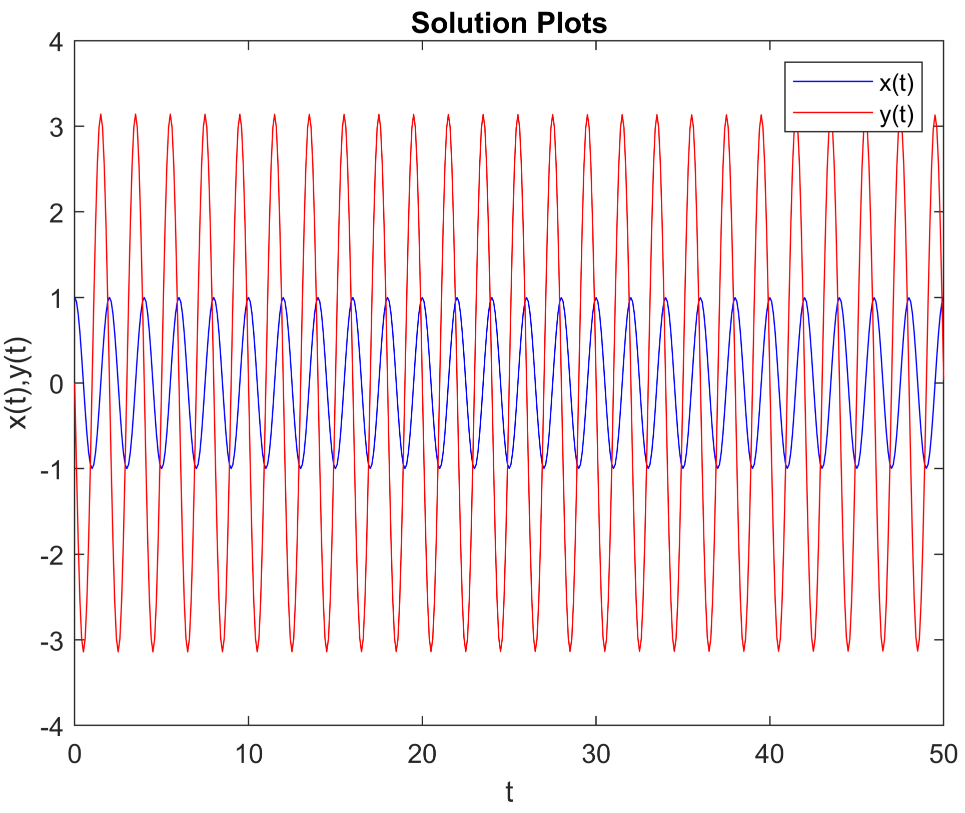
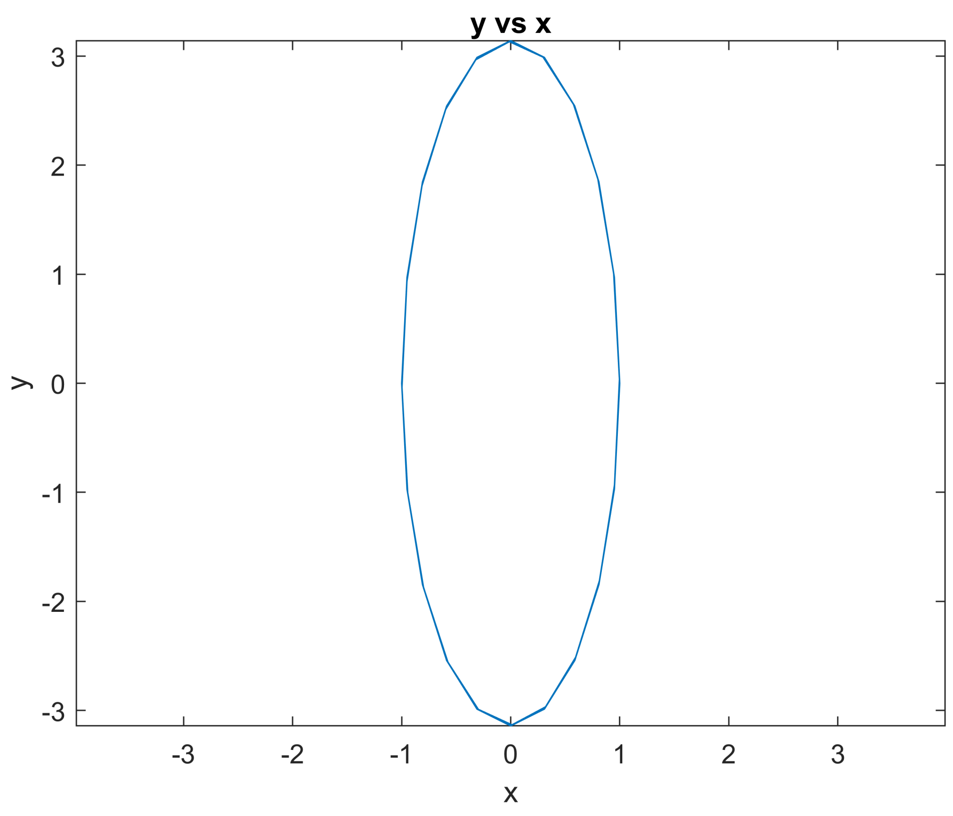
# Fourth Order Runge-Kutta for Systems

First we do a simple RK4 routine for a system of two equations.

```
% rk4 for system of two equations
%    x' = f(x,y,t)
%    y' = g(x,y)

clear
% Define functions
omega = 2*pi/2;
f = @(x,y,t) y;
g = @(x,y,t) -omega^2*x;

% Interval [a,b] and N+1 steps of size h
a = 0;
b = 50;
N = 500;
h = (b-a)/N;
t = a+(0:N)*h;

% Initial conditions
x(1)=1;
y(1)=0;

% RK4 Routine
for i=1:N
    j1 = h*f(x(i),y(i),t(i));
    k1 = h*g(x(i),y(i),t(i));

    j2 = h*f(x(i)+j1/2,y(i)+k1/2,t(i)+h/2);
    k2 = h*g(x(i)+j1/2,y(i)+k1/2,t(i)+h/2);

    j3 = h*f(x(i)+j2/2,y(i)+k2/2,t(i)+h/2);
    k3 = h*g(x(i)+j2/2,y(i)+k2/2,t(i)+h/2);

    j4 = h*f(x(i)+j3,y(i)+k3,t(i)+h);
    k4 = h*g(x(i)+j3,y(i)+k3,t(i)+h);

    x(i+1) = x(i)+(j1+2*j2+2*j3+j4)/6; % fixed - too many h's
    y(i+1) = y(i)+(k1+2*k2+2*k3+k4)/6; % fixed - too many h's
end

% Plot solutions
plot(t, x,'b',t,y,'r')
xlabel('t')
ylabel('x(t),y(t)')
title('Solution Plots')
legend('x(t)','y(t)')
```

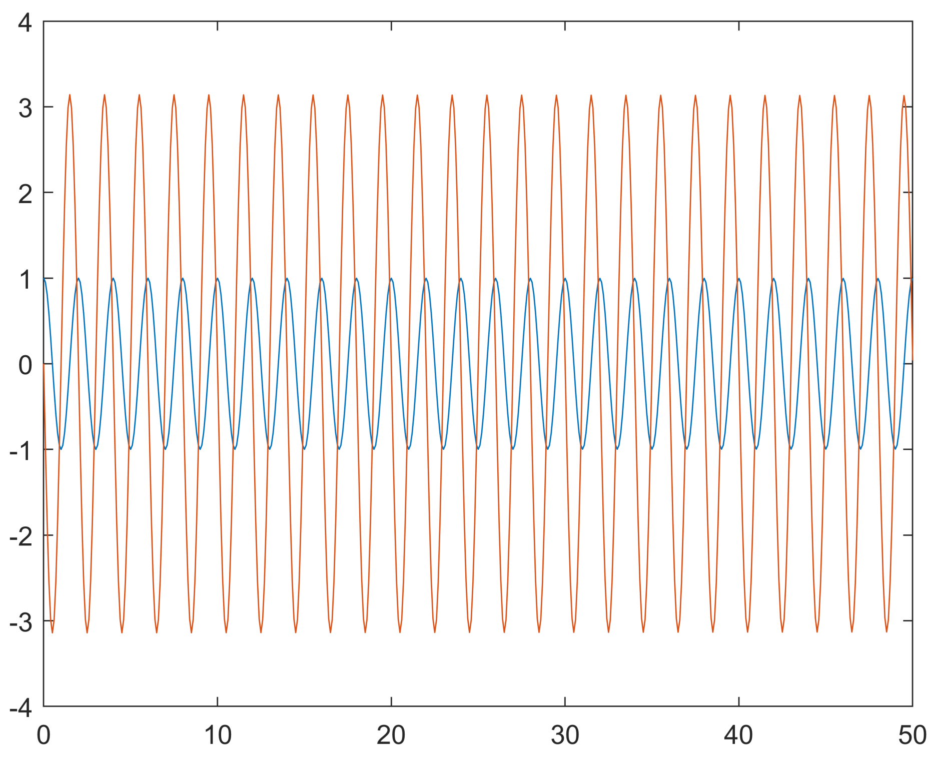


```
% Phase Plane Plot
plot(x,y)
xlabel('x')
ylabel('y')
title('y vs x')
axis equal
```



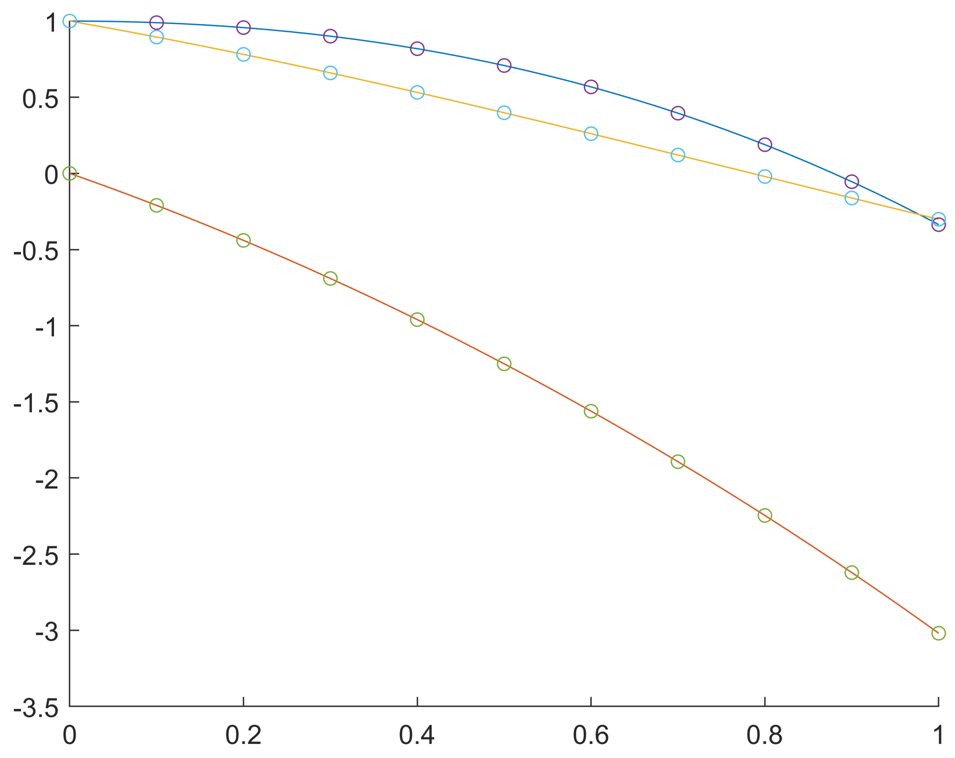
The same problem but using the RK4 function below.

```
odesys = @(t, y) [f(y(1),y(2),t); g(y(1),y(2),t)];
y0 = [1;0];
%t = 0:0.1:5;
z = RK4(odesys, t, y0);
plot(t,z)
```



Here is an example showing that RK$ can take more equations in the system. This is three ODEs in three unknowns.

```
% setup
odefun = @(t, y) [y(2); -y(1) - 2*exp(t) + 1; -y(1) - exp(t) + 1];
y0 = [1;0;1];

% ODE45 solution
[T, Y] = ode45(odefun, [0,1], y0);

% RK4 solution
t = 0:0.1:1;
y = RK4(odefun, t, y0);

% Compare results
figure;
hold on;
plot(T, Y);
plot(t, y, 'o','MarkerSize', 5)
hold off
```



```matlab
function y = RK4(odefun, tspan, y0)
% ODEFUN contains the ode functions of the system
% TSPAN  is a 1D vector of equally spaced t values
% Y0     contains the intial conditions for the system variables
% from
% https://stackoverflow.com/questions/43408704/solve-a-system-of-equations-with-runge-kutta-4-r
    % Initialise step-size variables
    t = tspan(:); % ensure column vector = (0:h:1)';
    h = t(2)-t(1);% define h from t
    N = length(t);

    % Initialise y vector, with a column for each equation in odefun
    y = zeros(N, numel(y0));
    % Starting conditions
    y(1, :) = y0(:)';  % Set intial conditions using row vector of y0

    k = zeros(4, numel(y0));              % Initialise K vectors
    b = repmat([1 2 2 1]', 1, numel(y0)); % RK4 coefficients

    % Iterate, computing each K value in turn, then the i+1 step values
    for i = 1:(N-1)
        k(1, :) = odefun(t(i), y(i,:));
        k(2, :) = odefun(t(i) + (h/2), y(i,:) + (h/2)*k(1,:));
        k(3, :) = odefun(t(i) + (h/2), y(i,:) + (h/2)*k(2,:));
        k(4, :) = odefun(t(i) + h, y(i,:) + h*k(3,:));

        y(i+1, :) = y(i, :) + (h/6)*sum(b.*k);
    end
end
```