t0=0; %initial t-value

tl=2; %last t-value

y0=1; %initial y-value

dstep=.5; %we collect the data only for a small number of t-values

stepsE=[.01,.001]; %this is the vector of steps for Euler method

stepsIE=[.025,.01]; %this is the vector of steps for improved Euler method

tvals=[t0:dstep:tl]; %vector of t-values for table

%calculating the exact values for our display t-values

yexact=-3/16+tvals/4+19/16\*exp(4\*tvals); %calculates the exact y-values for

%the vector of t-values

output=[tvals;yexact];

%calculating values for display t-values & step sizes using Euler

for j=1:length(stepsE)

h=stepsE(j); %fixes the step

t=t0; %initializes t-var

y=y0; %initializes y-var

yapprox=[y]; %initializes the vector of y-values

%we start the loop central to Euler’s method

for i=1:(tl-t0)/h %# of iterations is: (last t minus first t)/step

 f=1-t+4\*y; %yprime, an expression gotten from the ODE

 %Note that it is calculated before incrementing t

 t=t+h; %incrementing t

 y=y+f\*h; %calculating new y-value using tangent line approximation

 if rem((t-t0)+.0000001,dstep)<0.00001 %tests to see if we have

%reached a display step. “fudge” factors

% are used due to rounding issues.

yapprox=[yapprox,y]; %appends the new y to

 %vector of y-values.

 end

end

output=[output;yapprox];

end

%calculating values for display t-values & step sizes using Improved Euler

for j=1:length(stepsIE)

h=stepsIE(j); %fixes the step

t=t0; %initializes t-var

y=y0; %initializes y-var

yapprox=[y]; %initializes the vector of y-values

%we start the loop central to Euler’s method

for i=1:(tl-t0)/h %# of iterations is: (last t minus first t)/step

 f1=1-t+4\*y; %yprime for left pt, an expression gotten from the ODE

 %Note that it is calculated before incrementing t

 t=t+h; %incrementing t

 f2=1-t+4\*(y+f1\*h);%yprime for right pt, an expression gotten from the ODE

 y=y+h/2\*(f1+f2); %calculating new y-value using improved euler

 if rem((t-t0)+.0000001,dstep)<0.00001 %tests to see if we have

%reached a display step. “fudge” factors

% are used due to rounding issues.

yapprox=[yapprox,y]; %appends the new y to

 %vector of y-values.

 end

end

output=[output;yapprox];

end

%hack job for displaying the output

fprintf([' ','euler',' ','improved euler','\n']);

fprintf([' t ','exact y ',num2str(stepsE),' ',num2str(stepsIE),'\n']);

disp(output')