Derek Frank
dmfrank@ucsc.edu
AMS 147
3/3/10

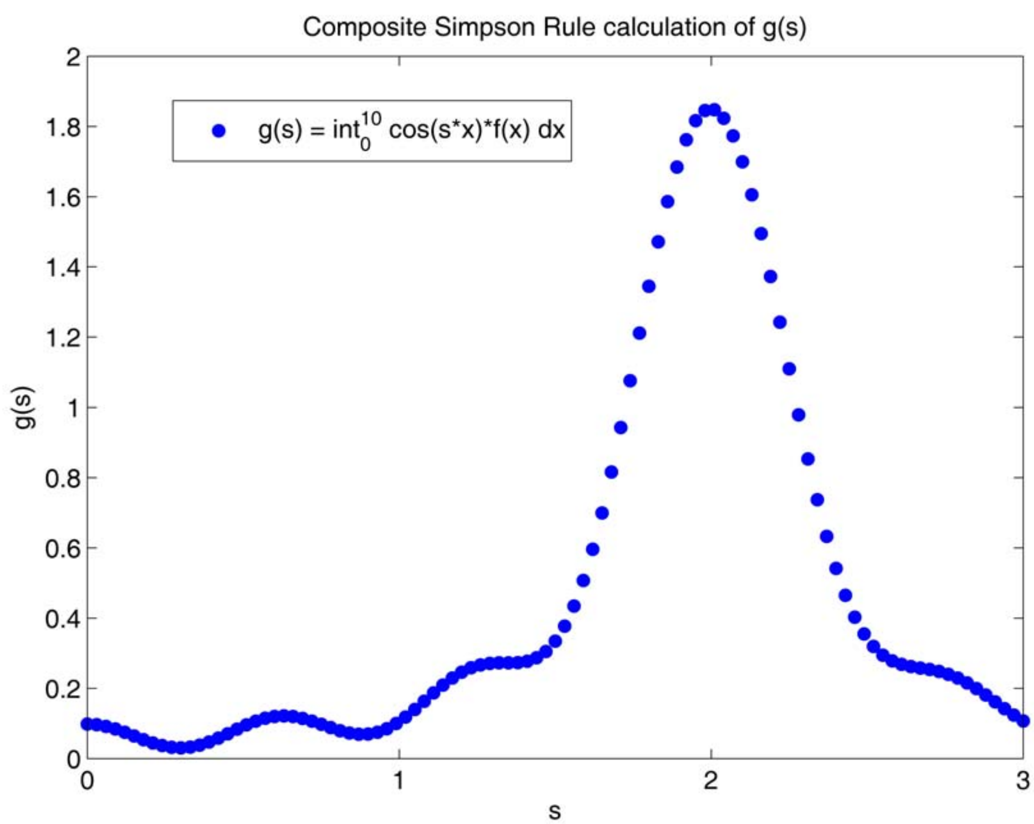
Homework #7

1.

a. In this first problem, I am to calculate $g(s) = {}^{10}\int_0 \cos(sx)f(x)\partial x$ using the composite Simpson rule, with $N=512$ and for $s\epsilon[0,3]$, and given $f(x)$ exists in the data found in "data4.txt."

b. Using Matlab, I will implement both the composite Simpson rule and cubic spline. The cubic spline is necessary to find a fitting function for $f(x)$ so that I may caluculate the integral. I will evaluate $g(s)$ for 100 points between [0,3], $\partial s={}^{3}/_{100}$, and store each value in a vector to plot with its corresponding s value.

c.



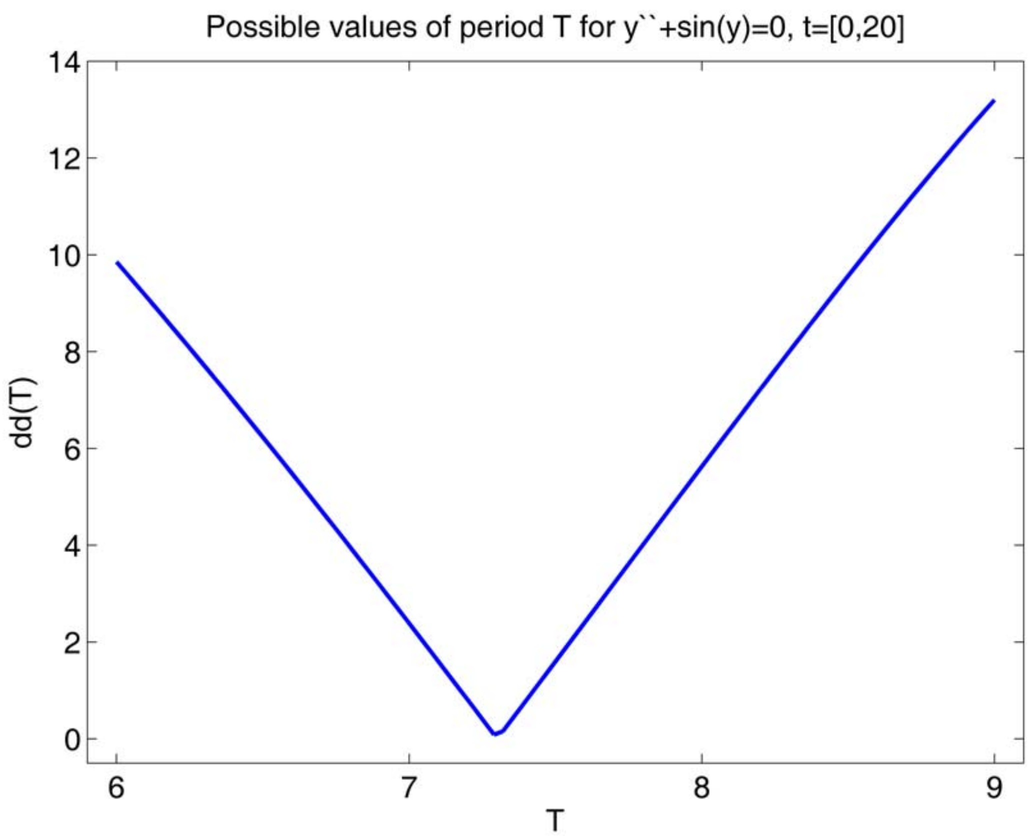
d. The results show $g(s)$ on the interval $s\epsilon[0,3]$. On this interval, a minimum is attained approximately when $s=0.3$ and $g(s)=0.0304855005$, while a maximum approximately occurs when $s=2.01$ and $g(s)=1.84761556$.

2.

a. In this second problem, I am to solve $y``+sin(y)=0$ for $y(0)=1.5$ and $y`(0)=0$ using the Runge-Kutta method, with time step $h=0.05$ from $t\epsilon[0,20]$. After noticing that this plot is periodic, with period T (somewhere between [5,10]), I am to find $dd(T)= \sqrt{(y(0.1j)-y(0.1j+T))^2}$, from $j\epsilon[0,80]$ (since t will exist for $t\epsilon[0,8]$), to find the distance between $y(t)$ and $y(t+T)$, since the definition of periodic requires these values to be equal. Additionally, the inner sums of the square root require the use of the spline interpolation function. The minimum of the curve $dd(T)$ with respect to T will be the best approximation of T for this set of data. Ideally, $dd(T)$ should be equal to zero at this minimum.

b. Using Matlab, I implement the Runge-Kutta method, the distance formula, and the spline interpolation function to find the minimum distance for values of $T\epsilon[6,9]$. I first solve for $y(t)$ using the Runge-Kutta method. I then plug these values into the distance formula I implemented, which requires the use of the spline method to fit a function to the t and $y(t)$ values attained with the Runge-Kutta method. I then plot the found $dd(T)$ values with their corresponding T values.

c.



d. The resulting show plot shows a linearly decreasing function until T approaches a minimum. The function then begins to increase linearly at the opposite rate of its previous decreasing rate. The minimum attained is approximately when $T=7.29$ and $dd(T)=0.086860407$.

3.

a. In this final problem, I am to use the golden search method combined with the cubic spline interpolation function to find the value of T that corresponds to the minimum value of $dd(T)$, which is ideally at $dd(T)=0$.
b. Using Matlab, I will implement the golden search method for $T\epsilon[6,9]$ to locate the minimum value.
c. Golden search $\Longrightarrow$ $T=7.300900$
d. The resulting minimum occurs when $T=7.300900$.
NOTE: I realize that another student attained a value of $T=7.300865$, yet I cannot account for this unexpected truncation.

<u>Appendix</u>:

1. "f.m"

```
function [y]=f(x,s)
% This function calculates f(x,s).
%
load -ascii data4.txt
xd=data4(:,1);
yd=data4(:,2);
%
%x=[0:0.05:10];
y_sp=spline(xd,yd,x);
y=cos(s*x).*y_sp;
```

"calc_sRK4a.m"

```
% This code uses the composite Simpson rule to calculate
% int_{a}^{b} cos(sx)f2(x) dx.
% The error is calculated using the exact solution.
%
clear
figure(5)
clf reset
axes('position',[0.15,0.13,0.75,0.75])
%
a=0; b=10;
%
s=[0:.03:3];
sizes=101;
%
N=512;
h=(b-a)/N;
x=a+[0:N]*h; x2=a+[0:N-1]*h+h/2;
%
%g=zeros(sizes);
%
disp(' ')
disp(['  The numerical result by the composite Simpson'])
disp(['  rule with N = ',num2str(N),' is'])
%
for i=1:sizes,
    y=f(x,s(i));
    y2=f(x2,s(i));
    g=(y(1)+y(N+1)+2*sum(y(2:N))+4*sum(y2))*h/6;
    disp(['        g(s) = ',num2str(g,'%16.8e'),'.'])
    disp(' ')
    % plot
```

```
    plot(s(i),g,'bo','markerfacecolor','b')
    hold on
end
%
set(gca,'fontsize',12)
axis([0,3,0,2])
set(gca,'xtick',[0:1:3])
%set(gca,'ytick',[-0.5:0.5:1])
title('Composite Simpson Rule calculation of g(s)')
xlabel('s')
ylabel('g(s)')
h1=legend('g(s) = int_{0}^{10} cos(s*x)*f(x) dx');
set(h1,'fontsize',12)
```

2. “f_sys.m”

```
function [z]=f_sys(w,t)
% This function calculates f_sys(w,t)
%
z=zeros(1,2);
theta=w(1);
v=w(2);
z(1)=v;
z(2)=-sin(theta);
```

“calc_sRK4c.m”

```
% This code implements the classical four stage fourth order
% Runge Kutta method to solve an ODE system. After the
% calculation, it saves the workspace to a data file.
%
clear
%
m=2;
w0c=[1.5, 0];
h=0.05;
nstep=20/h;
%
wc=zeros(nstep+1,m);
t=zeros(nstep+1,1);
t(1)=0;
wc(1,1:m)=w0c;
%
p=4;
d=[0,   1/2, 1/2, 1  ];
c=[0,   0,   0,   0  ;
   1/2, 0,   0,   0  ;
   0,   1/2, 0,   0  ;
   0,   0,   1,   0  ];
b=[1/6, 1/3, 1/3, 1/6];
k=zeros(p,m);
%
for j=1:nstep,
  for i=1:p,
    k(i,1:m)=h*f_sys(wc(j,1:m)+c(i,1:i-1)*k(1:i-1,1:m),
t(j)+d(i)*h);
  end
  wc(j+1,1:m)=wc(j,1:m)+b*k;
  t(j+1)=t(j)+h;
end
```

```
%
save data_sRK4c
```

"plot_sRK4.m"

```
% This code reads in the data file generated by calc_sRK4.m.
% Then it plots the two components of the numerical solution:
% position and velocity.
%
clear
figure(3);
clf reset
%
set(gcf,'position',[100,50,500,600])
set(gcf,'paperposition',[0.5,0.5,7.5,10.0])
%
load data_sRK4c
%
axes('position',[0.18,0.56,0.74,0.36])
%
plot(t,wc(1:nstep+1,1),'r--', 'linewidth', 2.0)
%
set(gca,'fontsize',14)
axis([-1,21,-1.6,1.6])
set(gca,'xtick',[0:5:20])
set(gca,'ytick',[-2:1:2])
xlabel('t_{[0,20]}')
ylabel('y_t')
title('Numerical solution by RK4')
h1=legend('y_0=1.5');
set(h1,'fontsize',12)
```

"dd.m"

```
function [dist]=dd(T,td,yd)
%
t1=[0:0.1:8];
y1=spline(td,yd,t1);
t2=T+t1;
y2=spline(td,yd,t2);
dist=norm(y1-y2);
```

"calc_d_T.m"

```
clear
%
load data_sRK4c
%
T=[6:3/100:9];
sizeT=101;
dd_T=zeros(1,sizeT);
%
for k=1:sizeT,
  dd_T(k)=dd(T(k),t,wc(1:nstep+1,1));
end
%
save data_d_T
```

"plot_d_T.m"

```
clear
figure(4)
```

```
clf reset
axes('position',[0.15,0.13,0.75,0.75])
%
load data_d_T
%
plot(T,dd_T,'linewidth',2.0)
axis([5.9,9.1,-.5,14])
set(gca,'fontsize',14)
set(gca,'xtick',[6:9])
%set(gca,'ytick',[0:5:15])
xlabel('T')
ylabel('dd(T)')
title('Possible values of period T for y``+sin(y)=0, t=[0,20]')
for i=1:sizeT,
    disp(['        T = ',num2str(T(i),'%16.8e'),'.'])
    disp(['        dd(T) = ',num2str(dd_T(i),'%16.8e'),'.'])
    disp(' ')
end
```

3. “golden_search.m”

```
clear
%
load data_sRK4c
td=t;
yd=wc(1:nstep+1,1);
%
a=6;
b=9;
tol=1.0e-10;
n=0;
%
g=(sqrt(5)-1)/2;
r1=a+(b-a)*(1-g);
f1=dd(r1,td,yd);
r2=a+(b-a)*g;
f2=dd(r2,td,yd);
%
while (b-a) > tol,
  n=n+1;
  if f1 < f2,
    b=r2;
    r2=r1;
    f2=f1;
    r1=a+(b-a)*(1-g);
    f1=dd(r1,td,yd);
  else
    a=r1;
    r1=r2;
    f1=f2;
    r2=a+(b-a)*g;
    f2=dd(r2,td,yd);
  end
end
T=(a+b)/2
```