Derek Frank
dmfrank@ucsc.edu
AMS 147
2/10/10

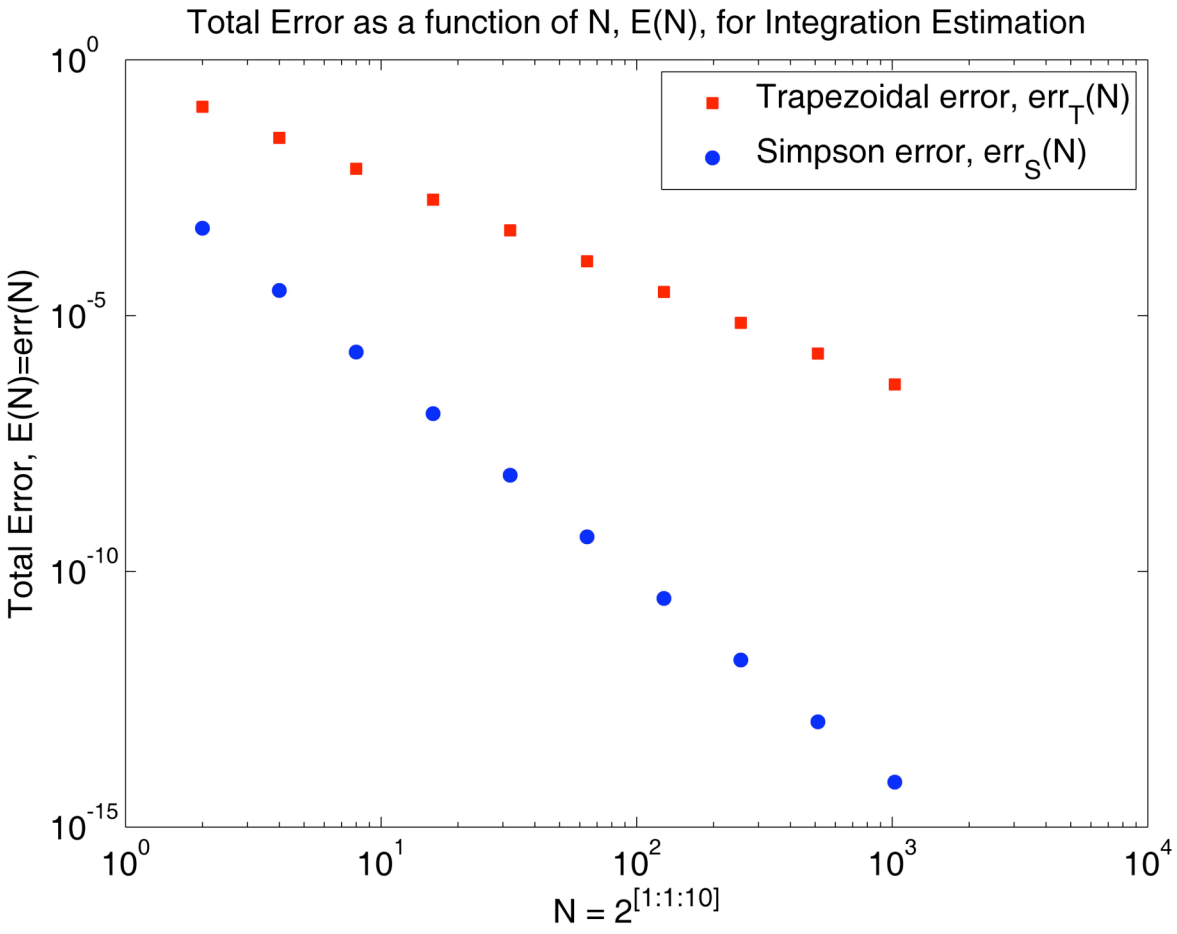
Homework #4

1)

a. In the first problem I am to solve for the error produced for each the composite Trapezoidal rule and the composite Simpson's rule when using the rules to estimate the ${}^{b}\int_{a} sinx\partial x$, where $a=0$ and $b=2$. As well, my interval of N, in the step size of $h=(b-a)/N$, is $N=2^{[1:1:10]}$.

b. Obviously, by implementing both the composite Trapezoidal rule and composite Simpson's rule in Matlab, I will attain 10 values of error, $E(N)$, as a function of N, for each rule. I begin by altering the given codes in "Num_integration.zip" to satisfy storing 10 values of error. This requires changing 1×1 vectors into 10×1 vectors and x and y values into 10×N or 10×N+1 matrices. As well, nested for loops are necessary to store and display information.

c.



d. The results show the approximation of total error as a function of N for each the composite Trapezoidal and Simpson rules. According to the graph, the total error obtained by the Simpson rule is not as great as that obtained by the Trapezoidal rule. This is due to the fact that the Simpson rule total error is a fourth order integration method, while the total error from the Trapezoidal rule is a second order method. Typically, the higher the order integration

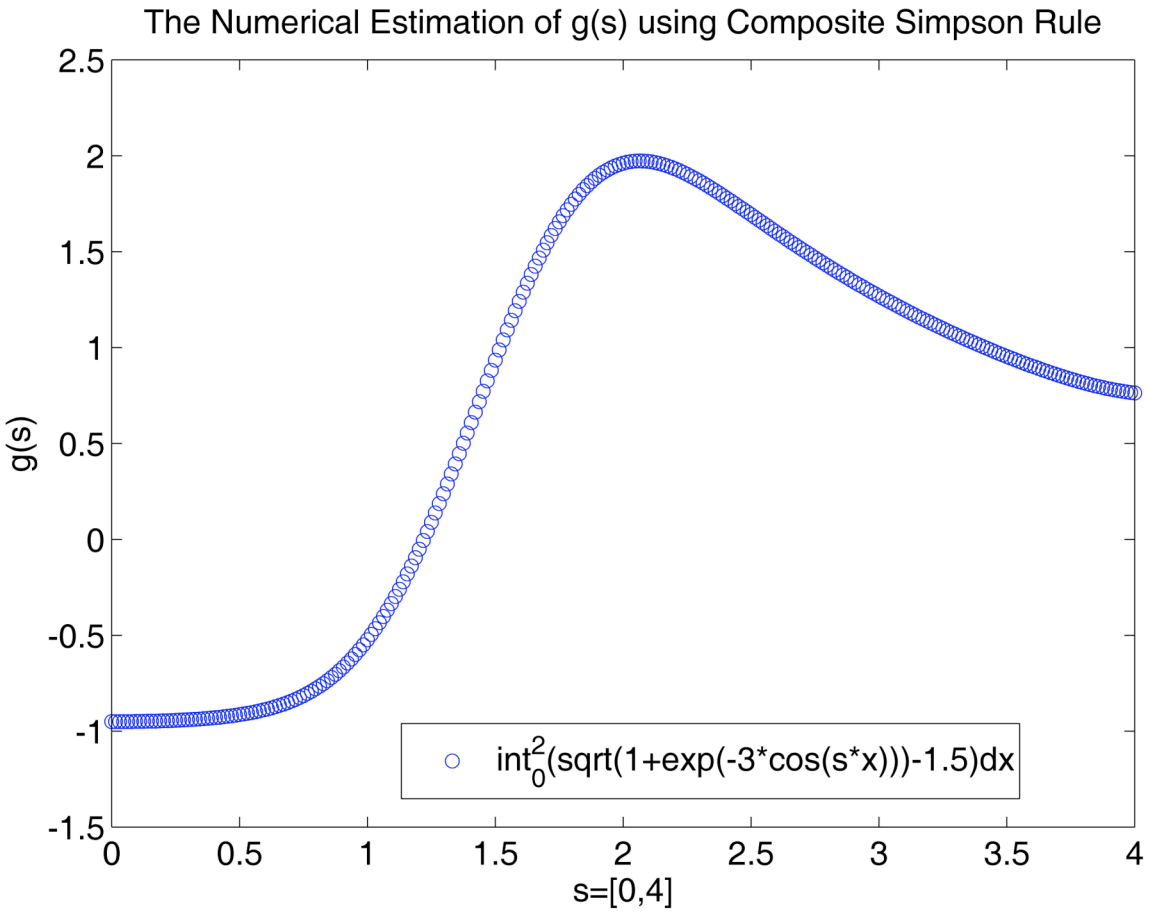
method used, the more accurate the approximation of an integration is on a computer. On this log scale, both errors apear linear, with the accuracy of approximation getting smaller as N gets bigger. This is because as N gets bigger, the step size becomes smaller.

2)

a. In this problem, I am to solve for the numerical estimation of $g(s)=\int_0^2\sqrt{(1+e^{-3\cos(sx)})}-1.5\,\partial x$ by implementing the composite Simpson rule in Matlab. My step size is $h=2/N$, where $N=256$. I am to plot $g(s)$ for $s=[0,4]$.

b. To solve this problem, I treat s as a constant, so that I am able to implement the composite Simpson rule. To attain a curve, I increment s by $\partial s=4/N$, which gives me a vector for s containing $N+1$ values. Then, I solve for each Simpson numerical estimation, S, value, using a for loop, and plugging into the composite Simposon formula its corresponding value of s with the entire vectors of x_i and $x_{i-1/2}$ each loop. I attain a vector for S the same size as s. I now plot the two vectors containing 257 values to view the curve.

c.



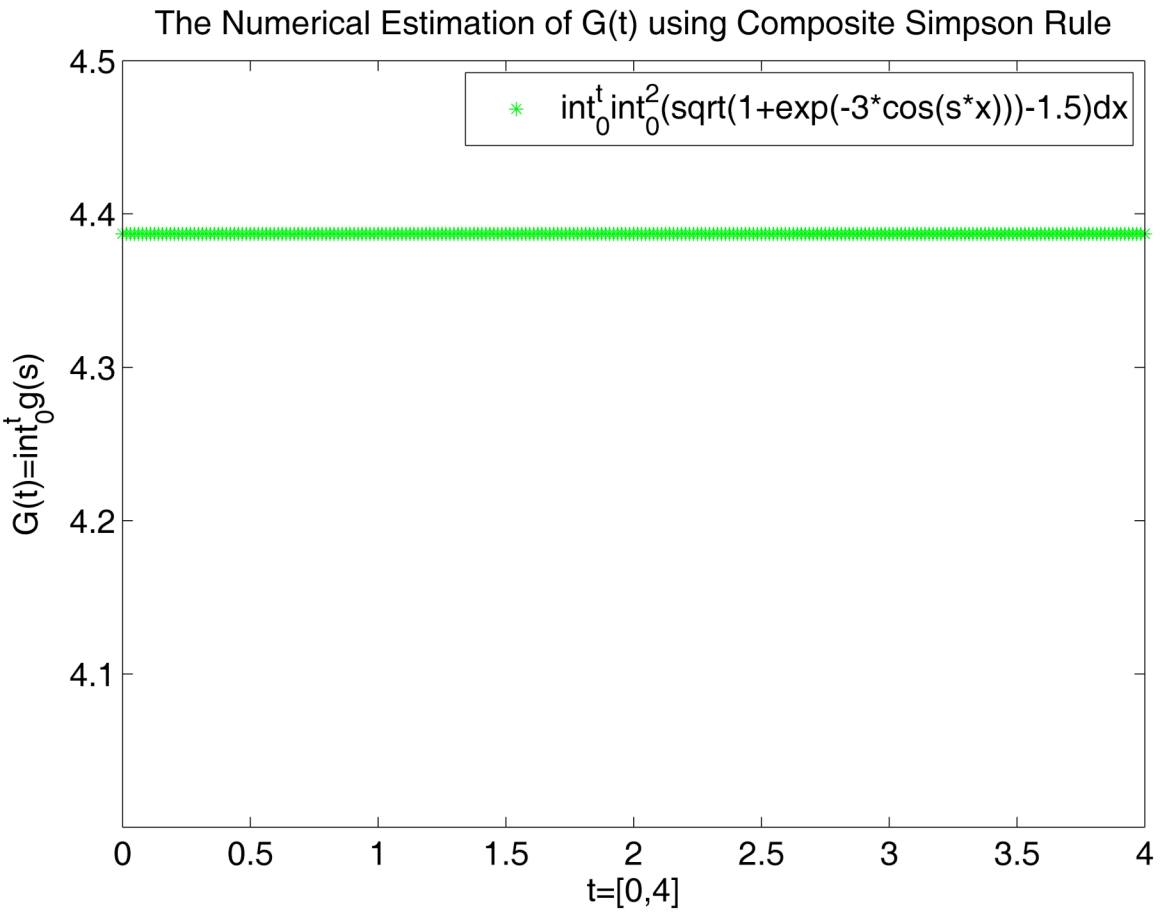
d. The results show the numerical estimation of $g(s)$ using the composite Simpson rule. g(s) is increasing very slowly from $s\approx[0,0.5]$, then begins to increase rapidly until $g(s)$ reaches a maximum at $s\approx2$, $g(s)\approx2$. Now, after $s\approx2$, $g(s)$ begins to decrease slowly.

3)

a. In this final problem, I am to again use the composite Simpson rule to calculate $G(t)= {}^{t}\int_0 g(s)\,\partial s = {}^{t}\int_0 {}^{2}\int_0 \sqrt{(1+e^{-3\cos(sx)})-1.5}\,\partial x\,\partial s$, where $t\ \varepsilon[0,4]$ and the step size for the inner integration numerical estimation is, again, $h(x)=2/N$ and for the outer integration is $h(s)=4/N$, $N=256$.

b. To solve this final problem, as usual using Matlab, I must focus on approximating the outer integral, since I already have the inner information, I can continue to determine how to calculate the outer integral. The outer integral requires, just as the inner does, two solved vectors for its variable. The inner integration required the sum of all incremented values of $f(x_i)$(vector length of 257) and $f(x_{i-1/2})$ (vector length of 256). The outer integration requires, similarly, the sum of all incremented values of $g(s_i)$ and $g(s_{i-1/2})$. So I must compute $g(s_{i-1/2})$. I can then plug these $g(s)$ vectors into the outer integral and solve just as the inner integration by solving with the vector of $t\ \varepsilon[0,4]$. I increment t by $4/N$ and attain a vector of length 257. After plugging everything in I am now ready to graph.

c.



d. The results show a horizontal line for $G(t)$ on the interval where $t\ \varepsilon[0,4]$. It is clear the function does not increase or decrease. $G(t)$ stays under 4.4 for the entire interval.

Appendix:

1. "f.m"

```
function [y]=f(x)
% This function calculates f(x).
%
y=sin(x);
```

"trapezoidal.m"

```
% This code uses the composite Trapezoidal rule to calculate
% int_{a}^{b} f(x) dx. f(x)=sinx
% The error is calculated using the exact solution.
%
a=0; b=2;
I=cos(a)-cos(b);
%
N=2.^[1:1:10];
M=10;
h=zeros(M);
x=zeros(M,N(M)+1);
y=zeros(M,N(M)+1);
T=zeros(M);
errT=zeros(M);
for i=1:M,
    h(i)=(b-a)/N(i);
    for j=0:N(i),
        x(i,j+1)=a+j*h(i);
        y(i,j+1)=f(x(i,j+1));
    end
    T(i)=(y(i,1)+y(i,N(i)+1)+2*sum(y(i,2:N(i))))*h(i)/2;
    errT(i)=abs(T(i)-I);
end
%
save trap.mat T errT N M
% display values
for i=1:M,
    disp(' ')
    disp(['  The numerical result by the composite Trapezoidal'])
    disp(['  rule with N = ',num2str(N(i)),' is'])
    disp(['         T = ',num2str(T(i),'%16.8e'),'.'])
    disp(['  The error is ',num2str(errT(i),'%16.8e'),'.'])
    disp(' ')
end
```

"simpson.m"

```
% This code uses the composite Simpson rule to calculate
% int_{a}^{b} f(x) dx.
% The error is calculated using the exact solution.
%
a=0; b=2;
I=cos(a)-cos(b);
%
N=2.^[1:1:10];
M=10;
h=zeros(M);
x=zeros(M,N(M)+1);
y=zeros(M,N(M)+1);
x2=zeros(M,N(M));
y2=zeros(M,N(M));
```

```
S=zeros(M);
errS=zeros(M);
for i=1:M,
    h(i)=(b-a)/N(i);
    for j=0:N(i),
        x(i,j+1)=a+j*h(i);
        y(i,j+1)=f(x(i,j+1));
    end
    for j=0:N(i)-1,
        x2(i,j+1)=a+j*h(i)+h(i)/2;
        y2(i,j+1)=f(x2(i,j+1));
    end

S(i)=(y(i,1)+y(i,N(i)+1)+2*sum(y(i,2:N(i)))+4*sum(y2(i,1:N(i))))*h(i)/6;
    errS(i)=abs(S(i)-I);
end
%
save simp.mat S errS
% display values
for i=1:M,
    disp(' ')
    disp(['  The numerical result by the composite Simpson'])
    disp(['  rule with N = ',num2str(N(i)),' is'])
    disp(['         S = ',num2str(S(i),'%16.8e'),'.'])
    disp(['  The error is ',num2str(errS(i),'%16.8e'),'.'])
    disp(' ')
end
```

"plot_trap&simp.m"

```
% Consider the integral from [a,b] of sinxdx.
% Two curves are approximated using first composite Trapezoidal rule
% and second with the composite Simpson rule.  Each is evaluated with
% a step size of h=(b-a)/N, where N=2^[1:1:10]. The total error is
% then solved for with the exact value = cosa-cosb.
% The code "trapezoidal.m" approximates total error for all N and
% stores the data in "trap.mat." The code "simpson.m" approximates
% total error for all N and stores the data in "simp.mat."
% This code loads in "trap.mat" and "simp.mat" and plots the errors as
% a function of N.
%
clear
figure(1)
clf
axes('position',[0.15,0.13,0.75,0.75])
%
load trap.mat T errT N M
load simp.mat S errS
% plot curve
for i=1:M,
    loglog(N(i), errT(i),'rs', 'markerfacecolor', 'r')
    hold on
    loglog(N(i), errS(i),'bo', 'markerfacecolor', 'b')
    hold on
end
set(gca,'fontsize',14)
xlabel('N = 2^[^1^:^1^:^1^0^]')
ylabel('Total Error, E(N)=err(N)')
title('Total Error as a function of N, E(N), of Composite Trapezoidal &
Simpson Rule')
legend('Trapezoidal error, errT(N)', 'Simpson error, errS(N)')
```

2. “fp.m”

```
function [yp]=fp(x,s)
% This function calculates fp(x,s).
%
yp=sqrt(1+exp(-3*cos(s*x)))-1.5;
```

“simpson_g.m”

```
% This code uses the composite Simpson rule to calculate
% g(s)=int_{0}^{2} fp(x)dx, s in [0,4].
%
clear
figure(2)
clf
axes('position',[0.15,0.13,0.75,0.75])
%
a=0;
N=256;
h=2/N;
ds=4/(N); s=[0:ds:4];
%
x=a+[0:N]*h;
x2=a+[0:N-1]*h+h/2;
%
y=zeros(N+1);
y2=zeros(N);
S=zeros(N+1);
for i=1:N+1,
    for j=1:N+1,
        y(j)=fp(x(j),s(i));
        if j~=N+1,
            y2(j)=fp(x2(j),s(i));
        end
    end
    S(i)=(y(1)+y(N+1)+2*sum(y(2:N))+4*sum(y2(1:N)))*h/6;
end
%
for i=1:N+1,
    plot(s(i), S(i),'bo')
    hold on
    disp(' ')
    disp(['  The numerical result by the composite Simpson'])
    disp(['  rule with N = ',num2str(N),' is'])
    disp(['         S = ',num2str(S(i),'%16.8e'),'.'])
    disp(' ')
end
axis([0,4,-1.5,2.5])
set(gca,'fontsize',14)
xlabel('s=[0,4]')
ylabel('g(s)')
title('The Numerical Estimation of g(s) using Composite Simpson Rule')
legend('int_{0}^{2}(sqrt(1+exp(-3*cos(s*x)))-1.5)dx')
```

3. “simpson_G.m”

```
% This code uses the composite Simpson rule to calculate
% g(s)=int_{0}^{2} fp(x)dx, s in [0,4].
%
clear
```

```
figure(3)
clf
axes('position',[0.15,0.13,0.75,0.75])
%
a=0; b=4;
N=256;
hx=2/N;
hs=b/N;
% first calculate g(s)
x=a+[0:N]*hx;
x2=a+[0:N-1]*hx+hx/2;
s=a+[0:N]*hs;
s2=a+[0:N-1]*hs+hs/2;
%
y=zeros(N+1);
y2=zeros(N);
g=zeros(N+1);
for i=1:N+1,
    for j=1:N+1,
        y(j)=fp(x(j),s(i));
        if j~=N+1,
            y2(j)=fp(x2(j),s(i));
        end
    end
    g(i)=(y(1)+y(N+1)+2*sum(y(2:N))+4*sum(y2(1:N)))*hx/6;
end
z=zeros(N);
z2=zeros(N);
g2=zeros(N);
for i=1:N,
    for j=1:N,
        z(j)=fp(x(j),s2(i));
        z2(j)=fp(x2(j),s2(i));
    end
    g2(i)=(z(1)+z(N)+2*sum(z(2:N-1))+4*sum(z2(1:N-1)))*hs/6;
end
%
% now calculate G(t)
t=[0:hs:4];
G=zeros(N+1);
for i=1:N+1,
    G(i)=(g(1)+g(N+1)+2*sum(g(2:N))+4*sum(g2(1:N)))*hs/6;
end
% plot and display results
for i=1:N+1,
    plot(t(i), G(i),'g*')
    hold on
    disp(' ')
    disp(['  The numerical result by the composite Simpson'])
    disp(['  rule with N = ',num2str(N),' is'])
    disp(['        S = ',num2str(G(i),'%16.8e'),'.'])
    disp(' ')
end
axis([0,4,4,4.5])
set(gca, 'ytick', [4.1:.1:4.5])
set(gca,'fontsize',14)
xlabel('t=[0,4]')
ylabel('G(t)=int_{0}^{t}g(s)')
title('The Numerical Estimation of G(t) using Composite Simpson Rule')
legend('int_{0}^{t}int_{0}^{2}(sqrt(1+exp(-3*cos(s*x)))-1.5)dx')
```