MATH 366 – Methods of Applied Mathematics II Clayton Johnson

HW6 : Ch. 21.1 Heun and Runge-Kutta Methods with MATLAB

Huen MATLAB Code

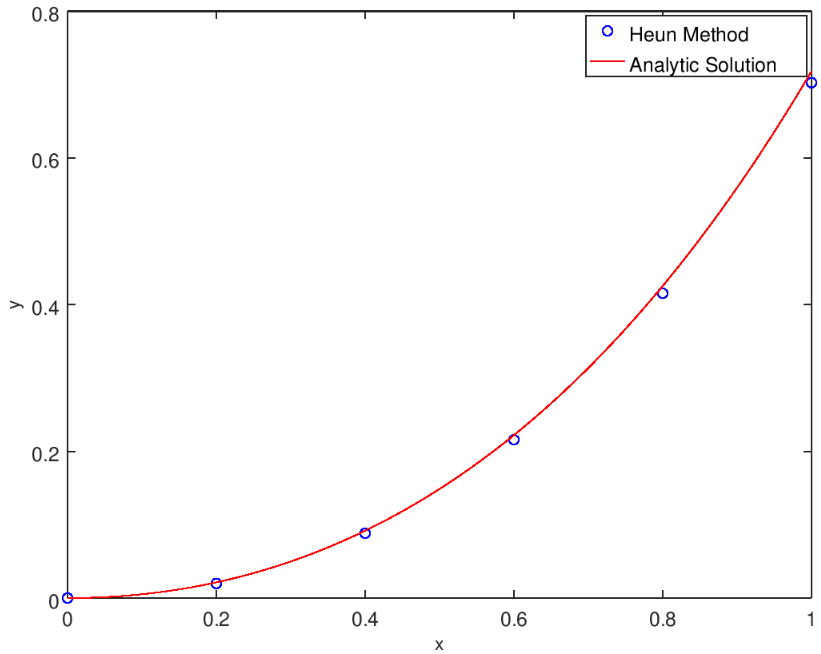
```
function heun211(x0,y0,h,N)

%Heun's Method from Chapter 21.1 (hence heun211).

%This method is used to solve y'=f(x,y), y(x0)= y0.

%f(x,y) is typed into the program code.

%The function inputs needed are

%%%% x0 = initial x-value

%%%% y0 = initial y-value

%%%% h is the step size for the x-nodes

%%%% N is the number of iterations

%If the exact solution g(x) is known in advance, then

%this program plots it along with the numerical solution.

%This works best if g is only a function of x.


%Derivative function

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


%Analytic (exact) solution of IVP for this problem.

%This g(x) is given on page 10 of our book for

%this f(x,y) and (x0,y0) = (0,0).

g = @(x) exp(x) - x - 1;


%Initialize x & y vectors

x = zeros(N+1,1);

y = zeros(N+1,1);
```

```
%Initial conditions
x(1) = x0;
y(1) = y0;

%Heun's Method
for n = 1:N+1;
  x(n+1) = x(n) + h;           % Next x-value
  k1 = h*f(x(n), y(n));        % k1 uses the current slope (this is the predictor)
  k2 = h*f(x(n+1), y(n)+k1);    % k2 uses the next slope (the predictor's slope)
  y(n+1) = y(n) + 0.5*(k1+k2);  % Next y-value uses the avg b/t the two slopes/ the corrector

  S(n) = n-1;       %Records step number for display in matrix R.
  X(n) = x(n);      %Records x value at step n
  Y(n) = y(n);      %Records y value of numerical solution at step n
  G(n) = g(x(n));   %Records y value of analytical solution at step n
end

%We now display results from above as columns of the matrix R.
%Use space key to position headers by trial & error after running program
%The "%5.4f" below specifies decimal format (the f part), with
%5 digits total, 4 digits to right of decimal point.
R=[S' X' Y' G']';  %Results matrix whose columns are S', X', etc.
fprintf(' n  x(n)  y(n)  Exact \n');  %These are the column headers
fprintf('%2d %2.2f %5.4f  %5.4f \n',R); %These adjust decimal formats

%Plot numerical solution y(n)
figure(1); clf(1)  %Set up new figure and clear previous figure
plot(x(1:N+1),y(1:N+1),'bo')  %Plot (x(n),y(n)) points using blue "o"
```

```
L = N*h;  %[0,L] is the x-axis interval for solution, used in linspace
%Plot analytic solution g(x) on 1000 x-values on [0,L]
hold all %retains current plots so new ones don't delete existing ones
xvalues = linspace(0,L,1000); %1000 equally spaced x-values from 0 to L
plot(xvalues,g(xvalues),'r') %Sample g on xvalues & plot in red

%Annotated plot
xlabel('x')
ylabel('y')
k = legend('Heun Method','Analytic Solution', 'Location','best');
set(k,'fontsize',12);
set(gca,'fontsize',12)  %gca = "get current axis" being used
```

Huen MATLAB Output

```
>> HW6heun(0, 0, 0.2, 5)
n  x(n)    y(n)      Exact
0  0.00   0.0000   0.0000
1  0.20   0.0200   0.0214
2  0.40   0.0884   0.0918
3  0.60   0.2158   0.2221
4  0.80   0.4153   0.4255
5  1.00   0.7027   0.7183
```



Runge-Kutta MATLAB Code

```
function runge211(x0,y0,h,N)

%Runge-Kutta Method from Chapter 21.1 (hence runge211).

%This method is used to solve y'=f(x,y), y(x0)= y0.

%f(x,y) is typed into the program code.

%The function inputs needed are

%%%% x0 = initial x-value

%%%% y0 = initial y-value

%%%% h is the step size for the x-nodes

%%%% N is the number of iterations

%If the exact solution g(x) is known in advance, then

%this program plots it along with the numerical solution.


%Derivative function

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

```
%Analytic (exact) solution of IVP for this problem.
%This works best if g is only a function of x.
%This g(x) is given on page 10 of our textbook.
g = @(x) exp(x) - x - 1;

%Initialize x & y vectors for numerical solution
x = zeros(N+1,1);
y = zeros(N+1,1);

%Initial conditions
x(1) = x0;
y(1) = y0;

%Runge-Kutta Method
for n = 1:N+1;
  x(n+1) = x(n) + h;
  k1 = h*f(x(n), y(n));
  k2 = h*f(x(n)+0.5*h, y(n)+0.5*k1);
  k3 = h*f(x(n)+0.5*h, y(n)+0.5*k2);
  k4 = h*f(x(n+1), y(n)+k3);
  y(n+1) = y(n) + (1/6)*(k1 + 2*k2 + 2*k3 + k4);

  S(n) = n-1;       %Records step number for display in matrix R.
  X(n) = x(n);      %Records x value at step n
  Y(n) = y(n);      %Records y value of numerical solution at step n
  G(n) = g(x(n));   %Records y value of analytical solution at step n
end

%We now display results from above as columns of the matrix R.
```

```
%Use space key to position headers by trial & error after running program
%The "%7.6f" below specifies decimal format (the f part), with
%7 digits total, 6 digits to right of decimal point.
R=[S' X' Y' G']';  %Results matrix whose columns are S', X', etc.
fprintf(' n  x(n)   y(n)     Exact \n');  %These are the column headers
fprintf('%2d %2.2f %7.6f  %7.6f \n',R); %These adjust decimal formats

%Plot numerical solution y(n)
figure(1); clf(1)  %Set up new figure and clear previous figure
plot(x(1:N+1),y(1:N+1),'bo')  %Plot (x(n),y(n)) points using blue "o"

L = N*h;  %[0,L] is the x-axis interval for solution, used in linspace
%Plot analytic solution g(x) on 1000 x-values on [0,L]
hold all %retains current plots so new ones don't delete existing ones
xvalues = linspace(0,L,1000); %1000 equally spaced x-values from 0 to L
plot(xvalues,g(xvalues),'r') %Sample g on xvalues & plot in red

%Annotated plot
xlabel('x')
ylabel('y')
k = legend('Runge-Kutta Method','Analytic Solution', 'Location','best');
set(k,'fontsize',12);
set(gca,'fontsize',12)  %gca = "get current axis" being used
```

Runge-Kutta MATLAB Output

```
>> HW6runge(0, 0, 0.2, 5)
n  x(n)    y(n)          Exact
0   0.00   0.000000    0.000000
```

1	0.20	0.021400	0.021403
2	0.40	0.091818	0.091825
3	0.60	0.222106	0.222119
4	0.80	0.425521	0.425541
5	1.00	0.718251	0.718282