$$

Quick check:

$f_i = 1$, then:

$$\frac{1}{3}\left[2\left(\frac{N}{2}-1\right) + 4\left(\frac{N}{2}-1\right)\right] h$$

$$2\left(\frac{N-1}{2}\right) h = (N-1)h \checkmark$$

Integration error

Trapezoid

$$\frac{\delta f}{f} = \mathcal{O}\left(\frac{[b-a]^3}{N^2}\right) f^{(2)}$$

Simpson's

$$\frac{\delta f}{f} = \mathcal{O}\left(\frac{[b-a]^5}{N^4}\right) f^{(4)}$$

Roundoff errors

$$\frac{\delta_a}{f} \simeq \sqrt{N}\,\epsilon_d$$

Reminder

$$\begin{pmatrix} \epsilon_{f32} = 10^{-7} \\ \epsilon_d = 10^{-15} \end{pmatrix}$$

Let's set $\frac{f^{(n)}}{f} \simeq 1$, $b-a=1$, $h = \frac{1}{N}$

Then $\sqrt{N}\,\epsilon_d \approx \frac{f^{(2)}(b-a)^3}{f N^2} = \frac{1}{N^2} \rightarrow N \simeq 10^6$

$\sqrt{N}\,\epsilon_d = 10^{-12}$

similar calculations for Simpson's

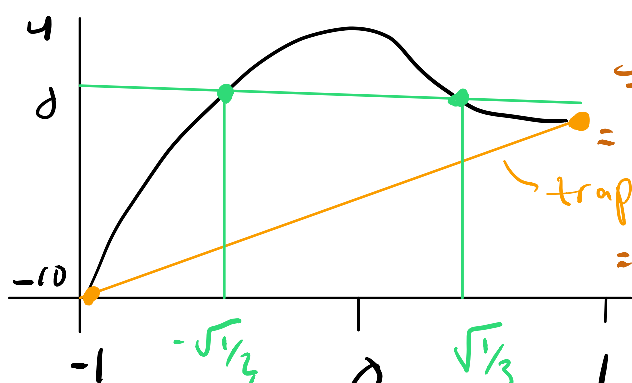
$N \simeq 2154$

$\sqrt{N}\,\epsilon_\alpha \simeq 5\times10^{-14}$

Gaussian Quadrature

$$\int_a^b f(x)\,dx \equiv \int_a^b W(x)\,g(x)\,dx \simeq \sum_{i=1}^{N} w_i\, g(x_i)$$

Select n points so that the resulting integral is perfect for a $2n-1$ polynomial.



$$\int_{-1}^{1} 7x^3 - 8x^2 - 3x + 3\,dx$$

$$= \frac{7}{4}x^4 - \frac{8}{3}x^3 - \frac{3}{2}x^2 + 3x\Big|_{-1}^{1}$$

$$= -2\,\frac{8}{3} + 2\cdot 3$$

$$= \frac{18}{3} - \frac{16}{3} = \frac{2}{3}$$

Steps

- remove singularities
- Select type of Quadrature
- Map interval to Gauss interval
- Compute function and weights at points

From our book

import numpy as np

```
from numpy import *
from sys import version

max_in = 11                              # Numb intervals
vmin = 0.; vmax = 1.                     # Int ranges
ME = 2.7182818284590452354E0             # Euler's const
w = zeros( (2001), float)
x = zeros( (2001), float)

def f(x):                                # The integrand
    return (exp( - x) )

def gauss(npts, job, a, b, x, w):
    m = i = j = t = t1 = pp = p1 = p2 = p3 = 0.
    eps = 3.E-14                         # Accuracy: ******ADJUST THIS*******!
    m = int((npts + 1)/2 )
    for i in range(1, m + 1):
        t = cos(math.pi*(float(i) - 0.25)/(float(npts) + 0.5) )
        t1 = 1
        while( (abs(t - t1) ) >= eps):
            p1 = 1. ;  p2 = 0.
            for j in range(1, npts + 1):
                p3 = p2;    p2 = p1
                p1 = ((2.*float(j)-1)*t*p2 - (float(j)-1.)*p3)/(float(j))
            pp = npts*(t*p1 - p2)/(t*t - 1.)
            t1 = t; t = t1  -  p1/pp
        x[i - 1] = - t;    x[npts - i] = t
        w[i - 1] = 2./( (1. - t*t)*pp*pp)
        w[npts - i] = w[i - 1]
    if (job == 0):
        for i in range(0, npts):
            x[i] = x[i]*(b - a)/2. + (b + a)/2.
            w[i] = w[i]*(b - a)/2.
    if (job == 1):
        for i in range(0, npts):
            xi = x[i]
            x[i] = a*b*(1. + xi) / (b + a - (b - a)*xi)
            w[i] = w[i]*2.*a*b*b/( (b + a - (b-a)*xi)*(b + a - (b-a)*xi))
    if (job == 2):
        for i in range(0, npts):
            xi = x[i]
            x[i] = (b*xi +  b + a + a) / (1. - xi)
            w[i] = w[i]*2.*(a + b)/( (1. - xi)*(1. - xi) )

def gaussint (no, min, max):
    quadra = 0.
```

global! bad.

also global

not needed

//2

make parameter

one based indexing! Just... No.

np.exp(1)

np.

np

one based again

don't use ;

could be array expressions

reorder to skip xi

*2

use **2 to square.

..2

using globals instead of return!

same name as built-in (ok)

```
    gauss(no, 0, min, max, x, w)                        # Returns pts & wts
    for n in  range(0, no):
        quadra   += f(x[n]) * w[n]                      # Calculate integral
    return quadra

for i in range(3, max_in + 1, 2):
    result = gaussint(i, vmin, vmax)
    print (" i ", i, " err ", abs(result - 1 + 1/ME))
print ("Enter and return any character to quit")
```

not needed