$ Supplement Note 8: Numerical Integration: The Trapezoidal Rule and The Simpson Rule

The definite integral $\int_a^b f(x)dx$ is defined in section 4.1, p.4-1 of **Lecture Note** as the *Riemann sum*:

$$\int_a^b f(x)dx \equiv \lim_{\max|\Delta x_i|\to 0} \sum_i^n f_i \Delta x_i; \quad \text{with} \quad \sum_i \Delta x_i = b - a \tag{1}$$

To be specific, the sum is taken over the n sub intervals $\{[x_i, x_{i+1}]\}_{i=1,\ldots,n}$ with $\Delta x_i \equiv x_{i+1} - x_i$, constructed from a series of numbers $\{x_0 = a, x_1, \cdots, x_{n+1} = b\}$. The quantities f_i are the values of the function $f(x)$ evaluated at *any point* x inside each sub intervals $[x_i, x_{i+1}]$. In the limit $\max|\Delta x_i| \to 0$, the sum of all different choices of f_i will converge to the same limit $I = \int_a^b f(x)dx$. For $f(x) > 0$, the Riemann sum (1) is the area surrounded by $x = a$, $x = b$, $y = f(x)$ and the x-axis.

The simplest approach (which will be adopt here) to evaluate the integral is to use equal intervals: $\Delta x_i \equiv x_{i+1} - x_i = h = \frac{b-a}{n}$. For example , the sum in (1) may be approximated by taking f_i either all at the left ends of each interval (I_1) or all at the right ends of each interval (I_2):

$$\int_a^b f(x)dx \approx I_1 \equiv \sum_i f_i \, (x_{i+1} - x_i) = h \sum_{i=1}^n f_i \tag{2a}$$

$$\int_a^b f(x)dx \approx I_2 \equiv \sum_i f_{i+1} \, (x_{i+1} - x_i) = h \sum_{i=1}^n f_{i+1} \tag{2b}$$

Trapezoidal Rule

In the *trapezoidal rule*, linear functions are used to approximate the function $f(x)$ in each sub interval $[x_i, x_{i+1}]$ so that the integrals in each sub interval are approximated by the areas of the trapezoids consisted of the x-axis, the lines connecting the two end points (x_i, f_i) and (x_{i+1}, f_{i+1}), and the two vertical end lines

$x = x_i, x = x_{i-1}$:

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{(f_{i+1} + f_i)}{2}(x_{i+1} - x_i) = \frac{h}{2}\Big(f_{i+1} + f_i\Big) \tag{3}$$

This leads to the *trapezoidal rule*, which is equal to the averaged value of I_1 and I_2 of (2):

$$\int_a^b f(x)dx \approx h\Big(\frac{f_0}{2} + f_1 + f_2 + \cdots + f_{n-1} + \frac{f_n}{2}\Big) = \frac{I_1 + I_2}{2} \tag{4}$$

Simpson Rule

In Simpson rule, the number n of sub intervals has to be an even number. The function f is approximated by a *piece-wise quadratic function* in each consecutive sub intervals, $[x_i, x_{i+1}], [x_{i+1}, x_{i+2}], i = 1, 3, 5, \ldots$, i.e. for $x \in [x_i, x_{i+2}], f(x) \approx g(t) = a_0 + a_1 t + a_2 t^2, t \equiv x - x_{i+1}$.

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx I_{i;\, i+2} \equiv \int_{-h}^{h} (a_0 + a_1 t + a_2 t^2)\, dt = 2ha_0 + \frac{2h^3}{3}a_2 \tag{5}$$

The coefficients a_0, a_1, a_2, can be obtained from the quadratic function representations for the approximations:

$$f_0 \equiv f(x_i) = g(-h) = a_0 - a_1 h + a_2 \frac{h^2}{2} \tag{6a}$$

$$f_1 \equiv f(x_{i+1}) = g(0) = a_0 \tag{6b}$$

$$f_2 \equiv f(x_{i+2}) = g(h) = a_0 + a_1 h + a_2 \frac{h^2}{2} \tag{6c}$$

This leads to

$$a_1 = \frac{f_2 - f_0}{h}; \quad a_2 = \frac{f_2 + f_0 - 2f_1}{h^2} \tag{7}$$

and Eq.(5) becomes:

$$I_{i;\, i+2} = \frac{h}{3}(f_0 + 4f_1 + f_2) = \frac{h}{3}\Big(f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\Big) \tag{8}$$

This leads to *Simpson's rule* for approximating the integral I

$$I \equiv \int_a^b f(x)dx \approx \frac{h}{3}\Big(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_1)\Big) \tag{9}$$

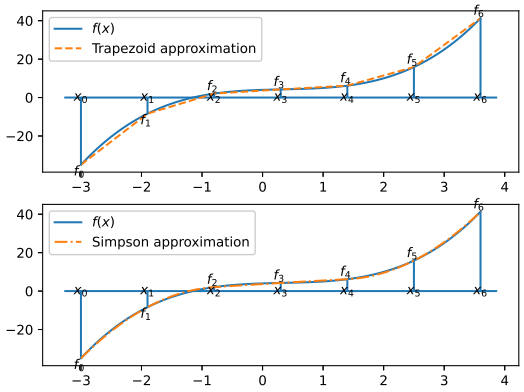

Figure 1: Trapezoid Rule and Simpson Rule

Examples:

In the first example in the python code *TrapezoidSimpson.py*, the integration of the cubic equation with an interval $h = 1.1$ as shown in Figure 1 is evaluated explicitly:

$$I = \int_{-3}^{3.6} (x - x^2 + 4 + x^3)\, dx = 25.5684 \tag{10}$$

The approximations I_1 and I_2 in Eq.(2) yield the results $I_1 = -16.5275$ and $I_2 = 67.3981$ respectively. The result from trapezoidal rule is given by 25.43530 which is equal to $\frac{1}{2}(I_1 + I_2)$ as expected. The value from Simpson rule is 25.5684 and is identical with the exact result (10). This agreement with the rather large value of h is related to the fact that the integrand is a cubic function. It is instructive to check this for different cubic functions with different values of h.

In the second example the following integration is evaluated:

$$I = \int_0^2 x^2 e^{-x^3} dx = \int_0^8 e^{-t} \frac{dt}{3} = \frac{1}{3}(1 - e^{-8}) = 0.3332215 \tag{11}$$

For $h = 0.2$, the approximations in Eq.(2) yield $I_1 = 0.333309$ and $I_2 = 0.333041$ respectively. The trapezoidal rule (4) yields a value $I = 0.333175 = \frac{1}{2}(I_1 + I_2)$. The value from Simpson rule is 0.333207. The magnitude of the error 0.000015 is within 5-th order of the step size, $h^5 = 0.00032$.

```
# TrapezoidSimpson.py
import numpy as np
# example 1: y= x - x**2+ 4+ x**3
x0, h, nTr, ng= -3,1.1, 6,20
xmax = x0+ nTr*h
x= np.linspace(x0,xmax,nTr+1)
y= x - x**2+ 4+ x**3
I1= h*np.sum(y[1:])
I2= h*np.sum(y[:-1])
Str = h*(np.sum(y[1:-1])+0.5*(y[0]+y[-1]))  # Trapezoidal rule
print(y)
y[1::2]= 4* y[1::2]      # Simpson rule
y1=  y[:-1]
y1[2::2]= 2* y1[2::2]    # Simpson rule
print(y)
Simp= h* np.sum(y)/3     # result from Simpson rule
print('I1,I2,0.5*(I1+I2)=',I1,I2,0.5*(I1+I2))
print('STr=',Str)
print('Simp=',Simp)
yy= xmax**2/2 - xmax**3/3+ 4*xmax+ xmax**4/4-(x0**2/2 - x0**3/3+ 4*x0+ x0**4/4)
print(yy)                # exact result
############
# example 2: y= x^2* exp(- x^3)
x0, h, nTr, ng= 0,0.2, 10,20
xmax = x0+ nTr*h
x= np.linspace(x0,xmax,nTr+1)
y= x**2* np.exp(- x**3)
I1= h*np.sum(y[1:])
I2= h*np.sum(y[:-1])    # Simpson rule
Str = h*(np.sum(y[1:-1])+0.5*(y[0]+y[-1]))  # Trapezoid rule
print(y)
y[1::2]= 4* y[1::2]    # Simpson rule
y1=  y[:-1]
```

```
y1[2::2]= 2* y1[2::2]  # Simpson rule
print(y)
Simp= h* np.sum(y)/3   # result from Simpson rule
print('I1,I2,0.5*(I1+I2)=',I1,I2,0.5*(I1+I2))
print('STr=',Str)
print('Simp=',Simp)
yy= (1-np.exp(-8))/3   #exact result
print(yy)
```

Finally, Figure 1 is produced by the following python code.

```
# DrawTrapezoidSimpson.py
import numpy as np
import matplotlib.pyplot as plt
x0, h, nTr, ng= -3,1.1, 6,20
nt, nSim = nTr*ng, nTr//2    #integer division
xmax = x0+ nTr*h
x= np.linspace(x0,xmax,nt+1)
xf =x[-1]
plt.figure()
ax1= plt.subplot(2,1,1)
#plt.title('Trapezoid approximation')
plt.hlines(0,x0-h/4,xf+h/4)
y= x - x**2+ 4+ x**3
ax1.plot(x, y, label=f'$f(x)$')
X = x[::ng]
Y = y[::ng]
#ax1.axis('off')
ax1.plot(X,Y, '--', label='Trapezoid approximation')
ax1.legend()
ax2= plt.subplot(2,1,2)
ax2.plot(x, y, label=f'$f(x)$')
for i,tf in enumerate(zip(X,Y)):
    t,f = tf
    ax2.vlines(t,0,f)
```

```
    ax1.vlines(t,0,f)
    ax2.text(t-0.12,-2.,f'$x_{{{i}}}$')
    ax1.text(t-0.12,-2.,f'$x_{{{i}}}$')
    tx = f+1.5 if f>0 else f-5.8
    ax1.text(t-0.12,tx,f'$f_{{{i}}}$')
    ax2.text(t-0.12,tx,f'$f_{{{i}}}$')
plt.hlines(0,x0-h/4,xf+h/4)
nx, Nx = 0,0
for i in range(nSim):
    n2= nx+2*ng
    xt= x[nx:n2+1]
    y0, y1, y2 = Y[Nx], Y[Nx+1], Y[Nx+2]
    x0, x1, x2 = X[Nx], X[Nx+1], X[Nx+2]
    a0,a1,a2 = y1, (y2-y0)/(2*h), (y2+y0-2*y1)/(2*h**2)
    z1 = a0+ a1*(xt-x1) + a2*(xt-x1)**2
    if nx == 0: z = z1
    else: z = np.append(z,z1[1:])
    nx = n2
    Nx +=2
ax2.plot(x, z, linestyle ='dashdot', label='Simpson approximation')
ax2.legend()
#plt.axis('off')
fig = plt.gcf()
fig.savefig('TrapezoidSimpson.eps', format='eps')
plt.show()
```