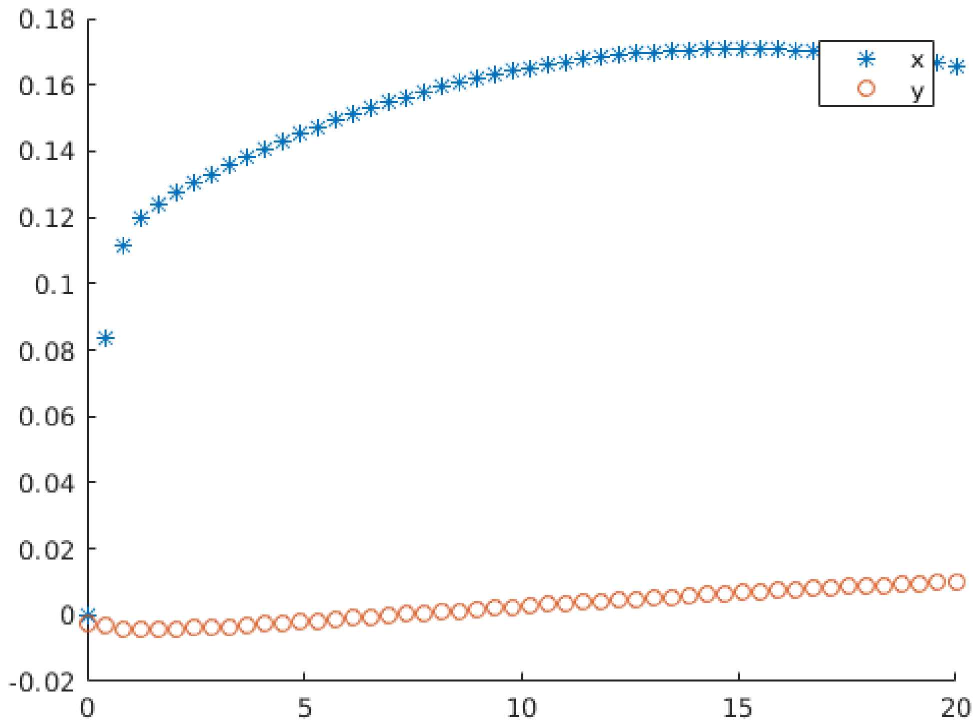
```
% problem 3 - two spring system

type spring_funcs.m

k2_vals = linspace(0, 20, 50);

for i = 1:length(k2_vals)
    % declare constants
    k1 = 10; % N/m
    k2 = k2_vals(i);
    L1 = 0.1; % m
    L2 = L1;
    m = 0.1; % kg
    D = 0.1; % m
    g = 9.81; % m/s
    param = [k1, k2, L1, L2, D, m];
    % parameters for numerical solution
    tol = 1e-6;
    s = [0, -(m*g/k1 + L1)]; % initial guess

    % do first step of newton's method
    [F, D] = spring_funcs(s, param);
    FDi = F / D; % F * D^-1
    sn = s - FDi;
    err = norm(sn - s);
    s = sn;

    % solve to tolerance
    while err > tol
        [F, D] = spring_funcs(s, param);
        FDi = F / D; % F * D^-1
        sn = s - FDi;
        err = abs(sn) - abs(s);
        s = sn;
    end

    % store values to plot
    xp(i) = s(1);
    yp(i) = s(2);
end

% plot values
hold on
plot(k2_vals, xp, '*')
plot(k2_vals, yp, 'o')
hold off
legend('x', 'y')


function [F, D] = spring_funcs(s, param)
% [F, D] = spring_funcs(s, param)
% calculate force and jacobian of force for dual spring problem
% F: force on mass
% D: Jacobian of force
% s: state vector [x, y]
% param: parameter array [k1, k2, L1, L2, D, m]

% get components from state vector
x = s(1);
y = s(2);
% get parameters
k1 = param(1);
k2 = param(2);
L1 = param(3);
L2 = param(4);
D = param(5);
m = param(6);
g = 9.81;
% alpha and beta terms
a = sqrt(x^2 + y^2);
b = sqrt((x-D)^2 + y^2);
% derivatives of alpha and beta
ax = -x / a;
ay = -y / a;
bx = (D - x) / a;
by = -y / b;
% forces
Fx = -k1*x + k1*L1*x/a - k2*(x-D) + k2*L2*(x-D)/b;
Fy = -k1*y + k1*L1*x/a - k2*y + k2*L2*y/b + m*g;
F = [Fx, Fy];
% jacobian of force
D(1, 1) = -k1 + k1*L1*((a - x*ax)/a^2) ...
          - k2 + k2*L2*(b - (x-D)*bx)/b^2;
D(2, 1) = -k1*L1*x*ay/a^2 - k2*L2*(x-D)*by/b^2;
D(1, 2) = - k1*L1*y*ax/a^2 - k2*L2*y*bx/b^2;
D(2, 2) = -k1 + k1*L1*(x - y*ay)/a^2 ...
          - k2 + k2*L2*(b - y*by)/b^2;
end
```



Published with MATLAB® R2019a