This code calculates the sum of squared numbers from 1 to “n”.

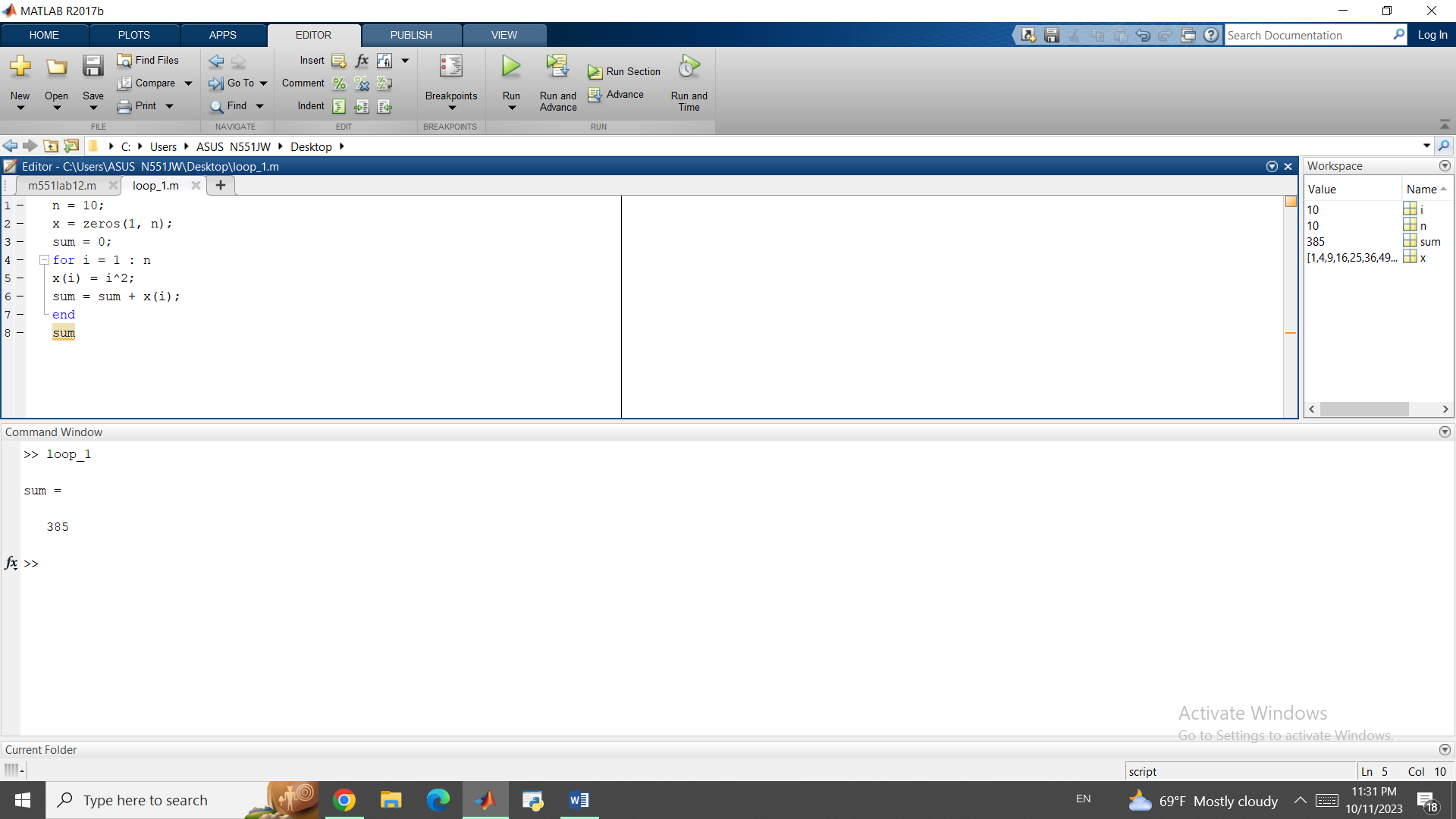
In the given code, n = 10 so:

**sum =**

if we want to explain it step by step with more details, we can say it assigns value of 10 to n and then creates a vector of length n named x.

sets initial value of sum to be zero and then creates a loop which iterates from i=1 to i=n and in each step assigns value of ith element of vector x to be equal to i2 and adds this value to be saved in variable sum.

Finally, the final value of sum will be shown as the output of this code which will be the sum of the squares of the numbers from 1 to n.



We know the numerical formula for the Euler method is:

yn+1 = yn + h\*f(xn,yn)

coming back to the provided code,we have to complete two sections.

We have to complete the exact solution as

y3 = 2\*exp(x3)-x3-1

and then we have complete the loop which is basically the formula for the Euler method:

%%% Euler's method

function [x,y] = Euler(a,b,y0,N)

f = @(x,y) x+y;

h = (b-a)/N;

x = a:h:b;

y = zeros(1,N+1);

x(1) = a;

y(1) = y0;

%%% main loop for Euler's method

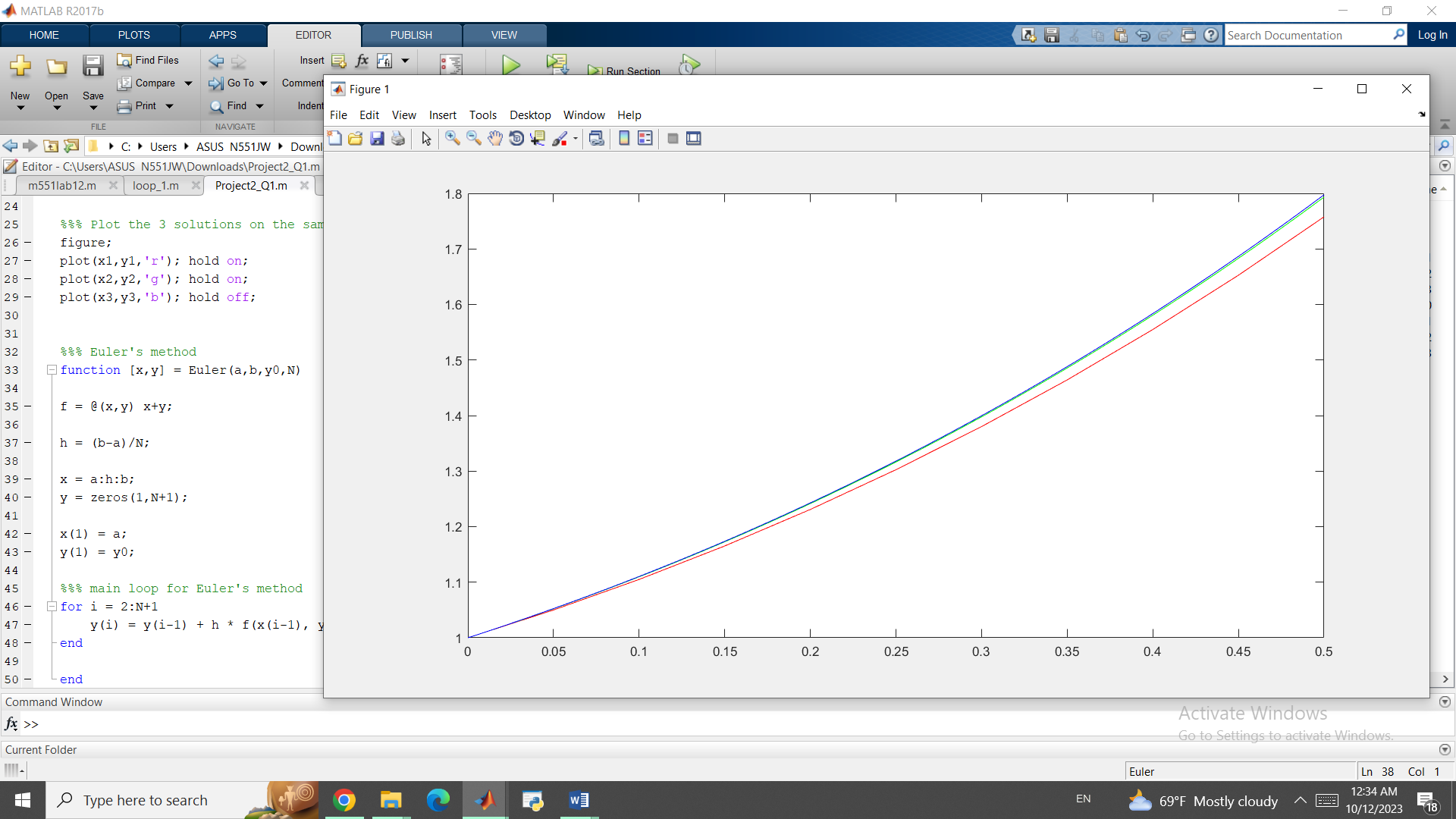
for i = 2:N+1

y(i) = y(i-1) + h \* f(x(i-1), y(i-1));%%% enter commands here so that the loop computes the approximate solutions %%%

end

end

After running the code, we get the following outcome:



close all;

clear all;

%%% Define the interval [a,b] and initial condition

a = 0;

b = 0.5;

y0 = 1;

%%% Find the approximated solution using Euler's method

N = 10; %%% number of steps

[x1,y1] = Euler(a,b,y0,N); %%% use the function "Euler" to find the approximate solution with step size h=0.05

N = 100; %%% number of steps

[x2,y2] = Euler(a,b,y0,N); %%% use the function "Euler" to find the approximate solution with step size h=0.005

%%% Define the exact solution

N = 100;

x3 = a:(b-a)/N:b;

y3 = 2\*exp(x3)-x3-1; %%%enter the exact solution here%%%

%%% Plot the 3 solutions on the same figure

figure;

plot(x1,y1,'r'); hold on;

plot(x2,y2,'g'); hold on;

plot(x3,y3,'b'); hold off;

%%% Euler's method

function [x,y] = Euler(a,b,y0,N)

f = @(x,y) x+y;

h = (b-a)/N;

x = a:h:b;

y = zeros(1,N+1);

x(1) = a;

y(1) = y0;

%%% main loop for Euler's method

for i = 2:N+1

y(i) = y(i-1) + h \* f(x(i-1), y(i-1));%%% enter commands here so that the loop computes the approximate solutions %%%

end

end

First, we correct the value of f (x, y) in the IVP to:

%%% Define f(x,y) in the IVP

f = @(x,y) 2\*x+y;

now we start the theoretical part. Since we have a first order IVP:

y’ – y = 2x

Homogeneous part:

λ -1 = 0 so λ = 1 and yh = Cex where C is a constant.

Particular part:

y = ax+b so y’ = a and substituting into the equation:

a-(ax+b) = 2x so a=b=-2 and yp = -2x-2

y = yp + yh so:

**y = Cex -2x -2**

Now we insert the given initial values for each section to determine the value of coefficient “C”:

In part a, y (0) = -2 so:

-2 = C-2 thus C=0 and

**y1 = -2x-2**

2b.

Here, y (-1) = 3 so:

3 = Ce-1 +2 -2 and thus C = 3e so:

**y2 = 3ex+1 -2x -2**

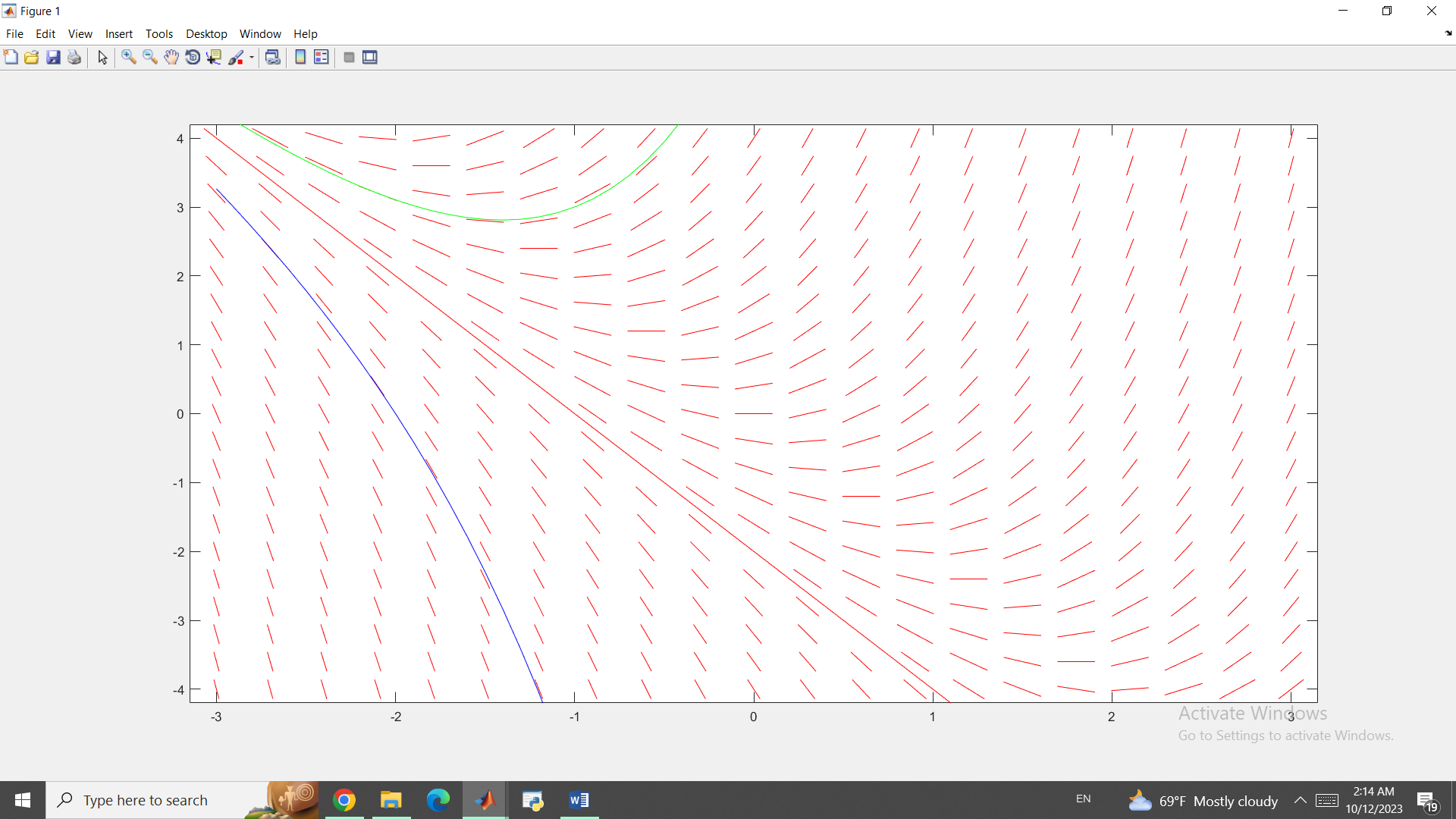
2c.

For the last part, y (-2) = 0 so:

0 = Ce-2 +4 -2 and thus C = -2e2 so:

**y3 = -2ex+2 -2x -2**

if we plot these 3 signals on the given interval, we will see the following figure:



The result for partb,y2, goes to Infinity whenever x→ or x→

Since:

=

And also:

Thus, y2 → whenever x →

clear all;

%%% Define f(x,y) in the IVP

f = @(x,y) 2\*x+y;

%%% Draw the direction field

figure;

dirfield(f, -3:.3:3, -4:.4:4);hold on;

%%% Define the solutions to the IVP in Q2(a)-(c)

x = -3:0.1:3;

y1 = -2.\*x-2;

y2 = 3.\*exp(x+1)-2.\*x-2;

y3 = -2.\*exp(x+2)-2.\*x-2;

%%% Plot the 3 solutions on the direction field

plot(x,y1,'r');hold on;

plot(x,y2,'g');hold on;

plot(x,y3,'b');hold off;

function dirfield(f,tval,yval)

% dirfield(f, t1:dt:t2, y1:dy:y2)

%

% plot direction field for first order ODE y' = f(t,y)

% using t-values from t1 to t2 with spacing of dt

% using y-values from y1 to t2 with spacing of dy

%

% f is an @ function, or an inline function,

% or the name of an m-file with quotes.

%

% Example: y' = -y^2 + t

% Show direction field for t in [-1,3], y in [-2,2], use

% spacing of .2 for both t and y:

%

% f = @(t,y) -y^2+t

% dirfield(f, -1:.2:3, -2:.2:2)

[tm,ym]=meshgrid(tval,yval);

dt = tval(2) - tval(1);

dy = yval(2) - yval(1);

fv = vectorize(f);

if isa(f,'function\_handle')

fv = eval(fv);

end

yp=feval(fv,tm,ym);

s = 1./max(1/dt,abs(yp)./dy)\*0.35;

h = ishold;

quiver(tval,yval,s,s.\*yp,0,'.r'); hold on;

quiver(tval,yval,-s,-s.\*yp,0,'.r');

if h

hold on

else

hold off

end

axis([tval(1)-dt/2,tval(end)+dt/2,yval(1)-dy/2,yval(end)+dy/2])