Jared Rivera

804603106

CEE 103 HW#4

%HW4 P1a

%Approximates Pi using Trapzoidal rule

clear all; close all; clc

b=1;

a=-1;

n=64;

h=(b-a)/n;

T=zeros(1,n+1);

x=linspace(-1,1,n+1);

f=sqrt(1-x.^2);

T(1)=(h/2)\*f(1);

for i=2:n

T(i)=h\*f(i);

end

T(n+1)=(h/2)\*f(n+1);

piapprox=2\*sum(T);

%HW4 P1b

%Approximates Pi using Simpsons rule

clear all; close all; clc

b=1;

a=-1;

n=64;

h=(b-a)/n;

T=zeros(1,n+1);

x=linspace(-1,1,n+1);

f=sqrt(1-x.^2);

T(1)=(h/3)\*f(1);

for i=2:n

if mod(i,2)~=0

T(i)=(2\*h/3)\*f(i);

else

T(i)=(4\*h/3)\*f(i);

end

end

T(n+1)=(h/3)\*f(n+1);

piapprox=2\*sum(T);

%HW4 P2

%Trapzoidal rule

clear all; close all; clc

b=1;

a=0;

n=4;

h=(b-a)/n;

T=zeros(1,n+1);

x=linspace(0,1,n+1);

f=3\*x.^3-4\*x;

T(1)=(h/2)\*f(1);

for i=2:n

T(i)=h\*f(i);

end

T(n+1)=(h/2)\*f(n+1);

Tf=sum(T);

%HW4 P2

%Simpsons rule

clear all; close all; clc

b=1;

a=0;

n=2;

h=(b-a)/n;

T=zeros(1,n+1);

x=linspace(0,1,n+1);

f=3\*x.^3-4\*x;

T(1)=(h/3)\*f(1);

for i=2:n

if mod(i,2)~=0

T(i)=(2\*h/3)\*f(i);

else

T(i)=(4\*h/3)\*f(i);

end

end

T(n+1)=(h/3)\*f(n+1);

Tf=sum(T);

function [ f ] = f( x )

%Function in P3

f=5\*sin(2\*x.^4)+cos(x.^8);

end

function [ df ] = df( x )

%First derivative of function in P3

df=40\*x.^3.\*cos(2\*x.^4)-8\*x.^7.\*sin(x.^8);

end

%HW4 P3a

%Approximates Trapzoidal rule with refinement scheme

clear all; close all; clc

b=2;

a=-1;

n=3;

epsil=0.0001;

count=0;

while count<1000

for i=1:n+1

h(i)=(b-a)/n;

if count==0

initial(i)=a+h(i)\*i;

else

final(i)=a+h(i)\*i;

end

end

T=zeros(1,n+1);

x=linspace(a,b,n+1);

for i=1:n

T(i)=0.5\*h(i)\*(f(x(i))+f(x(i+1)));

end

Tf=sum(T);

for i=1:n

E(i)=(-1/12)\*h(i)^2\*(df(x(i+1))-df(x(i)));

end

ET=sum(E);

Error=Tf-4.8332650685747847;

if abs(ET)<=epsil

break

end

for i=1:n

eta(i)=abs(E(i))/max(abs(E));

NofNodes=length(x);

nofNewNodes=0;

if eta(i)>=0.5

xnew=(x(i+nofNewNodes)+x(i+1+nofNewNodes))/2;

x=[x(1:i+nofNewNodes),xnew,x(i+1+nofNewNodes:end)];

NofNodes=NofNodes+1;

nofNewNodes=nofNewNodes+1;

end

end

n=NofNodes-1;

count=count+1;

end

theta=abs(ET/Error);

fprintf('Number of iterations: %i\n',n);

fprintf('Tn(f) = %f\n',Tf);

fprintf('Asymptotic Error ET = %e\n',abs(ET));

fprintf('True Error E = %e\n',Error);

fprintf('Effectivity Index: %f\n',theta);

plot(x,T);

axis([-1,2,-6\*10^-3,6\*10^-3]);

hold on

plot(initial,[0 0 0 0],'.');

plot(final,zeros(1,length(final))-10^-3,'.');

Number of iterations: 3097

Tn(f) = 4.833365

Asymptotic Error ET = 9.905449e-05

True Error E = 1.003988e-04

Effectivity Index: 0.986610

%HW4 P3b

%Approximates Trapzoidal rule with uniform refinement scheme

clear all; close all; clc

b=2;

a=-1;

n=3;

epsil=0.0001;

count=0;

while count<1000

for i=1:n+1

h(i)=(b-a)/n;

end

T=zeros(1,n+1);

x=linspace(a,b,n+1);

for i=1:n

T(i)=0.5\*h(i)\*(f(x(i))+f(x(i+1)));

end

Tf=sum(T);

for i=1:n

E(i)=(-1/12)\*h(i)^2\*(df(x(i+1))-df(x(i)));

end

ET=sum(E);

Error=Tf-4.8332650685747847;

if abs(ET)<=epsil

break

end

for i=1:n

eta(i)=abs(E(i))/max(abs(E));

NofNodes=length(x);

nofNewNodes=0;

xnew=(x(i+nofNewNodes)+x(i+1+nofNewNodes))/2;

x=[x(1:i+nofNewNodes),xnew,x(i+1+nofNewNodes:end)];

NofNodes=NofNodes+1;

nofNewNodes=nofNewNodes+1;

end

n=NofNodes-1;

count=count+1;

end

theta=abs(ET/Error);

fprintf('Number of iterations: %i\n',n);

fprintf('Tn(f) = %f\n',Tf);

fprintf('Asymptotic Error ET = %e\n',abs(ET));

fprintf('True Error E = %e\n',Error);

fprintf('Effectivity Index: %f\n',theta);

Number of iterations: 6144

Tn(f) = 4.833290

Asymptotic Error ET = 2.516832e-05

True Error E = 2.525358e-05

Effectivity Index: 0.996624

%HW4 P3c

%Approximates Trapzoidal rule with uniform refinement scheme and asymptotic

%error addition

clear all; close all; clc

b=2;

a=-1;

n=3;

epsil=0.0001;

count=0;

while count<1000

h=zeros(1,n+1);

for i=1:n+1

h(i)=(b-a)/n;

end

x=linspace(a,b,n+1);

for i=1:n

E(i)=(-1/12)\*h(i)^2\*(df(x(i+1))-df(x(i)));

end

ET=sum(E);

T=zeros(1,n+1);

for i=1:n

T(i)=0.5\*h(i)\*(f(x(i))+f(x(i+1)))+E(i);

end

Tf=sum(T);

Error=Tf-4.8332650685747847;

if abs(Error)<=epsil

break

end

nofNewNodes=0;

eta=zeros(1,n);

for i=1:n

eta(i)=abs(E(i))/max(abs(E));

NofNodes=length(x);

xnew=(x(i+nofNewNodes)+x(i+1+nofNewNodes))/2;

x=[x(1:i+nofNewNodes),xnew,x(i+1+nofNewNodes:end)];

NofNodes=NofNodes+1;

nofNewNodes=nofNewNodes+1;

end

n=NofNodes-1;

count=count+1;

end

theta=abs(ET/Error);

fprintf('Number of iterations: %i\n',n);

fprintf('Tn(f) = %f\n',Tf);

fprintf('Asymptotic Error ET = %e\n',abs(ET));

fprintf('True Error E = %e\n',Error);

fprintf('Effectivity Index: %f\n',theta);

Number of iterations: 1536

Tn(f) = 4.833289

Asymptotic Error ET = 4.026931e-04

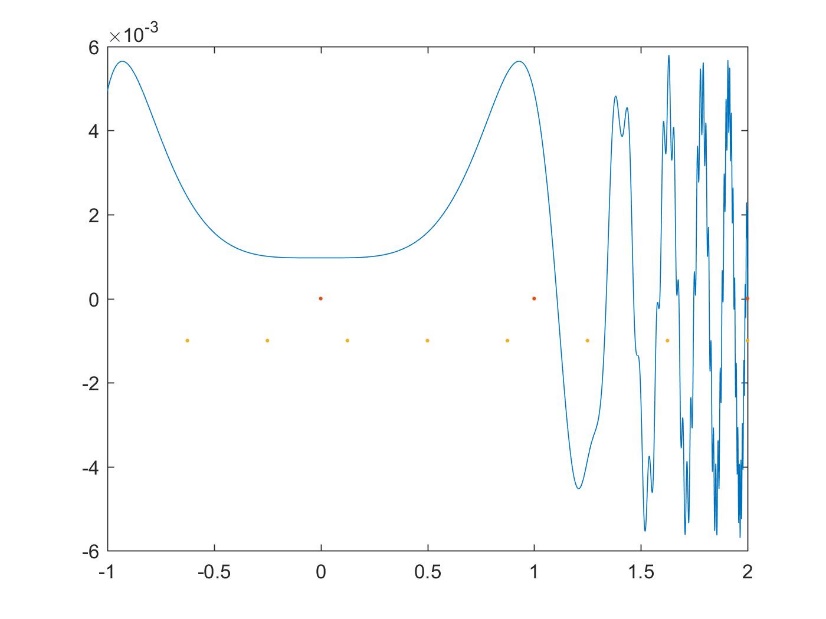
True Error E = 2.398863e-05

Effectivity Index: 16.786830

3d)

|  |  |  |
| --- | --- | --- |
|  | n | Θn |
| Part a) | 3097 | 0.986610 |
| Part b) | 6144 | 0.996624 |
| Part c) | 1536 | 16.786830 |

The adaptive refinement scheme, relative to the uniform one, has both a lower number of iteration and a lower effectivity index. This can be expected due to the fact that an adaptive method will create many fewer evaluation points, which is computationally less expensive but also leaves room for more error, so we deal with the tradeoff we often face with limited resources of time and computation. Part c) is fairly unique in that it has very few evaluation points, and an extremely high effectivity index. This also makes sense in that a large effectivity shows an upper error bound, which when operating off of tells the code it requires fewer points of evaluation to refine its approximation. This method is quick and overestimates error by a large margin, which can be expected when dealing with a piecewise linear numerical method.

3e)

In the figure, Tn(f) is plotted versus x, with the initial distribution of evaluation points shown as red dots. The evaluation points after 5 uniform refinements are shown as orange dots, and by inspection one can see that they have an even distribution and are more frequent than the initial evaluation points, telling us that the algorithm is working smoothly.