$ Supplement Note 11: Numerical Integration: The Trapezoidal and Simpson Rules

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 subintervals $\{[x_i, x_{i+1}]\}_{i=1,\ldots,n}$ 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 subintervals $[x_i, x_{i+1}]$. In the limit $\max|\Delta x_i| \to 0$, the sum of all different choices of f_i will converge to the same limit $\int_a^b f(x)dx$. 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 subintervals: $\Delta x_i \equiv x_{i+1} - x_i = h = \frac{b-a}{n}$. 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 at the right ends of each interval (I_2):

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

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

Trapezoidal Rule

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

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

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

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

$$\int_a^b f(x)dx \approx h\Big[\frac{f(x_0)}{2} + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{f(x_n)}{2}\Big] = \frac{I_1 + I_2}{2} \tag{4}$$

Simpson Rules

In Simpson rule, the number n of subintervals has to be an even number. The function f is approximated by a *piecewise quadratic functions* over each consecutive subintervals, $[x_i, x_{i+1}], [x_{i+1}, x_{i+2}], i = 1, 3, 5, \ldots$, 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 value $I_{i;\, i+2}$ may readily be obtained from the representation:

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

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

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

Hence we obtain

$$I_{i;\, i+2} = \frac{h}{3}(f_0 + 4f_1 + f_2) = \frac{h}{3}\Big(f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\Big) \tag{7}$$

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

$$\begin{aligned} I \equiv \int_a^b f(x)dx \approx \frac{h}{3}\Big(&f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots \\ &+ 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_1)\Big) \end{aligned} \tag{8}$$

As a first example, let us consider the following integration (with a substitution $t = x^2$ is used):

$$I = \int_{0.4}^{1.2} xe^{-x^2}dx = \int_{0.16}^{1.44} \frac{e^{-t}dt}{2} = \frac{1}{2}(e^{-0.16} - e^{-1.44}) = 0.307608015 \tag{9}$$

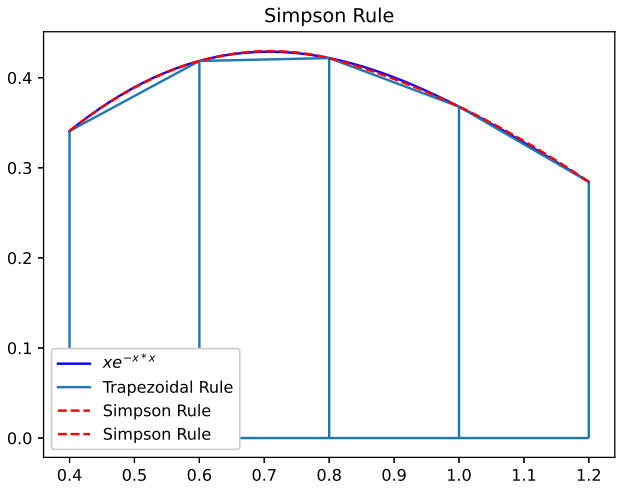

Figure 1: Trapezoidal and Simpson Rule

In the python code *NumericalInt.py* a value $h = 0.2$ is used for the length of subintervals. The approximations are $I_1 = 0.2985$ and $I_2 = 0.3098$ respectively, and the Trapezoidal Rule (4) yields a value $I = 0.30418$. On the other hand, the result from Simpson Rule is 0.307652. which should be compared with the exact value of 0.307608015. The results from Simpson Rule with smaller values of $h = 0.1, 0.05$ and 0.025 are also calculated. They are given by 0.3076107, 0.3076081838 and 0.3076080257 respectively. Many more sophisticate procedures are available in packages such as *scipy*, but Simpson Rule are capable of dealing most numeric integration.

```
# NumericalInt.py
# numerical integration: Trapezoidal Rule and Simpson Rule
import numpy as np
import matplotlib.pyplot as plt
###3# plotting int_0.4^1.2 xe^{-x^2} dx
plt.title('Numerical Integration')
tx = np.linspace(0.4,1.2,51)
tx2 = tx**2
```

```
ftx= tx*np.exp(-tx2)
plt.plot(tx,ftx,'b',label='$ xe^{-x*x}$')
########
x = np.linspace(0.4, 1.2, 5)
h = x[1]- x[0]
print('h=', h)
xx = x*x
f= x*np.exp(-xx)
print('x=', x)
print('f=', f)
sum1 = h*np.sum(f[1:])
sum2 = h*np.sum(f[:-1])
value = 0.5*(np.exp(-0.16)- np.exp(-1.44))
print('value, sum1, sum2=',value, sum1, sum2)
trape = h*(0.5*f[0]+ np.sum(f[1:-1])+0.5* f[-1])
print('From Trapezoidal Rule:', trape)
plt.plot(x,f,label='Trapezoidal Rule')
for t,ft in zip(x, f):
    plt.vlines(t,0,ft)
plt.hlines(0,0.4,1.2)
#### simpson
f2 = f[::2]
simp = (f[0]+ 4* np.sum(f[1::2])+ 2* np.sum(f2[1:-1])+  f[-1])*h/3
print('From Simpson Rule:', simp)
#### Further more accurate results from Simpson Rule
n = 4
print('Further Results from Simpson Rules with smaller value of h')
for i in range(3):
    ng = 2* n +1
    n = n*2 # prepare for next loop
    x1 = np.linspace(0.4, 1.2, ng)
    h1 = x1[1]- x1[0]
    xx1 = x1*x1
```

```
    f1= x1*np.exp(-xx1)
    f21 = f1[::2]
    simp1 = (f1[0]+ 4* np.sum(f1[1::2])+ 2* np.sum(f21[1:-1])+  f1[-1])*h1/3
    print('h=',h1, 'From Finer Simpson Rule:', simp1)
#### find the coefficients for quadratic fit
xa, fa = x[2:] , f[2:]
ca =np.polyfit(xa, fa,2)
xb, fb=  x[:3], f[:3]
cb =np.polyfit(xb, fb,2)
#plot quadratic approximation !!for Simpson Rule
txa, txb = tx[25:], tx[:26]
fca= np.polyval(ca,txa) #result from quadratic fit
fcb= np.polyval(cb,txb) #result from quadratic fit
print(fca)
plt.plot(txa,fca,'r--',label='Simpson Rule')
plt.plot(txb,fcb,'r--',label='Simpson Rule')
plt.legend()
plt.title('Simpson Rule')
plt.legend()
fig = plt.gcf()
fig.savefig('TrapezoidalSimpson.eps', format='eps')
plt.show()
```