Chapter-07 Numerical Integration

Date	@November 1, 2024
tag	done

Midpoint Rule

interval [a,b] = [x0,x1]

$Z_0 = (a+b)/2$

1. **Divide the Interval**:
    - The interval [a,b] is divided into N equal subintervals.
    - The width h of each subinterval is given by:

      $h = \frac{b-a}{N}$
2. The endpoints of the subintervals are:

   $x_i = a + i \cdot h \quad \text{for } i = 0, 1, \ldots, N.$
3. **Identify the Midpoints**:
    - The midpoint zi of each subinterval [xi,xi+1] is:

      $$z_i = \frac{x_i + x_{i+1}}{2}$$
    - Since $x_{i+1} = x_i + h$, we can write:

      $z_i = \frac{x_i + (x_i + h)}{2} = x_i + \frac{h}{2}$
4. **Apply the Midpoint Rule**:
    - The midpoint rule approximation for each subinterval is:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx h \cdot f\left(z_i\right)$$

- Therefore, the integral over the entire interval [a,b] is approximated by:

$$I_N[f] \approx h \sum_{i=0}^{N-1} f\left(z_i\right)$$

N is directly prop to approximation

```
f = lambda x: np.sin(x)
N = 10 # intervals
a = 0.0; b= 2.0*np.pi; h=(b-a)/N
x = np.linspace(a, b, N+1)
z = (x[1:] + x[:-1]) / 2.0
w = h*np.ones_like(z)
quad = np.inner(w, f(z))
print('Result =', quad)
```

Result = -1.3139323596589723e-16

Let $N = 10$ (no of subintervals)

$a = 0$, $b = 2\pi$

1) $h = \frac{b-a}{N} = \frac{2\pi - 0}{10} = 0.2\pi$

2) Compute the Endpoints x_i of subintervals:

$x_i = a + i.h$ for $i = 0, 1, \ldots, N$

for $i = 0$ to 10:

$x_0 = 0 + 0(0.2\pi) = 0$

$x_1 = 0.2\pi$

$x_2 = 0.4\pi$

$x_3 = 0.6\pi$

$x_4 = 0.8\pi$

$x_5 = \pi$

$x_6 = 1.2\pi$

$x_7 = 1.4\pi$

$x_8 = 1.6\pi$

$x_9 = 1.8\pi$

$x_{10} = 2\pi$

3) Compute the Midpoints z_i

$$z_i = \frac{x_i + x_{i+1}}{2}$$

$$z_0 = \frac{x_0 + x_1}{2} = \frac{0 + 0.2\pi}{2} = 0.1\pi$$

$$z_1 = \frac{x_1 + x_2}{2} = \frac{0.2\pi + 0.4\pi}{2} = 0.3\pi$$

$$z_2 = \frac{0.4\pi + 0.6\pi}{2} = 0.5\pi$$

$$z_3 = \frac{0.6\pi + 0.8\pi}{2} = 0.7\pi$$

$$z_4 = \frac{0.8\pi + \pi}{2} = 0.9\pi$$

$$z_5 = \frac{\pi + 1.2\pi}{2} = 1.1\pi$$

$$z_6 = \frac{1.2\pi + 1.4\pi}{2} = 1.3\pi$$

$$z_7 = \frac{1.4\pi + 1.6\pi}{2} = 1.5\pi$$

$z_8 = \frac{1.6\pi + 1.8\pi}{2} = 1.7\pi$

$z_9 = \frac{1.8\pi + 2\pi}{2} = 1.9\pi$

4) Evaluate $f(z_i) = \sin(z_i)$ for each midpoint.

$f(z_0) = \sin(0.1\pi)$ $\quad$ $f(z_5) = \sin(1.1\pi)$

$f(z_1) = \sin(0.3\pi)$ $\quad$ $f(z_6) = \sin(1.3\pi)$

$f(z_2) = \sin(0.5\pi)$ $\quad$ $f(z_7) = \sin(1.5\pi)$

$f(z_3) = \sin(0.7\pi)$ $\quad$ $f(z_8) = \sin(1.7\pi)$

$f(z_4) = \sin(0.9\pi)$ $\quad$ $f(z_9) = \sin(1.9\pi)$

5) Calculate the Approximate Integral.

midpoint formula:

$$I_N \approx h \sum_{i=0}^{N-1} f(z_i)$$

$h = 0.2\pi$

$I_N = (0.2\pi)(\sin 0.1\pi + \sin 0.3\pi + \sin 0.5\pi + \sin 0.7\pi + \sin 0.9\pi + \sin 1.1\pi + \sin 1.3\pi + \sin 1.5\pi + \sin 1.7\pi + \sin 1.9\pi)$

$N = 1$

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

$f(x) = x^2$ $\quad$ $[2, 5]$

1) $h = \frac{5-2}{1} = 3$

2) $x_i = a + i \cdot h$ $\quad$ $i = 0, 1$

$x_0 = 2 + 0(3) = 2$

$x_1 = 2 + 1(3) = 5$

3) $z_0 = \frac{x_0 + x_1}{2} = \frac{2+5}{2} = 3.5$

4) $f(z_0) = \sin(3.5)\ (3.5)^2 = 12.25$

5) $h(12.25) = (3)(12.25) = 36.75$

$N = 2$

1) $h = \frac{b-a}{N} = \frac{5-2}{2} = \frac{3}{2}$

2) $x_i = a + ih$ for $i = 0, 1, 2$

$x_0 = 2 + 0(3/2) = 2$

$x_1 = 2 + 1(3/2) = 7/2$

$x_2 = 2 + 2(3/2) = 5$

3) $z_0 = \frac{2 + \frac{7}{2}}{2} = 11/4$

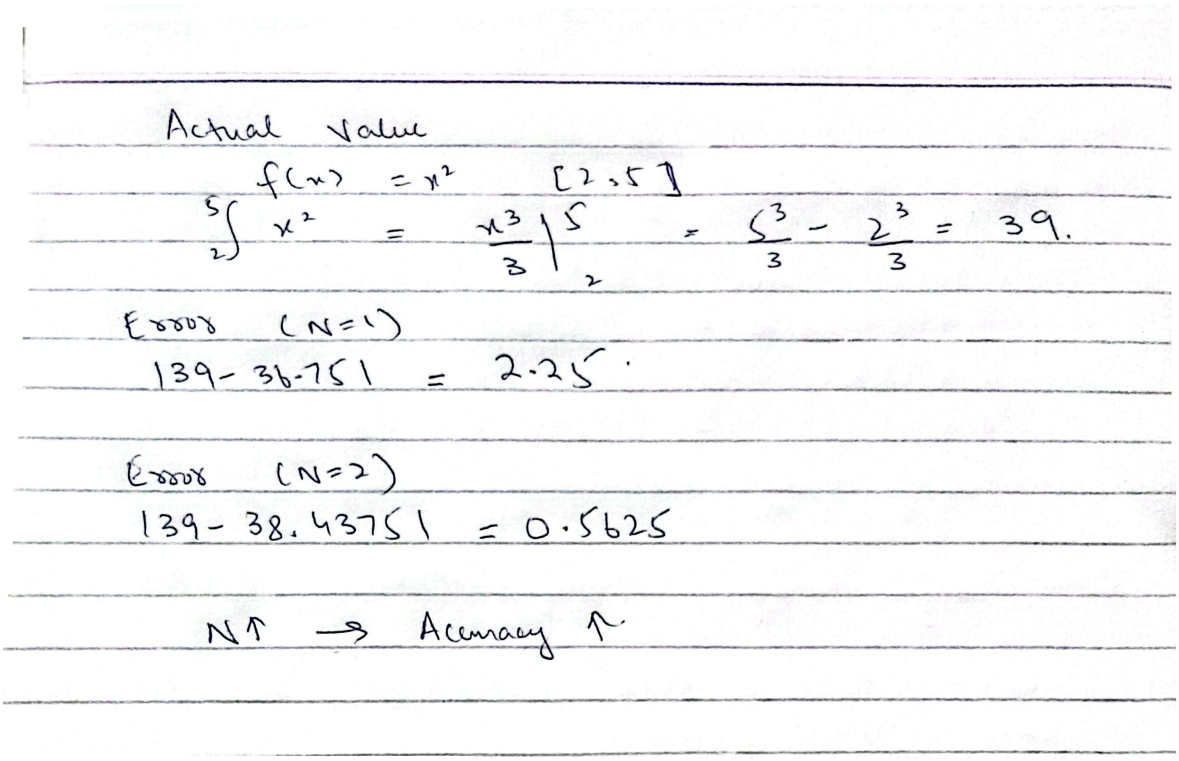
$z_1 = \frac{\frac{7}{2} + 5}{2} = 17/4$

4) $f(z_0) = (11/4)^2 = 121/16$

$f(z_1) = (17/4)^2 = 289/16$

5) $(3/2)\left(121/16 + 289/16\right)$

$= 38.4375$



Application in Classical Mechanics

force f at a location x

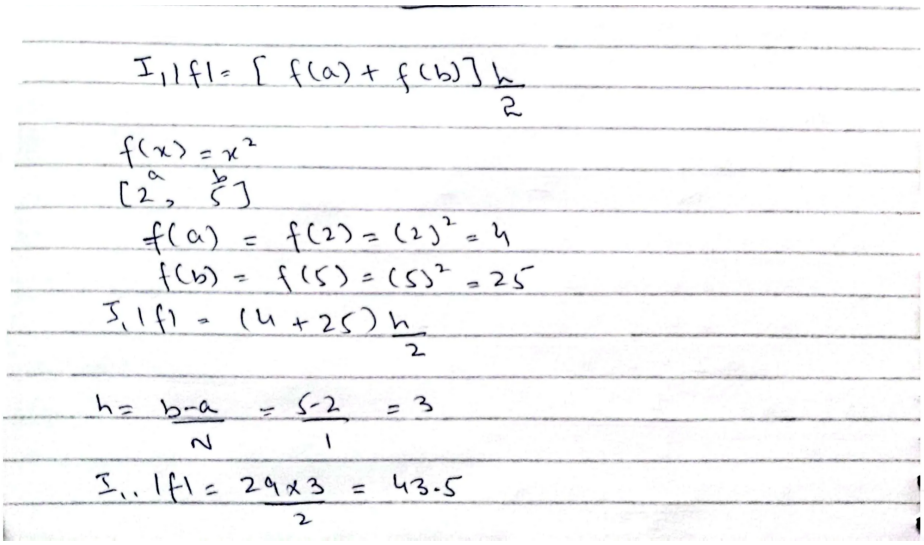
x_i	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4
$f(x_i)$	0.0	0.45	1.45	2.3	3.1	3.1	3.1	2.5	1.1	1.1	1.1	0.8	0.6	0.3	0.0

compute work done by the particular force using the trapezoidal rule

```
from scipy.integrate import trapezoid
import numpy as np

x = np.arange(0.0,1.5,0.1)
y = np.array([0.0, 0.45, 1.45, 2.3, 3.1, 3.1, 3.1,
              2.5, 1.1, 1.1, 1.1, 0.8, 0.6, 0.3, 0.0])
print( trapezoid(y,x) )
#2.1
```

Newton-Cotes Quadrature Rules

Trapezoidal Rule



Composite Trapezoidal Rule

1. **Calculate** $h = \frac{b-a}{N}$
2. **List the** x_i **Values**:
   - Compute each xi using $x_i = a + i \cdot h$

     $i = 0, 1, \ldots, N$
3. **Evaluate** $f(x_i)$ **at Each Point**.
4. **Plug Into the Composite Formula**:
   - Compute:

$$I_N[f] \approx \frac{h}{2} \left[f(x_0) + 2f(x_1) + \cdots + 2f(x_{N-1}) + f(x_N) \right].$$

5. **Calculate the Final Result**.

$N = 2$

1) $h = \frac{b-a}{N} = \frac{5-2}{2} = \frac{3}{2}$

2) Calc x_i values

$x_i = a + ih \qquad i = 0, 1, 2.$

$x_0 = 2 + 0(3/2) = 2$

$x_1 = 2 + 1(3/2) = 7/2$

$x_2 = 2 + 2(3/2) = 5$

3) Evaluate $f(x_i)$

$f(x_0) = f(2) = (2)^2 = 4$

$f(7/2) = (7/2)^2 = 49/4 = 7\frac{1}{2}\ 49/4$

$f(5) = (5)^2 = 25$

4) $I_N[f] = \frac{h}{2}[f(x_0) + 2f(x_1) + \cdots f(x_2)]$

$= \frac{3}{4}\left[4 + 2\left(\frac{49}{4}\right) + 25\right]$

$= \frac{3}{4}[4 + 24.5 + 25]$

$= \frac{3}{4}[4 + 7 + 25]$

$= \frac{3}{4}[37] = \frac{3}{4} \times 18 =$

$= 40.125.$

Error $= |39 - 40.125| = 1.125.$

$N \uparrow \rightarrow$ approx $\uparrow$

```
def trapezoidal(f, a, b, N):
    # Composite trapezoidal rule with N subintervals
    x = np.linspace(a, b, N+1)
    y = f(x)
    h = (b - a)/N
    # First add the internal nodes
```

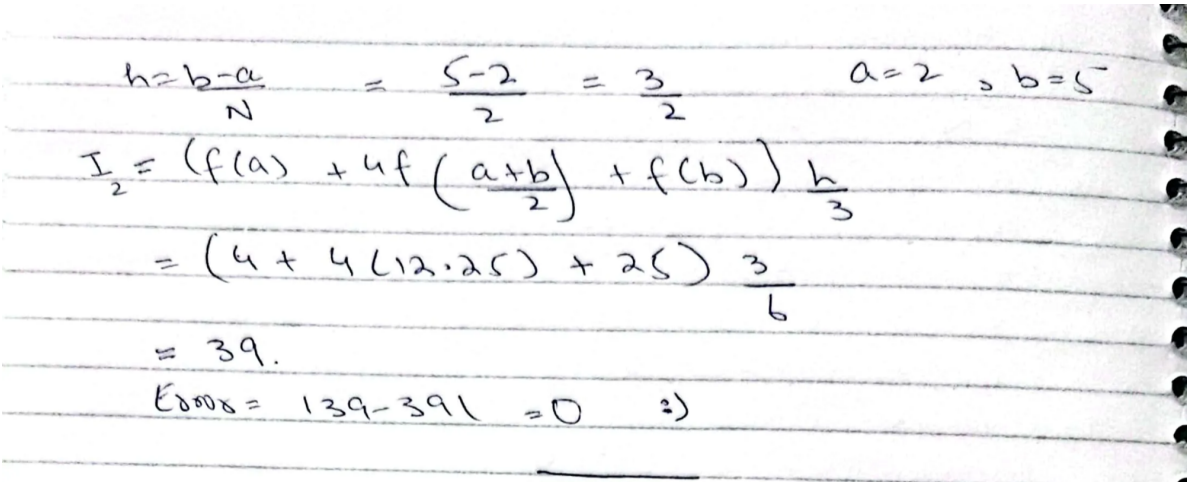
```
    sum = 0.0
    for i in range(1,N):
        sum += 2.0*y[i]
    # Add the two boundary nodes
    sum = 0.5*h*(f(a) + sum + f(b))
    return sum

f = lambda x: np.sin(x)

trapezoidal(f, 0.0, np.pi/2.0, 10)
```

```
from scipy.integrate import trapezoid
f = lambda x: np.sin(x)
x = np.linspace(0.0, np.pi/2.0, 11)
y = f(x)
print( trapezoid(y,x) )
#0.9979429863543573
```

Simpson’s Rule

Simpson's 1/3 rule uses N=2 and thus quadratic polynomials:

$$I_2 = \sum_{i=0}^{2} w_i f(x_i) = \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right] \frac{h}{3}$$



Composite Simpson's Rule

1. **Calculate** $h = \frac{b-a}{N}$.
2. **List the Points xi**:
   - Compute $x_i = a + i \cdot h$

   $i = 0, 1, \ldots, N$
3. **Evaluate** $f(x_i)$
   - Compute $f(x_i)$ for each xi.
4. **Apply the Composite Simpson's Rule Formula**:

$$I_N[f] \approx \frac{h}{3} \left[f(x_0) + 4f(x_1) + 2f(x_2) + \cdots + 2f(x_{N-2}) + 4f(x_{N-1}) + f(x_N) \right]$$

5. **Compute the Final Approximation**:
   - Multiply each $f(x_i)$ by its respective coefficient.
   - Sum up all the terms.
   - Multiply by h/3 to get the final approximation.

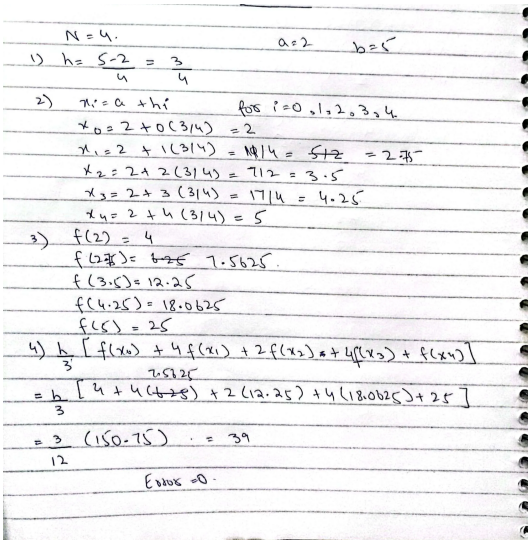
Coefficients Explanation

1. **Endpoints**
   - The function values at the endpoints $f(x_0)$ and $f(x_N)$ are multiplied by **1**. These are the first and last terms.
2. **Odd Index Points**

- The function values at these points are multiplied by **4**. These are the terms with odd indices, starting from x_1 and ending at x_{N-1}.

3. **Even Index Points**
   - The function values at these points are multiplied by **2**. These are the terms with even indices, starting from x_2 and ending at x_{N-2}.

- **Simpson's rule is more accurate** than the trapezoidal rule for functions that are well approximated by parabolas.
- **N must be even**, so if you are given an odd number of subintervals, you will need to adjust N accordingly.




```
def Simpson(f, a, b, N):
# Composite Simpson rule with N subintervals
    h = (b - a)/float(N)
    x=np.linspace(a, b, N + 1)
    s = f(x)
    s[1:N]=2*s[1:N]
    s[1:N:2]=2*s[1:N:2]
    return h/3.0*np.sum(s)

f = lambda x: np.sin(x)
```

```
Simpson(f, 0.0, np.pi/2.0, 10)
```

1.0000033922209004

```
from scipy.integrate import simpson
f = lambda x: np.sin(x)
x = np.linspace(0.0, np.pi/2.0, 11)
y = f(x)
print( simpson(y,x) )
#1.0000033922209006
```

Gauss-Legendre Quadrature Rules

table will be given

N	x_i	w_i
1	0	2
2	$\pm\sqrt{\frac{1}{3}}$	1
3	0	$\frac{8}{9}$
	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$
5	0	$\frac{128}{225}$
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}$
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	$\frac{322-13\sqrt{70}}{900}$

$f(x) = x^2$ interval $[2, 5]$

1) Transform the integral from the interval $[2, 5]$ to $[-1, 1]$ as Gaussian Legendre is defined over $[-1, 1]$.

$$\int_2^5 x^2 dx = \int_{-1}^{1} f\left(\frac{3}{2}t + \frac{7}{2}\right)$$

$$x = \frac{b-a}{2} \cdot t + \frac{a+b}{2}$$

$a = 2$, $b = 5$

$$x = \frac{5-2}{2} t + \frac{2+5}{2} = \frac{3}{2} t + \frac{7}{2}$$

Change Integral Bounds

$$\frac{dx}{dt} = \frac{3}{2} + 0 = \frac{3}{2} dt$$

$$dx = \frac{3}{2} dt$$

$$\int_2^5 x^2 dx = \int_{-1}^{1} \left(\frac{3}{2}t + \frac{7}{2}\right)^2 \cdot \frac{3}{2} dt$$

$$= \frac{3}{2} \int_{-1}^{1} \left(\frac{3}{2}t + \frac{7}{2}\right) dt$$

2) For a 2 point Gaussian-Legendre over $[-1, 1]$, the nodes and weights are:

$$t_1 = \frac{-1}{\sqrt{3}}$$

$$t_2 = \frac{1}{\sqrt{3}}$$

$w_1 = 1$

$w_2 = 1$

3) Evaluate ~~f(x)~~ two points at the transformed nodes for t_1 and t_2.

For $t_1 = \frac{-1}{\sqrt{3}}$

$$x_1 = \frac{3}{2}\left(\frac{-1}{\sqrt{3}}\right) + \frac{7}{2} = \frac{7-\sqrt{3}}{2} = 2.633947$$

For $t_2 = \frac{1}{\sqrt{3}}$

$$x_2 = \frac{3}{2}\left(\frac{1}{\sqrt{3}}\right) + \frac{7}{2} = \frac{7+\sqrt{3}}{2} = 4.366$$

4) Compute $f(x_1)$ and $f(x_2)$

$f(x_1) = (x_1)^2 = 6.938$

$f(x_2) = (x_2)^2 = 19.0662$

5) Final Approximation

$$\int_2^5 f(x)\,dx = \frac{3}{2}\left[w_1 \cdot f(x_1) + w_2 \cdot f(x_2)\right]$$

$$= \frac{3}{2}\left[(1) f(x_1) + (1) f(x_2)\right]$$

$$= 38.9995$$

$$= 39$$

```
def GaussLegendre3(f, a, b):
    # Implementation of Gauss-Legendre quadrature with
    # 3 nodes in general interval $[a,b]$
    N = 2
    # define quadrature weights and nodes
    w = np.array([5.0/9.0, 8.0/9.0, 5.0/9.0])
    z = np.array([-np.sqrt(3.0/5.0), 0.0, np.sqrt(3.0/5.0)])
    # implement formula (\ref{eq:gengauleg})
    c1 = (b-a)/2.0
    c2 = (a+b)/2.0
    s = c1*np.inner(w, f( c1*z + c2 ))
    return s

 f = lambda x: 0.5*np.exp(x/2.0)*np.cos(x/2.0)
GaussLegendre3(f, -2.0, 2.0)
```

```
from scipy.integrate import fixed_quad
f = lambda x: 0.5*np.exp(x/2.0)*np.cos(x/2.0)
print( fixed_quad(f,-2.0,2.0,n=3) )
#(1.9333904692642976, None)
```

```
from scipy.integrate import quadrature
f = lambda x: 0.5*np.exp(x/2.0)*np.cos(x/2.0)
print( quadrature(f,-2.0,2.0,tol=1.e-15,maxiter=100) )
#(1.9334214962992706, 9.692340263711685e-10)
```