MIE334: Numerical Methods **Assignment 4**

Due: **April 13th**, **11:59 pm**, 2021

Differential equation and initial conditions:

At i=0,

We use a step size of . At i=1,

Repeat for all further i’s:

|  |  |  |
| --- | --- | --- |
| **i** | **(m)** | **(˚C)** |
| 0 | 0 | 24 |
| 1 | 4 | 35.1326 |
| 2 | 8 | 44.9742 |
| 3 | 12 | 53.6746 |
| 4 | 16 | 61.3660 |
| 5 | 20 | 68.1655 |

In conclusion, the water temperature at the outlet was found to be 68.166 ˚C.

**(2)**

Analytical Solution:

Trapezoidal Rule Solution:

Number of function evaluations needed:

**Calculations:**

|  |  |  |
| --- | --- | --- |
| Number of segments | I\_trap | Error (%) |
| 1 | 0 | 100 |
| 2 | -4.7123 | 335.62 |
| 3 |  | 100 |
| 4 | 0.9760 | 51.20 |
| 5 | 1.3695 | 31.52 |
| 6 | 1.5708 | 21.46 |
| 7 | 1.6883 | 15.58 |
| 8 | 1.7631 | 11.84 |
| 9 | 1.8138 | 9.31 |

Gauss Quadrature Solution:

For , .

For ,

For ,

For ,

Number of function evaluations needed:

**Calculations:**

|  |  |  |
| --- | --- | --- |
| Number of Gauss points | I\_gauss | Error (%) |
| 1 | -9.4248 | 571.24 |
| 2 | 8.6022 | -330.11 |
| 3 | 0.3844 | 80.78 |
| 4 | **2.1880** | -9.4 |

**Comments:**

Using trapezoidal rule required more function evaluations than using Gaussian quadrature to obtain an integral with . Using trapezoidal rule, a 9 segment approximation must be used, resulting in 10 function evaluations. However, using Gaussian quadrature only needed 4 Gaussian points, resulting in 4 function evaluations. This aligns with our expectations since Gaussian quadrature generally converges faster in problems where we know the function being integrated. Thus, using Gaussian quadrature is faster and more efficient.

**Trapezoidal Solution Code :**

|  |
| --- |
| integrals=[];  n=1;  e\_t = 1;  e\_t\_array = [];    % Loop through values of n, the number of segments  while e\_t > 0.1  % Create n+1 equally spaced points which define n segments.  x=linspace(0,3\*pi,n+1);  % Evaluate f(x) at each x point.  y=sin(x)    % Set initial sum value  sum = y(1);    % If only 1 segment exists, do not sum over intermediate terms (since they do not exist)  if n >=2    % Iterates over the intermediate points in x to sum them.  % Intermediate points are found at the beginning of every nth  % segment, starting at n=2.  for j=2:n  sum = sum + 2\*y(j);  end  end  % Add the final value in the integral.  sum = sum + y(end);    % Calculate final integral  int = (3\*pi/n)\*sum/2;    % Calculate true error from int  e\_t = abs((2-int))/2;    % Concatenate int and e\_t to arrays  integrals = [integrals,int];  e\_t\_array = [e\_t\_array,e\_t];    % Increment n  n = n+1;  end    % decrement n by 1 to obtain true number of segments used.  num\_segs = n-1; |

**Gauss Solution Code:**

|  |
| --- |
| %Uses 1D gaussian quadrature method to find integral of sinx  integrals=[];  n=1;  e\_t = 1;  e\_t\_array = [];    b= 3\*pi;  a= 0;    % Create stacks for weights and gauss points.  coefs = [2,1,1,5/9,8/9,5/9,(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36,(18-sqrt(30))/36]  points = [0,-1/sqrt(3),1/sqrt(3),-sqrt(3/5),0,sqrt(3/5),0.339981,-0.339981,0.861136,-0.861136]    % Loop through values of n, the number of segments  while e\_t > 0.1  %Gather relevant points and weights from stack  cur\_coefs = coefs(1:n)  cur\_points = points(1:n)    % Pop used coefficients from the front of stack  coefs = coefs(n+1:end)  points = points(n+1:end)    % Calculate integral  sum = 0;    for i=1:n  % Find transformed x  x=(b+a+(b-a)\*cur\_points(i))/2  % Find sin(x) and Multiply by weight  sum = sum + cur\_coefs(i)\*(sin(x))  end    % Find final integral  int = ((b-a)/2)\*sum;    % Calculate true error from int  e\_t = abs((2-int))/2;    % Concatenate int and e\_t to arrays  integrals = [integrals,int];  e\_t\_array = [e\_t\_array,e\_t];    % Increment n  n = n+1;  end |