CEE 451

Coding Problem Number 2: Trapezoidal Rule

We wish to compute numerically the integral I of some function f(x) from x = x_A to x = x_B; i.e.

$$I = \int_{x_A}^{x_B} f(x)dx$$

One way to do this using the trapezoidal rule. This will be explained more in class. We divide the interval $x_A \le x \le x_B$ into N intervals, each with length Δx, where

$$\Delta x = \frac{x_B - x_A}{N}$$

We then discretize this interval as

$$x_i = x_A + (i-1)\Delta x \quad , \quad i = 1..N+1$$

According to the trapezoidal rule, the integral between points x_i and x_{i+1} is approximated as

$$\int_{x_i}^{x_{i+1}} f(x)dx \cong \frac{1}{2}(f_i + f_{i+1})\Delta x$$

where

$$f_i = f(x_i)$$

The value of I is then

$$I = \sum_{i=1}^{N}\left[\int_{x_i}^{x_{i+1}} f(x)dx\right] \cong \sum_{i=1}^{N}\left\{\frac{1}{2}\left[f_i + f_{i+1}\right]\right\}\Delta x =$$
$$\left(\frac{1}{2}f_1 + f_2 + f_3 + ... + f_N + \frac{1}{2}f_{N+1}\right)\Delta x$$

Problem:

Write a program to compute the integral from x_A to x_B of the functions

$$f(x) = e^{-\lambda x} \qquad (1)$$

where λ is a user-specified value, and

$$f(x) = e^{\sin(x)+\sqrt{x}} \qquad (2)$$

Implement your code for x_A = 0 and x_B = 4 (and λ = 1). Show results for N = 2, 3… up to some value of N_δ beyond which the value of I changes within only some specified tolerance; i.e. if I_N is the estimate of I obtained using N intervals, then I_{N_δ} should be such that the relative fractional difference ε satisfies the condition

$$\varepsilon = \left|\frac{\left(I_{N_\delta+1} - I_{N_\delta}\right)}{\frac{1}{2}\left(I_{N_\delta+1} + I_{N_\delta}\right)}\right| < \delta \qquad (3)$$

where δ is a user-specified tolerance. That is, find a value of N that is large enough so that value of I essentially stops changing for larger N. Check your code against the analytical value for the integral of (1). Can you find an analytical value for the integral of (2)?

Please execute this code using arrays. In VBA, for example, $x_i \rightarrow$ x(i) can be expressed as

```
    For i = 1 to N+1
        x(i) = xA + (i – 1)* Dx
    Next i
```

You can e.g. define a Function statement with the name “fff” (or whatever you want) that looks like

```
    Function fff(dumx)
        fff = exp(-lamda*dumx)
    End Function
```

and then compute the values $f_i \rightarrow$ f(i) as

```
    f(i) = fff(x(i))
```

(You will want to use a For loop to compute all the values f(i)).
Then the essential part of the programming will look something like

```
    Intit = 0
    For i = 1 to N
        Intit = Intit + 0.5*(f(i)+ f(i+1)*Dx
    Next i
```

You will specify x_A, x_B and N as input parameters. You will need to have as output the value of Intit. You may want to write the code to directly compute values of I using N = 2, 3, 4, 5, etc. (as high as you want to go), and checking how much more accurate the solution gets as N increases.

Your solution will consist of a) code text + printout of any GUI’s if used (e.g. Excel spreadsheet as input/output), b) a specification of your choice of δ, and c) output for the two integrals for all value of N up to $N_{\delta+1}$.

NOTE. The FIRST STEP is to write a working code that does nothing more than compute the integral for a specified form for f(x) (you will also have to code in the analytical form for f’(x), specified limits x_A and x_B and a specified value for N. You should then integrate this code into a larger code that calculates the integral for a N = 2, 3, 4… up to whatever it takes to realize a value of the integral that no longer significantly changes with increasing N.