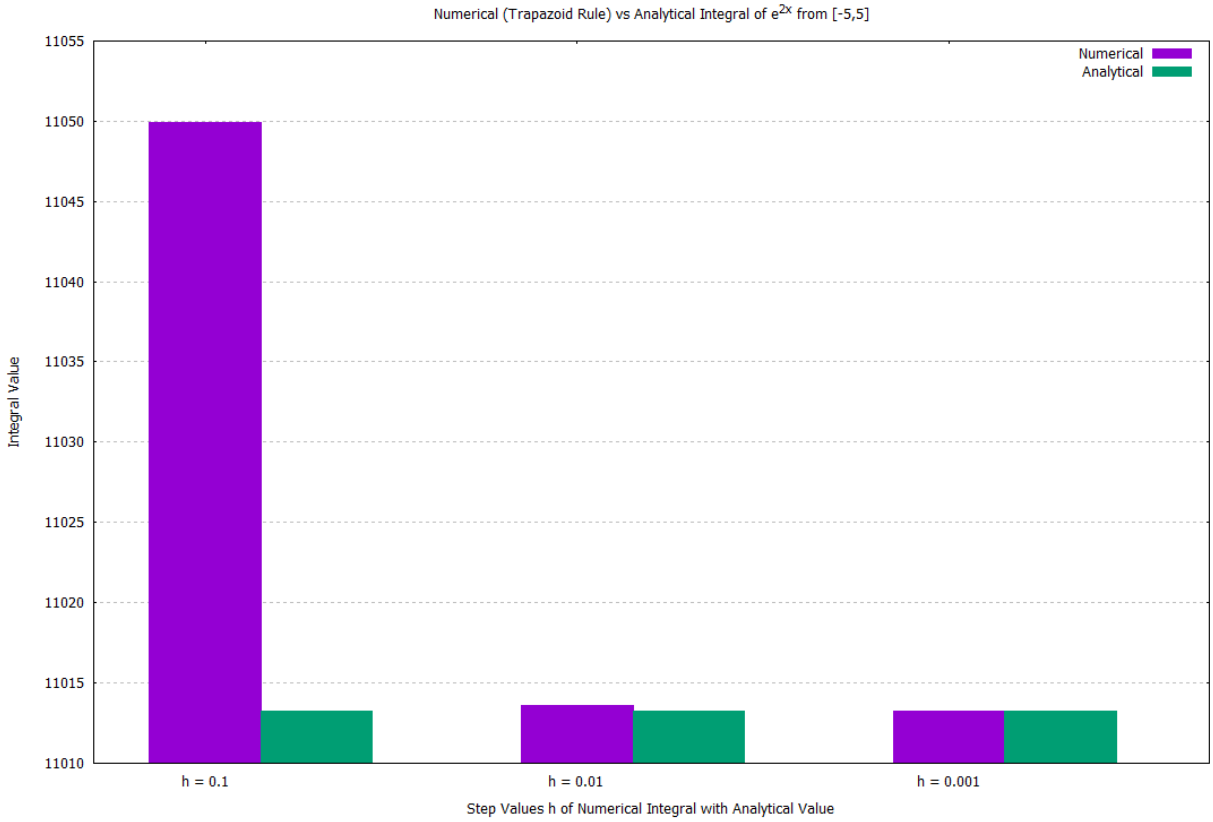
Andrew Camps

330 - Homework 3

9/18/2016

Problem 1: Trapezoid Rule

Graph 1:



Description:

This graph shows the Trapezoid Rule with three different h values. As the h values get smaller the values of the numerical integral become more accurate. As seen, there is much more error in the 0.1 value of h compared to the other two values of h. The function used in the graph above is e^2x from [-5, 5].

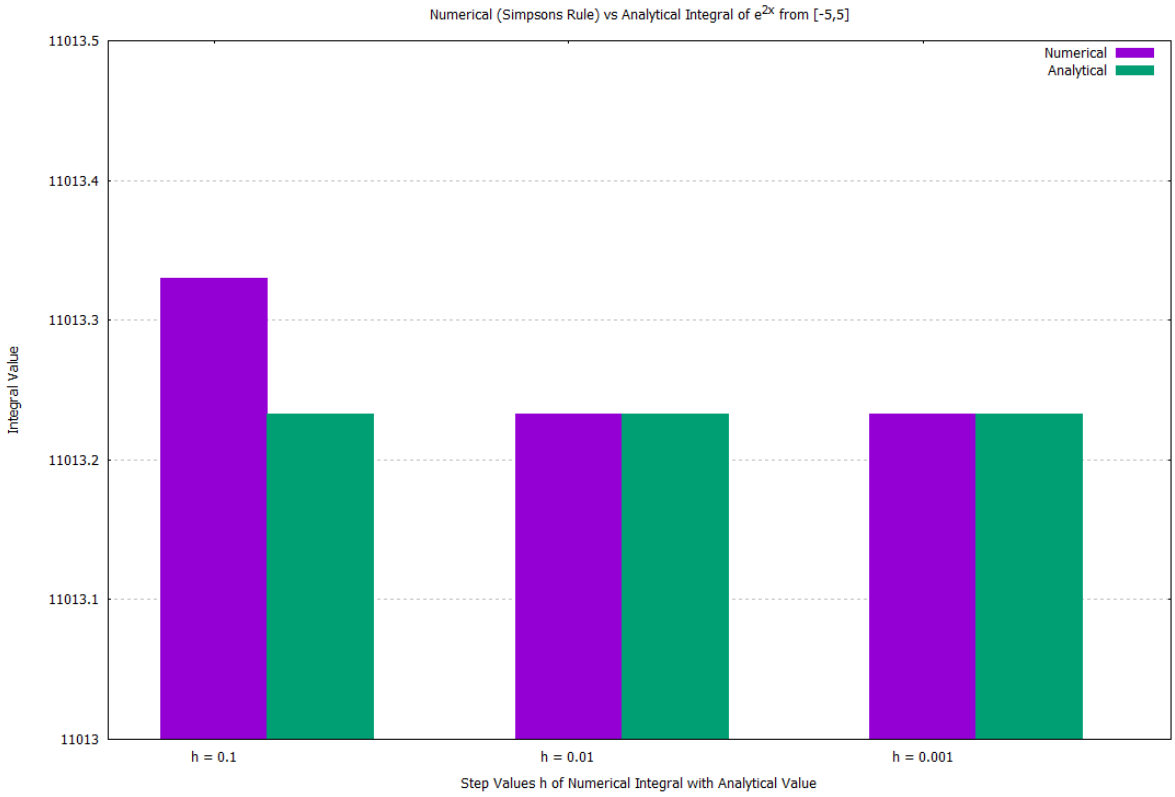
Function Code:

```
function [ y ] = integrateTrap( func, h, lower, upper )
%Integrate Trapazoid Rule
%   This function takes a numerical integral of a function given the upper
%   and lower bounds and step value using the Trapezoid Rule

numPoints = ((upper - lower) / h) + 1; %number of total points
yA = zeros(1, ceil(numPoints)); %getting memory
x = zeros(1, ceil(numPoints));
k = 2;

yA(1) = (h / 2) * (func(lower)); %for the first point
x(2) = lower + h;

%yT(1) = func(lower) * (h / 2);% First point calculated without the loop
%x(2) = lower + h; %First point to be used in the loop

while k < numPoints %Goes till the point before the last point
    yA(k) = h * func(x(k)); %Calculates all the middle points
    k = k + 1; %increments array
    x(k) = x(k - 1) + h; %increments x
end


x(k) = x(k - 1) + h;
yA(k) = (h / 2) * func(x(k)); %Caluculates the last point outside the loop

y = sum(yA); %Sets output
```

GNU Code:

```
#!/gnuplot
clear
reset

#Labels
set xlabel 'Step Values h of Numerical Integral with Analytical Value'
set ylabel 'Integral Value'
set title 'Numerical (Trapazoid Rule) vs Analytical Integral of e^2^x from [-5,5]'
set xtics ("h = 0.1" 0.3, "h = 0.01" 1.3, "h = 0.001" 2.3,)

#Axis range
set xrange [0:3]
set yrange [11010:11055]

#Style
set boxwidth 0.3
set style fill solid
set ytics 5
set grid ytics

#Plot
plot 'Problem1.dat' every 2 using 1:2 with boxes ls 1 title "Numerical", 'Problem1.dat' every
2::1 using 1:2 with boxes ls 2 title "Analytical"

#Save
save "HW3_1.gnu"

#EOF
```

Problem 2: Simpson's Rule



Description:

This graph shows Simpson's Rule with three different h values. As the h values get smaller, the values of the numerical integral become more accurate. The function used in the graph above is e^2x from [-5, 5]. Compared to the Trapezoid Rule, Simpson's Rule is more accurate right away on step values h = 0.1 and then continue to become more accurate as the values get smaller.

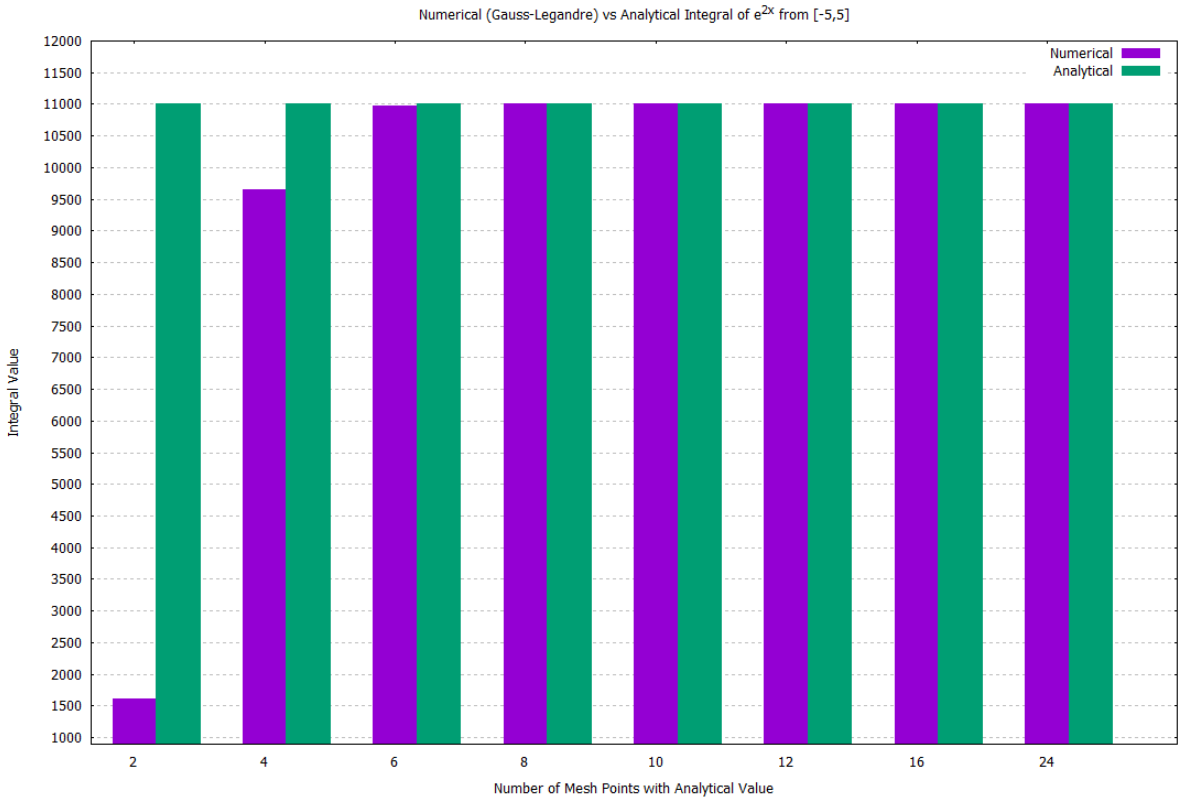
Function Code:

```
function [ y ] = integrateSimpson( func, h, lower, upper )
%Integrate Simpson's Rule
%   This function takes a numerical integral of a function given the upper
%   and lower bounds and step value using Simpson's Rule

numPoints = ((upper - lower) / h) + 1; %number of points
yA = zeros(1, ceil(numPoints)); %Getting memory
x = zeros(1, ceil(numPoints));
k = 2;

yA(1) = (h / 3) * (func(lower)); %For the first point
x(2) = lower + h;

while k < numPoints %Goes till the point before the last point

    if mod(k, 2) == 0
        yA(k) = ((4 * h) / 3) * func(x(k)); %Calculates all the middle points
    else
        yA(k) = ((2 * h) / 3) * func(x(k)); %Calculates all the middle points
    end

    k = k + 1; %increments array
    x(k) = x(k - 1) + h; %increments x

end


x(k) = x(k - 1) + h;
yA(k) = (h / 3) * func(x(k)); %Caluculates the last point outside the loop

y = sum(yA); %Sets output

end
```

GNU Code:

```
#!/gnuplot
clear
reset

#Labels
set xlabel 'Step Values h of Numerical Integral with Analytical Value'
set ylabel 'Integral Value'
set title 'Numerical (Simpsons Rule) vs Analytical Integral of e^2^x from [-5,5]'
set xtics ("h = 0.1" 0.3, "h = 0.01" 1.3, "h = 0.001" 2.3,)

#Axis Range
set xrange [0:3]
set yrange [11013:11013.5]

#Style
set boxwidth 0.3
set style fill solid
set ytics .1
set grid ytics

#Plot
plot 'Problem2.dat' every 2 using 1:2 with boxes ls 1 title "Numerical", 'Problem2.dat' every
2::1 using 1:2 with boxes ls 2 title "Analytical"

#Save
save "HW3_2.gnu"

#EOF
```

Problem 3:



Description:

This last graph is a numerical integral of e^2x from [-5, 5] using the Gauss-Legandre Method with eight different values of mesh points shown on the x-axis. Starting with only two mesh points the integral is not very accurate, but as more mesh points are introduced the accuracy increases drastically. As you can see at about six mesh points the integral is fairly accurate and begins to level off starting at eight mesh points. To test this integral function I first tried it on a couple of integral I could easily calculate or knew off the top of my head such as x^2 from [-5, 5] equal to about 83.3. Other function I tried were cos(x) from [-pi, pi] as well as x from [-5, 5]. All tested function came out with accurate results at about 6, 8 or 10 mesh points.

Function Code:

```
function [ y ] = integrateGL( func, N, lower, upper )
%Integrate Gauss-Legandre
%   Function import mesh points from .DAT files and calculates the integral
%   of various function using Gauss-Legandre Integration

%Importing the various weights in the data files based on the value of N
if N == 2
    A = importdata('GAUSS-02.DAT');
elseif N == 4
    A = importdata('GAUSS-04.DAT');
elseif N == 6
    A = importdata('GAUSS-06.DAT');
elseif N == 8
    A = importdata('GAUSS-08.DAT');
elseif N == 10
    A = importdata('GAUSS-10.DAT');
elseif N == 12
    A = importdata('GAUSS-12.DAT');
elseif N == 16
    A = importdata('GAUSS-16.DAT');
elseif N == 24
    A = importdata('GAUSS-24.DAT');
else
    A = 0;
end

k = 1;
mul = (upper - lower) / 2; %For less clutter

while k <= N %Going through all the mesh points

    f = A(k,:); %to grab the right points in the data
    yA(k) = (f(2) * func((mul * f(1)) + ((upper + lower) / 2))); %Part of calculation
    k = k + 1;

end

y = mul * sum(yA); %Add all up and multiply by final number

end
```

GNU Code:

```
#!/gnuplot
clear
reset

#Labels
set xlabel 'Number of Mesh Points with Analytical Value'
set ylabel 'Integral Value'
set title 'Numerical (Gauss-Legandre) vs Analytical Integral of e^2^x from [-5,5]'
set xtics ("2" 0.2, "4" 0.8, "6" 1.4, "8" 2, "10" 2.6, "12" 3.2, "16" 3.8, "24" 4.4, )

#Axis Range
set xrange [0:5]
set yrange [900:12000]

#Style
set boxwidth 0.2
set style fill solid
set ytics 500
set grid ytics

#Plot
plot 'Problem3.dat' every 2 using 1:2 with boxes ls 1 title "Numerical", 'Problem3.dat' every
2::1 using 1:2 with boxes ls 2 title "Analytical"
```

```
#Save
save "HW3_3.gnu"

#EOF
```

Main Matlab Code:

```
%Problem 1: Trapazoid Rule
clear
h(1) = 0.1; %Step Widths
h(2) = 0.01;
h(3) = 0.001;
lowerBound = -5; %Bounds
upperBound = 5;

y(1) = integrateTrap(@(x)exp(2*x), h(1), lowerBound, upperBound); %Inegrating using the Trapazoid
rule with each step width
y(2) = integral(@(x)exp(2*x), lowerBound, upperBound); %Analytical Integral for gnu
y(3) = integrateTrap(@(x)exp(2*x), h(2), lowerBound, upperBound);
y(4) = integral(@(x)exp(2*x), lowerBound, upperBound);
y(5) = integrateTrap(@(x)exp(2*x), h(3), lowerBound, upperBound);
y(6) = integral(@(x)exp(2*x), lowerBound, upperBound);

z = [0.3, 0.6, 1.3, 1.6, 2.3, 2.6]; %For gnu plotting
A = [z;y];
A = A';

save('Problem1.DAT', 'A', '-ASCII'); %Export to .txt for gnuplot to read

%%
%Problem 2: Simpson's Rule
clear
h(1) = 0.1; %Step Widths
h(2) = 0.01;
h(3) = 0.001;
lowerBound = -5; %Bounds
upperBound = 5;

y(1) = integrateSimpson(@(x)exp(2*x), h(1), lowerBound, upperBound); %Inegrating using Simpson's
rule with each step width
y(2) = integral(@(x)exp(2*x), lowerBound, upperBound); %Analytical Integral for gnu
y(3) = integrateSimpson(@(x)exp(2*x), h(2), lowerBound, upperBound);
y(4) = integral(@(x)exp(2*x), lowerBound, upperBound); %Analytical Integral for gnu
y(5) = integrateSimpson(@(x)exp(2*x), h(3), lowerBound, upperBound);
y(6) = integral(@(x)exp(2*x), lowerBound, upperBound); %Analytical Integral for gnu

z = [0.3, 0.6, 1.3, 1.6, 2.3, 2.6]; %For gnu plotting
A = [z;y];
A = A';

save('Problem2.DAT', 'A', '-ASCII'); %Export to .txt for gnuplot to read

%%
%Problem 3: GL Integration
clear
k = 1;
num = 16;
N(1) = 2; %Mesh Points
N(3) = 4;
N(5) = 6;
N(7) = 8;
N(9) = 10;
N(11) = 12;
N(13) = 16;
N(15) = 24;
lowerBound = -5; %Bounds
upperBound = 5;

while k <= num
    if mod(k, 2) == 1
```

```
        y(k) = integrateGL(@(x)exp(x * 2), N(k), lowerBound, upperBound); %Inegrating using GL
with each number of mesh points
    else
        y(k) = integral(@(x)exp(2*x), lowerBound, upperBound); %Analytical Integral for gnu
    end
    k = k + 1;
end

z = [0.2, 0.4, 0.8, 1, 1.4, 1.6, 2, 2.2, 2.6, 2.8, 3.2, 3.4, 3.8, 4, 4.4, 4.6]; %For gnu plotting
A = [z;y];
A = A';

save('Problem3.DAT', 'A', '-ASCII'); %Export to .txt for gnuplot to read
```