Problem Set 11 – Due Dec 4 at Midnight.

Submit to the course website. Make sure your name is in the file header. Note any collaborations in the header as well. Each student must turn in their own set of solutions. This is the last graded PSET!

11_1 – Question 22.2 – for each, write your own iterative algorithm (do not use Chapra or Matlab functions)

22.2 Solve the following problem over the interval from $x = 0$ to 1 using a step size of 0.25 where $y(0) = 1$. Display all your results on the same graph.

$$\frac{dy}{dx} = (1 + 4x)\sqrt{y}$$

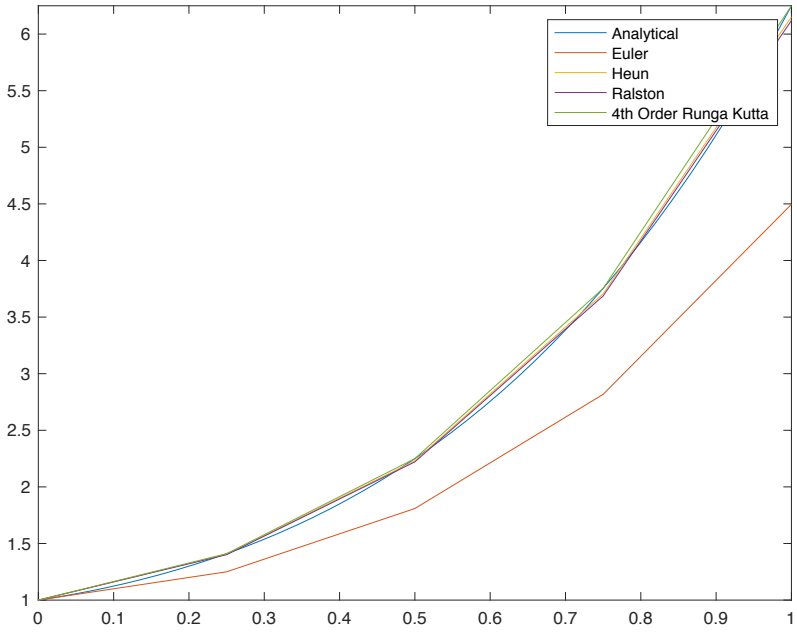
(a) Analytically.
(b) Using Euler's method.
(c) Using Heun's method without iteration.
(d) Using Ralston's method.
(e) Using the fourth-order RK method.

Solution:

```
% PSET11_1
% Coded by Nigel F. Reuel on 11.18.2016
% This code solves problem 22.2
%
% Part A)
% Separation of variables yields (y)^(-1/2)dy = (1+4x)dx.  Integrating both
% sides will give you 2*y^(1/2) = x+2*x^2+C.  We know that y(0) = 1, so C
% must = 2.  Let's make the answer an anonymous function
Y1 = @(x) ((x+2*x^2+2)/2)^2;
fplot(Y1,[0 1])
hold on
%
% I am going to do parts B, C, and D all in the same loop
h = 0.25;
t = (0:h:1)'; n = length(t);
y0 = 1;
dydt = @(x,y) (1+4.*x).*(y).^(1/2);
% Preallocate memory
yb = y0*ones(n,1); %y for part b euler method
yc = y0*ones(n,1); %y for part c Heun method
yd = y0*ones(n,1); %y for part d ralston method
for i = 1:n-1 %implement Euler's method
    yb(i+1) = yb(i) + dydt(t(i),yb(i))*(t(i+1)-t(i)); % <-- Euler
    % ----
    ycnext = yc(i) + dydt(t(i),yc(i))*(t(i+1)-t(i)); % <-- Next step for Heun
    SlopeAvg = (dydt(t(i),yc(i))+dydt(t(i+1),ycnext))/2; % Avg slope for Heun
    yc(i+1) = yc(i) + SlopeAvg*(t(i+1)-t(i)); % Calculate next step for Heun
    %-----
    k1 = dydt(t(i),yd(i));
    k2 = dydt(t(i)+3/4*h,yd(i)+3/4*k1*h);
    yd(i+1) = yd(i)+ (1/3*k1+2/3*k2)*h;
end
plot(t,yb,t,yc,t,yd)
% Part (e) 4th order Runga Kutta Model
[tp,yp] = rk4sys(dydt,[0 1],y0,h);
plot(tp,yp)
legend('Analytical','Euler','Heun','Ralston','4th Order Runga Kutta')
```

```
function [tp,yp] = rk4sys(dydt,tspan,y0,h,varargin)
% rk4sys: fourth-order Runge-Kutta for a system of ODEs
% [t,y] = rk4sys(dydt,tspan,y0,h,p1,p2,...): integrates
% a system of ODEs with fourth-order RK method
% input:
% dydt = name of the M-file that evaluates the ODEs
% tspan = [ti, tf]; initial and final times with output
% generated at interval of h, or
% = [t0 t1 ... tf]; specific times where solution output
% y0 = initial values of dependent variables
% h = step size
% p1,p2,... = additional parameters used by dydt
% output:
% tp = vector of independent variable
% yp = vector of solution for dependent variables
if nargin<4,error('at least 4 input arguments required'), end
if any(diff(tspan)<=0),error('tspan not ascending order'), end
n = length(tspan);
ti = tspan(1);tf = tspan(n);
if n == 2
t = (ti:h:tf)'; n = length(t);
if t(n)<tf
t(n+1) = tf;
n = n+1;
end
else
t = tspan;
end
tt = ti; y(1,:) = y0;
np = 1; tp(np) = tt; yp(np,:) = y(1,:);
i=1;
while(1)
tend = t(np+1);
hh = t(np+1) - t(np);
if hh>h,hh = h;end
while(1)
if tt+hh>tend,hh = tend-tt;end
k1 = dydt(tt,y(i,:),varargin{:})';
ymid = y(i,:) + k1.*hh./2;
k2 = dydt(tt+hh/2,ymid,varargin{:})';
ymid = y(i,:) + k2*hh/2;
k3 = dydt(tt+hh/2,ymid,varargin{:})';
yend = y(i,:) + k3*hh;
k4 = dydt(tt+hh,yend,varargin{:})';
phi = (k1+2*(k2+k3)+k4)/6;
y(i+1,:) = y(i,:) + phi*hh;
tt = tt+hh;
i=i+1;
if tt>=tend,break,end
end
np = np+1; tp(np) = tt; yp(np,:) = y(i,:);
if tt>=tf,break,end
end
end
```

Output:



11_2 – Question 22.18. Plot all four dependent variables versus time. At what time does CB2 = 5?

22.18 The reaction $A \rightarrow B$ takes place in two reactors in series. The reactors are well mixed but are not at steady state. The unsteady-state mass balance for each stirred tank reactor is shown below:

$$\frac{dCA_1}{dt} = \frac{1}{\tau}(CA_0 - CA_1) - kCA_1$$

$$\frac{dCB_1}{dt} = -\frac{1}{\tau}CB_1 + kCA_1$$

$$\frac{dCA_2}{dt} = \frac{1}{\tau}(CA_1 - CA_2) - kCA_2$$

$$\frac{dCB_2}{dt} = \frac{1}{\tau}(B_1 - CB_2) + kCB_2$$

where CA_0 = concentration of A at the inlet of the first reactor, CA_1 = concentration of A at the outlet of the first reactor (and inlet of the second), CA_2 = concentration of A at the outlet of the second reactor, CB_1 = concentration of B at the outlet of the first reactor (and inlet of the second), CB_2 = concentration of B in the second reactor, τ = residence time for each reactor, and k = the rate constant for reaction of A to produce B. If CA_0 is equal to 20, find the concentrations of A and B in both reactors during their first 10 minutes of operation. Use $k = 0.12$/min and $\tau = 5$ min and assume that the initial conditions of all the dependent variables are zero.

Solution:

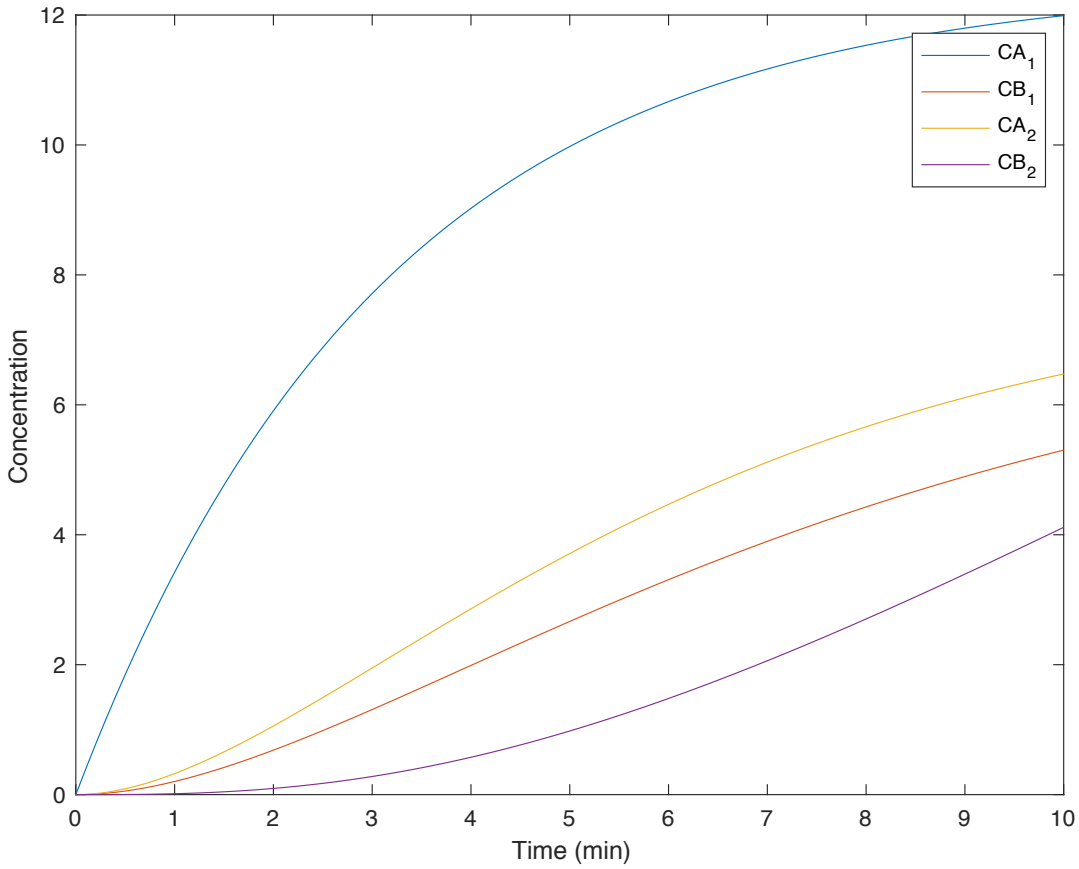
```
% PSET11_2
% Coded by NFR on 11.18.2016
% This code solves problem 22.18 from the textbook
%
% Setup all of the rate equations as a sub function.  Then use the rk4sys
[tp,yp] = rk4sys(@sys,[0 10],[0 0 0 0],0.1);
plot(tp,yp)
xlabel('Time (min)')
ylabel('Concentration')
legend('CA_1','CB_1','CA_2','CB_2')


function dy = sys(t,y)
% y(1) = A1
% y(2) = B1
% y(3) = A2
% y(4) = B2
%
k = 0.12;
tao = 5;
Cao = 20;
dy = [1/tao*(Cao-y(1))-k*y(1);
    -1/tao*y(2)+k*y(1);
    1/tao*(y(1)-y(3))-k*y(3);
    1/tao*(y(2)-y(4))+k*y(4)];
end

function [tp,yp] = rk4sys(dydt,tspan,y0,h,varargin)
% rk4sys: fourth-order Runge-Kutta for a system of ODEs
% [t,y] = rk4sys(dydt,tspan,y0,h,p1,p2,...): integrates
% a system of ODEs with fourth-order RK method
% input:
% dydt = name of the M-file that evaluates the ODEs
% tspan = [ti, tf]; initial and final times with output
% generated at interval of h, or
% = [t0 t1 ... tf]; specific times where solution output
% y0 = initial values of dependent variables
% h = step size
% p1,p2,... = additional parameters used by dydt
% output:
% tp = vector of independent variable
% yp = vector of solution for dependent variables
if nargin<4,error('at least 4 input arguments required'), end
if any(diff(tspan)<=0),error('tspan not ascending order'), end
```

```
n = length(tspan);
ti = tspan(1);tf = tspan(n);
if n == 2
t = (ti:h:tf)'; n = length(t);
if t(n)<tf
t(n+1) = tf;
n = n+1;
end
else
t = tspan;
end
tt = ti; y(1,:) = y0;
np = 1; tp(np) = tt; yp(np,:) = y(1,:);
i=1;
while(1)
tend = t(np+1);
hh = t(np+1) - t(np);
if hh>h,hh = h;end
while(1)
if tt+hh>tend,hh = tend-tt;end
k1 = dydt(tt,y(i,:),varargin{:})';
ymid = y(i,:) + k1.*hh./2;
k2 = dydt(tt+hh/2,ymid,varargin{:})';
ymid = y(i,:) + k2*hh/2;
k3 = dydt(tt+hh/2,ymid,varargin{:})';
yend = y(i,:) + k3*hh;
k4 = dydt(tt+hh,yend,varargin{:})';
phi = (k1+2*(k2+k3)+k4)/6;
y(i+1,:) = y(i,:) + phi*hh;
tt = tt+hh;
i=i+1;
if tt>=tend,break,end
end
np = np+1; tp(np) = tt; yp(np,:) = y(i,:);
if tt>=tf,break,end
end
end
```

Output:


12
10
8
6
4
2
0
0
1
2
3
4
5
6
7
8
9
10
Concentration
Time (min)
CA1
CB1
CA2
CB2

11_3 – Question 22.19. Let's assume the time unit is hours. Please plot temp and concentration from 0 to 5 hours. Use subplots. What is the max reactor temperature? When is 95% conversion achieved?

22.19 A nonisothermal batch reactor can be described by the following equations:

$$\frac{dC}{dt} = -e^{(-10/(T+273))}C$$

$$\frac{dT}{dt} = 1000e^{(-10/(T+273))}C - 10(T - 20)$$

where C is the concentration of the reactant and T is the temperature of the reactor. Initially, the reactor is at 16 °C and has a concentration of reactant C of 1.0 gmol/L. Find the concentration and temperature of the reactor as a function of time.

Solution:

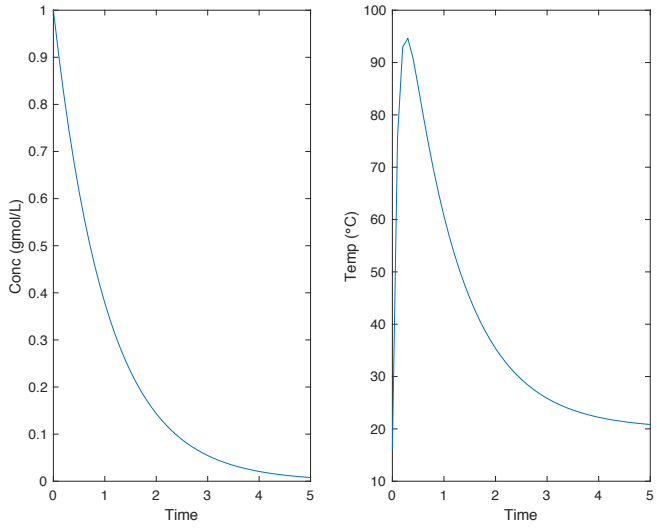
```
% PSET11_3
% Coded by Nigel F. Reuel on 11.18.2016
% This code solves problem 22.19 from the book
%
[tp,yp] = rk4sys(@sys,[0 5],[1 16],0.1);
subplot(1,2,1)
plot(tp,yp(:,1))
xlabel('Time')
ylabel('Conc (gmol/L)')
subplot(1,2,2)
plot(tp,yp(:,2))
xlabel('Time')
ylabel('Temp (\circC)')
% Add more here?  It would be interesting to figure out what TIME it hits
% 99% depletion of the reactant.


function dy = sys(t,y)
% y(1) = C
% y(2) = T
%
dy = [-exp(-10/(y(2)+273))*y(1);
    1000*exp(-10/(y(2)+273))*y(1)-10*(y(2)-20)];
end

function [tp,yp] = rk4sys(dydt,tspan,y0,h,varargin)
% rk4sys: fourth-order Runge-Kutta for a system of ODEs
% [t,y] = rk4sys(dydt,tspan,y0,h,p1,p2,...): integrates
% a system of ODEs with fourth-order RK method
% input:
% dydt = name of the M-file that evaluates the ODEs
% tspan = [ti, tf]; initial and final times with output
% generated at interval of h, or
% = [t0 t1 ... tf]; specific times where solution output
% y0 = initial values of dependent variables
% h = step size
% p1,p2,... = additional parameters used by dydt
% output:
% tp = vector of independent variable
% yp = vector of solution for dependent variables
```

```
if nargin<4,error('at least 4 input arguments required'), end
if any(diff(tspan)<=0),error('tspan not ascending order'), end
n = length(tspan);
ti = tspan(1);tf = tspan(n);
if n == 2
t = (ti:h:tf)'; n = length(t);
if t(n)<tf
t(n+1) = tf;
n = n+1;
end
else
t = tspan;
end
tt = ti; y(1,:) = y0;
np = 1; tp(np) = tt; yp(np,:) = y(1,:);
i=1;
while(1)
tend = t(np+1);
hh = t(np+1) - t(np);
if hh>h,hh = h;end
while(1)
if tt+hh>tend,hh = tend-tt;end
k1 = dydt(tt,y(i,:),varargin{:})';
ymid = y(i,:) + k1.*hh./2;
k2 = dydt(tt+hh/2,ymid,varargin{:})';
ymid = y(i,:) + k2*hh/2;
k3 = dydt(tt+hh/2,ymid,varargin{:})';
yend = y(i,:) + k3*hh;
k4 = dydt(tt+hh,yend,varargin{:})';
phi = (k1+2*(k2+k3)+k4)/6;
y(i+1,:) = y(i,:) + phi*hh;
tt = tt+hh;
i=i+1;
if tt>=tend,break,end
end
np = np+1; tp(np) = tt; yp(np,:) = y(i,:);
if tt>=tf,break,end
end
end
```

Output:



11_4. Solve Chapra problem 24.11. Plot the concentration of A as a function of distance.

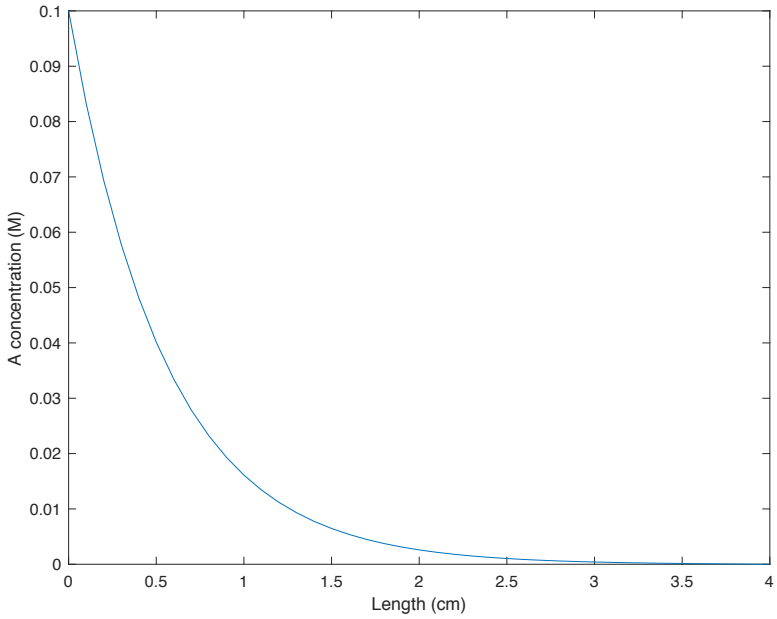
> **24.11** Compound A diffuses through a 4-cm-long tube and reacts as it diffuses. The equation governing diffusion with reaction is
>
> $$D\frac{d^2A}{dx^2} - kA = 0$$
>
> At one end of the tube ($x = 0$), there is a large source of A that results in a fixed concentration of 0.1 M. At the other end of the tube there is a material that quickly absorbs any A, making the concentration 0 M. If $D = 1.5 \times 10^{-6}$ cm^2/s and $k = 5 \times 10^{-6}$ s^{-1}, what is the concentration of A as a function of distance in the tube?

Solution:

```
% Coded by NFR on 12.5.2016
% This code solves problem 24.11 from the textbook
%
% Use shooting method, two shots because the problem is linear.
% 2017 comment - or you can use fzero or fminsearch to find the unknown
% transformed variable.
A0 = 0.1/1000; %mol/cm^3
zg1 = -.001; %First guess for z(0)
y0 = [A0 zg1];
span = [0 4];
[tp,yp] = ode45(@dydt,span,y0);
hend1 = yp(end,1);
zg2 = .001; %Second guess for z(0)
y0 = [A0 zg2];
span = [0 4];
[tp,yp] = ode45(@dydt,span,y0);
hend2 = yp(end,1);
% Interpolate between the two to find the correct z(0) that gives us the
% desired end concentration of 0 mol/cm^3
EndDesired = 0;
Slope = (zg2-zg1)/(hend2-hend1);
zcorrect = Slope*(EndDesired-hend1)+zg1;
% Use the correct z value to solve and plot h vs. x:
y0 = [A0 zcorrect];
[tp,yp] = ode45(@dydt,span,y0);
plot(tp,yp(:,1)*1000) %NOTE: converted units back to Molar
xlabel('Length (cm)')
ylabel('A concentration (M)')

function dy = dydt(t,y)
% t = x, y(1) = A, y(2) = z;
k = 5*10^-6; %-s
D = 1.5*10^-6; %cm^2/s
dy = [y(2);
    k*y(1)/D];
end
```

Output:


A concentration (M)
Length (cm)
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.5
1
1.5
2
2.5
3
3.5
4

11_5. Solve Chapra problem 24.18 (problem statement on next page

24.18 Under a number of simplifying assumptions, the steady-state height of the water table in a one-dimensional. unconfined groundwater aquifer (Fig. P24.18) can be mod- eled with the following second-order ODE:

$$Kh\frac{-d^2h}{dp2} \quad N = 0$$

where z distance (m), K — hydraulic conductivity (m/d), h — height of the water table (m), h — the average height of the water table (m), and N infiltration rate (m/d).

Solve for the height of the water table for x — 0 to 1000 m where â(0) - 10 m and /t(1000) - 5 m. Use the following parameters for the calculation: N = 1 m/d and N 0.0001 m/d. Set the average height of the water table as the average of the boundary conditions. Obtain your solution with (a) the shooting method and (b) the finite-difference method (As = 100 m).

Solution:

```
% Coded by Nigel F. Reuel on 12.2.2016
% This solves problem 24.18 in the book
%
% Part (A)
% This equation is linear, so we can do two shots and then interpolate.
zg1 = -1; %First guess for z(0)
y0 = [10 zg1];
span = [0 1000];
[tp,yp] = ode45(@dydt,span,y0);
hend1 = yp(end,1);
zg2 = 1; %Second guess for z(0)
y0 = [10 zg2];
span = [0 1000];
[tp,yp] = ode45(@dydt,span,y0);
hend2 = yp(end,1);
% Interpolate between the two to find the correct z(0) that gives us the
% desired end height of h(1000) = 5m; This is one way to do interpolation:
EndDesired = 5;
Slope = (zg2-zg1)/(hend2-hend1);
zcorrect = Slope*(EndDesired-hend1)+zg1
% Another way to do interpolation (with interp1)
zcorrect = interp1([hend1 hend2],[zg1 zg2],EndDesired)
% Use the correct z value to solve and plot h vs. x:
y0 = [10 zcorrect];
[tp,yp] = ode45(@dydt,span,y0);
plot(tp,yp(:,1))
% Part B - see paper work for derivation of tridiagonal matrix
%  2019: NOTE this uses Chapra's tridiag function that requires the leading
%  and trailing zeros.
e = ones(1,9);
e(1) = 0;
g = ones(1,9);
g(9) = 0;
f = ones(1,9)*-2;
r = ones(1,9)*-0.1333;
```

```
r(1) = -10.1333;
r(9) = -5.13333;
hvec = Tridiag(e,f,g,r);
% Add the end values ot your internal points:
hvec = [10 hvec 5];
xvec = 0:100:1000;
hold on
plot(xvec,hvec,'o')
xlabel('X distance (m)')
ylabel('Water table height (m)')


function x = Tridiag(e,f,g,r)
% Tridiag: Tridiagonal equation solver banded system
% x = Tridiag(e,f,g,r): Tridiagonal system solver.
% input:
% e = subdiagonal vector
% f = diagonal vector
% g = superdiagonal vector
% r = right hand side vector
% output:
% x = solution vector
n=length(f);
% forward elimination
for k = 2:n
factor = e(k)/f(k-1);
f(k) = f(k) - factor*g(k-1);
r(k) = r(k) - factor*r(k-1);
end
% back substitution
x(n) = r(n)/f(n);
for k = n-1:-1:1
x(k) = (r(k)-g(k)*x(k+1))/f(k);
end
end

function dy = dydt(t,y)
% t = x, y(1) = h, y(2) = z
K = 1;
N = 0.0001;
hbar = 7.5; % Average of the boundary conditions; Is this right?  Seems like
this should be calculated after the fact.
dy = [y(2);
    -N/(K*hbar)];
end
```

Output:

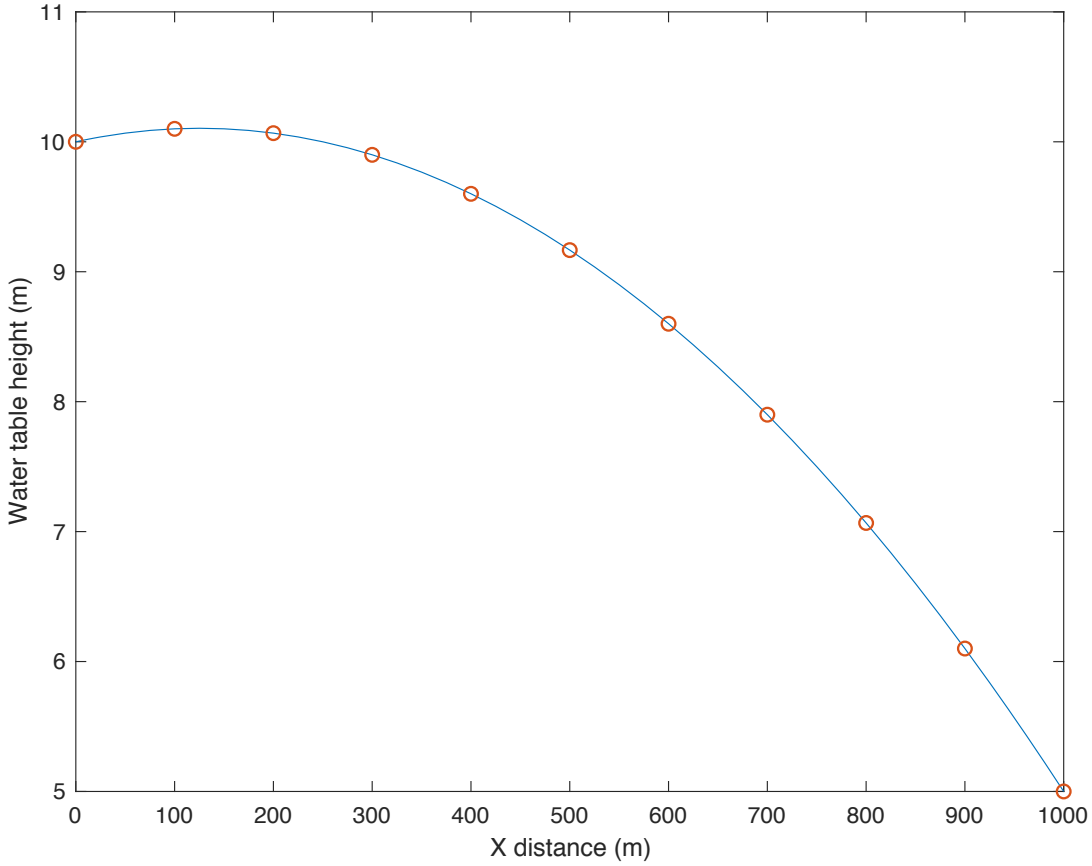
```
>> PSET11_5

zcorrect =

   0.0017


zcorrect =
```

0.0017


11
10
9
8
7
6
5
Water table height (m)
0
100
200
300
400
500
600
700
800
900
1000
X distance (m)