## Table of Contents



# Lecture 25 Demo

```
clear
clc
close 'all'
```

# Define the problem to be solved.

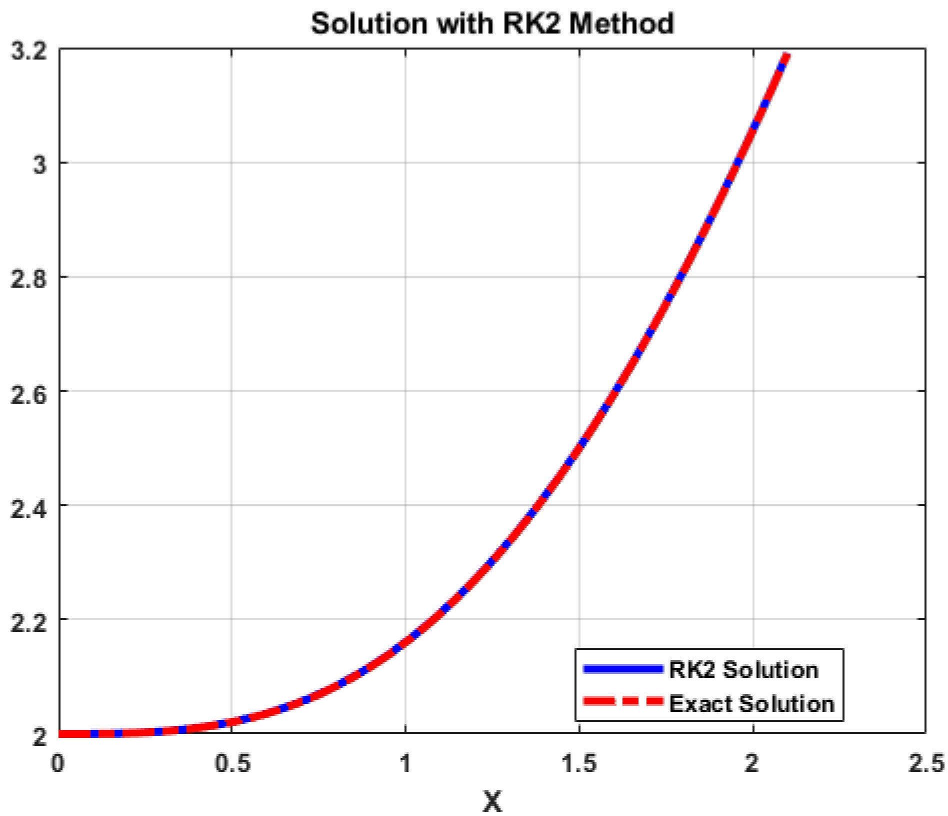
$y' = x^2/y, \quad y(0) = 2$

```
f = @(x,y) (x.^2)./y;
y_exact = @(x) sqrt((2/3)*x.^3 + 4);
xMin = 0; xMax = 2.1;
x_gold = linspace(xMin,xMax,1000);
```

# Second-Order Runge-Kutta

```
N = 30;
yINI = 2;
x = linspace(xMin,xMax,N);
y_RK2 = odeRK2(f,xMin,xMax,N,yINI);

figure(1)
plot(x,y_RK2,'-b',...
    x_gold,y_exact(x_gold),'-.r',...
    'linewidth',3);
title("Solution with RK2 Method",'fontsize',14,...
    'fontweight','bold');
xlabel('X','fontsize',12,'fontweight','bold');
legend('RK2 Solution','Exact Solution','location','best');
grid on;
set(gca,'fontsize',10,'fontweight','bold');
```



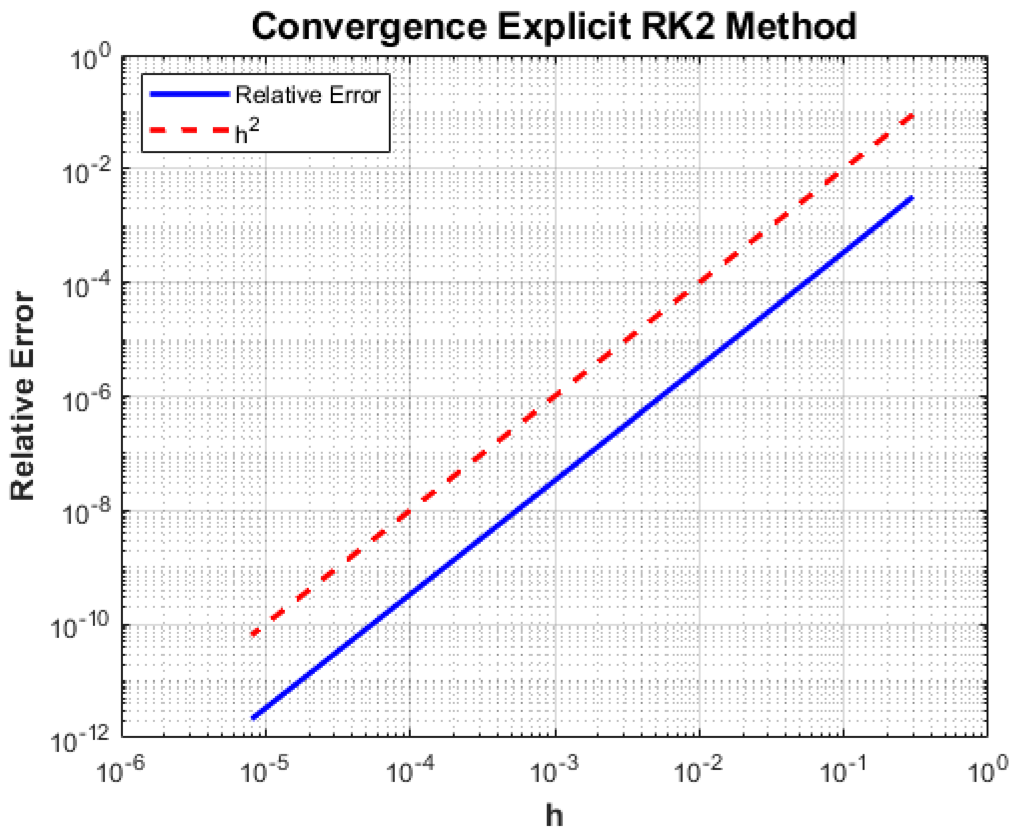

# Convergence Test

```
N = 3:18;
t = length(N);
err_array = nan(1,t);
h_array = nan(1,t);
for s = 1:t
   Nx = 2^(N(s));
   x = linspace(xMin, xMax, Nx);
   h = x(2)-x(1);
   h_array(s) = h;
   y_ns = odeRK2(f,xMin,xMax,Nx,yINI);
   err_array(s) = norm(y_exact(x)-y_ns,2)./...
       norm(y_exact(x),2);
end

err_gage = h_array.^2;

figure(2)
loglog(h_array,err_array,'-b',...
    h_array,err_gage,'--r','linewidth',2);
title("Convergence Explicit RK2 Method",...
    'fontsize',14,'fontweight','bold');
xlabel('h','fontsize',12,'fontweight','bold');
ylabel('Relative Error','fontsize',12,'fontweight','bold');
```

```
grid on
legend('Relative Error','h^2','location','best');
```




# Use Generalized Explicit RK method based on Butcher Tableau

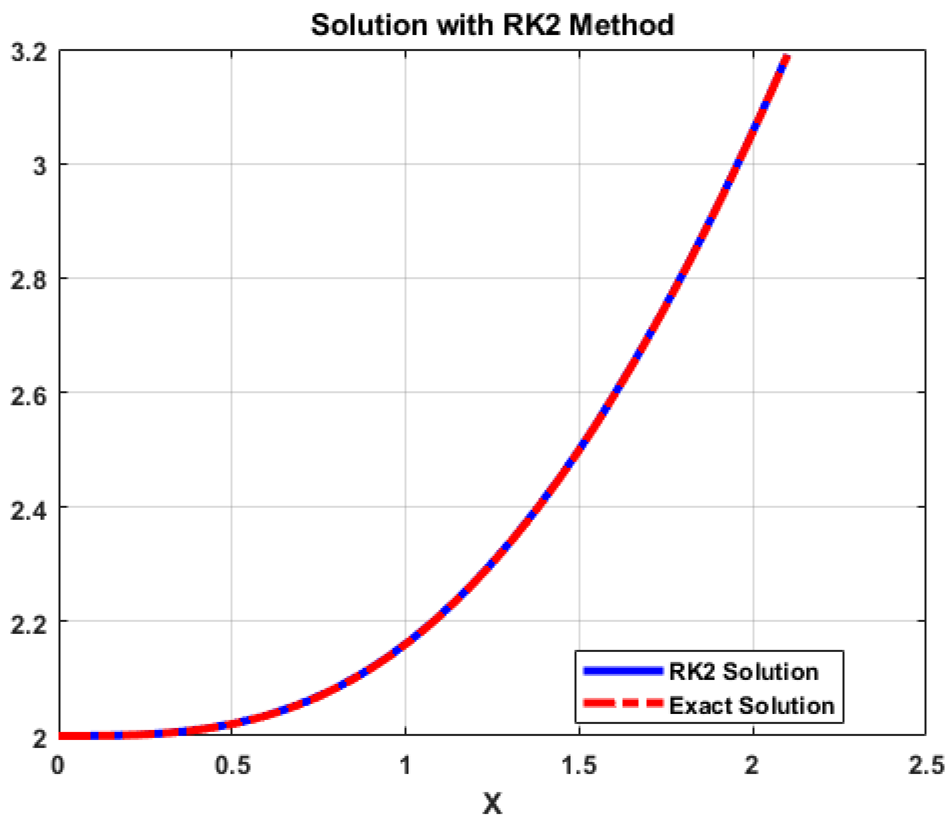
Butcher Tableau for 2nd order RK

```
s = 2;
BT = zeros(s+1,s+1);
C = [0; 1; 0];
B = [0 1/2 1/2];
A = [0 0;
    1 0;];
BT(:,1) = C;
BT(end,:) = B;
BT(1:s,2:end) = A;

N = 30;
yINI = 2;
x = linspace(xMin,xMax,N);
y_RK2 = odeExplicitRK(f,xMin,xMax,N,yINI,BT);

figure(3)
plot(x,y_RK2,'-b',...
```

```
    x_gold,y_exact(x_gold),'-.r',...
    'linewidth',3);
title("Solution with RK2 Method",'fontsize',14,...
    'fontweight','bold');
xlabel('X','fontsize',12,'fontweight','bold');
legend('RK2 Solution','Exact Solution','location','best');
grid on;
set(gca,'fontsize',10,'fontweight','bold');
```



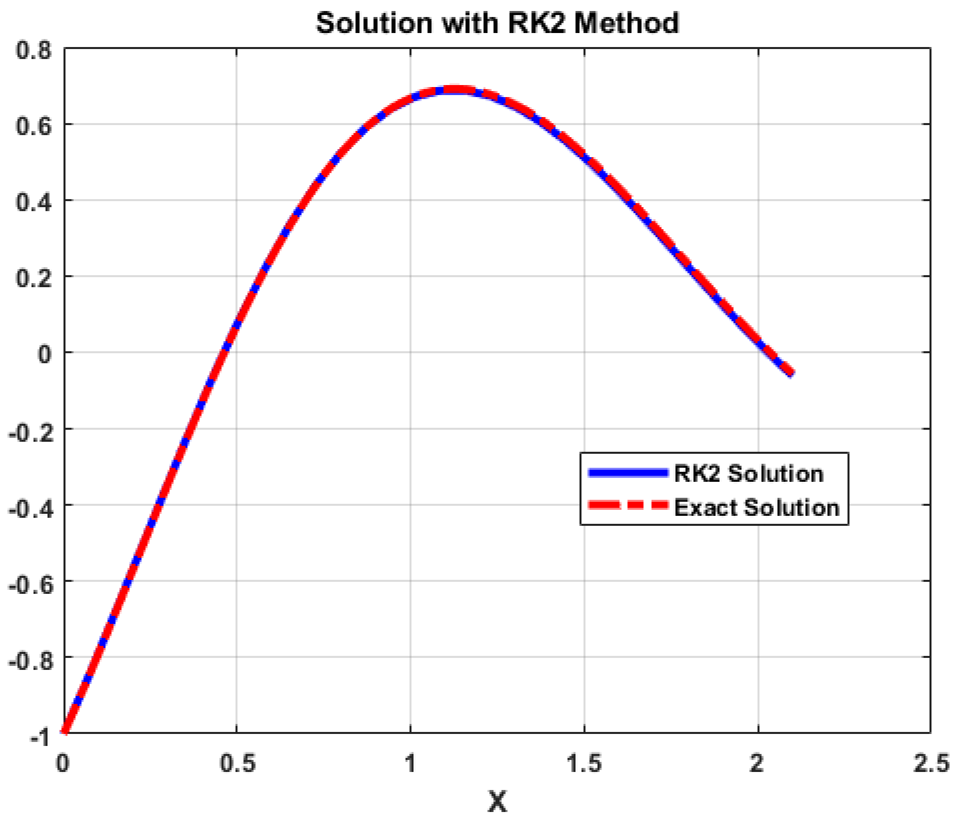

# Generalize for System of ODEs

```
N = 30;
f = @(t,y) ex2(t,y);
yINI = [-1 2]; % initial values
x = linspace(xMin,xMax,N);
ys_RK2 = odesExplicitRK(f,xMin,xMax,N,yINI,BT);

y_exact = @(x) exp(-x./2).*(-cos(2*x)+0.75*sin(2*x));

figure(4)
plot(x,ys_RK2(1,:),'-b',...
    x_gold,y_exact(x_gold),'-.r',...
    'linewidth',3);
title("Solution with RK2 Method",'fontsize',14,...
    'fontweight','bold');
xlabel('X','fontsize',12,'fontweight','bold');
```

```
legend('RK2 Solution','Exact Solution','location','best');
grid on;
set(gca,'fontsize',10,'fontweight','bold');
```



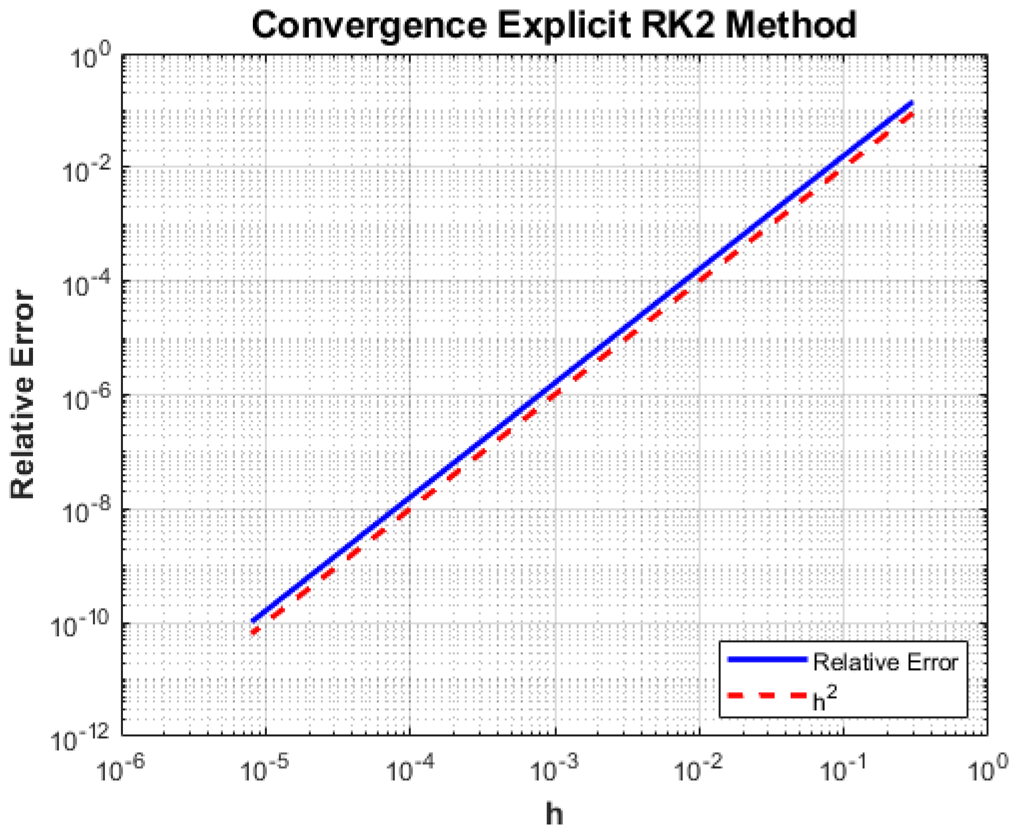

# Convergence Test

```
N = 3:18;
t = length(N);
err_array = nan(1,t);
h_array = nan(1,t);
for s = 1:t
   Nx = 2^(N(s));
   x = linspace(xMin, xMax, Nx);
   h = x(2)-x(1);
   h_array(s) = h;
   y_ns = odesExplicitRK(f,xMin,xMax,Nx,yINI,BT);
   err_array(s) = norm(y_exact(x)-y_ns(1,:),2)./...
       norm(y_exact(x),2);
end

err_gage = h_array.^2;

figure(5)
loglog(h_array,err_array,'-b',...
    h_array,err_gage,'--r','linewidth',2);
title("Convergence Explicit RK2 Method",...
```

```
    'fontsize',14,'fontweight','bold');
xlabel('h','fontsize',12,'fontweight','bold');
ylabel('Relative Error','fontsize',12,'fontweight','bold');
grid on
legend('Relative Error','h^2','location','best');
```




# Local Functions

```
function y = odeRK2(ODE,a,b,N,yINI)
% function y = odeRK2(ODE,a,b,h,yINI)
% y = solution
% ODE = function handle for y'
% a,b = begining and end of the interval for solution
% N = number of steps between a and b
% yINI = initial value for the solution

A = [0 0;
    1 0]; % RK matrix
B = [0.5 0.5]; % weights
c = [0 1]';% sample points
stages = 2;

x = linspace(a,b,N);
y = nan(1,N);
y(1) = yINI;
h = x(2)-x(1);
```

```matlab
for t = 1:(N-1)
    Xi = nan(1,stages);

    for s = 1:stages
       Xi(s) = y(t);
       for i = 1:(s-1)
          Xi(s) = Xi(s) + h*A(s,i)*ODE(x(t)+c(i)*h,Xi(i));
       end
    end

    y(t+1) = y(t);
    for i = 1:stages
       y(t+1) = y(t+1) + h*B(i)*ODE(x(t)+c(i)*h,Xi(i));
    end

end

end

function y = odeExplicitRK(ODE,a,b,N,yINI,BT)
% function y = odeExplicitRK(ODE,a,b,h,yINI,BT)
% y = solution
% ODE = function handle for y'
% a,b = begining and end of the interval for solution
% N = number of steps between a and b
% yINI = initial value for the solution
% BT = Butcher Tableau

% get Butcher Tableau Parameters
s = length(BT)-1;
c = BT(1:s,1);
B = BT(s+1,2:end);
A = BT(1:s,2:end);
stages = s;

x = linspace(a,b,N);
y = nan(1,N);
y(1) = yINI;
h = x(2)-x(1);
for t = 1:(N-1)
    Xi = nan(1,stages);

    for s = 1:stages
       Xi(s) = y(t);
       for i = 1:(s-1)
          Xi(s) = Xi(s) + h*A(s,i)*ODE(x(t)+c(i)*h,Xi(i));
       end
    end

    y(t+1) = y(t);
    for i = 1:stages
       y(t+1) = y(t+1) + h*B(i)*ODE(x(t)+c(i)*h,Xi(i));
    end
```

```
end
end

function y = odesExplicitRK(ODE,a,b,N,yINI,BT)
% function y = odeExplicitRK(ODE,a,b,h,yINI,BT)
% y = solution (vector)
% ODE = function handle for y'
% a,b = begining and end of the interval for solution
% N = number of steps between a and b
% yINI = initial value for the solution
% BT = Butcher Tableau

% get Butcher Tableau Parameters
s = length(BT)-1;
c = BT(1:s,1);
B = BT(s+1,2:end);
A = BT(1:s,2:end);
stages = s;

x = linspace(a,b,N);
sys_size = length(yINI);
y = nan(sys_size,N);
y(:,1) = yINI;
h = x(2)-x(1);
for t = 1:(N-1)
    Xi = nan(sys_size,stages);

    for s = 1:stages
       Xi(:,s) = y(:,t);
       for i = 1:(s-1)
          Xi(:,s) = Xi(:,s) + h*A(s,i)*ODE(x(t)+c(i)*h,Xi(:,i));
       end
    end

    y(:,t+1) = y(:,t);
    for i = 1:stages
       y(:,t+1) = y(:,t+1) + h*B(i)*ODE(x(t)+c(i)*h,Xi(:,i));
    end

end
end

function dw = ex2(~,w) % generally expect 2 arguments for solvers (IV
 first)
dw = nan(2,1);
dw(1) = w(2);
dw(2) = -0.25*(4*w(2) + 17*w(1));
end
```

*Published with MATLAB® R2018a*