```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Exercise 6 - Non linear Least squares                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;clear all;close all;
%% TASK 1
% Write a simulation function simstep that takes the following seven real
% numbers C1;C2;C3; px(0); vx(0);T;D as inputs, and computes from them the
% state px(T); vx(T) at the time dT,assuming a constant value D on the
% interval [0;dT].

C1 = 10;C2 = 0.1;C3 = 0.6;p0 = 0;v0 = 0;deltaT = 0.1;Du = 0.4;
[Px,Vx] = simstep(C1,C2,C3,p0,v0,deltaT,Du);
%% TASK 2
% Write a simloop function around simstep that simulates N time steps of
% length dT and takes as input, besides C1;C2;C3;px(0);vx(0);T and N a
% vector of values Dk, k = 1,..,N. As output, the function should generate
% the values px(k*dT) and vx(k*dT) for k = 0,1,..,N. T. Plot the trajectory.

N = 100;                                           % nunmber of iterations
Dk = ones(1,N);                                    % constant input
[p_xk,v_xk] = simloop(C1,C2,C3,p0,v0,Dk,N,deltaT);

time = linspace(0,deltaT*(N-1),N);
figure(1);
subplot(2,1,1);plot(time,v_xk,'ob');grid on;
title('Velocity trajectory');xlabel('Time');ylabel('Velocity');
subplot(2,1,2);plot(time,p_xk,'og');grid on;
title('Position trajectory');xlabel('Time');ylabel('Position');

%% TASK 3
% Load data6 1.txt. These are time-dependent measurements of the form
% |time|velocity|D|. We assume no noise on the measurements of time and on D,
% and the velocity measurements i.i.d. measurement errors.

data = importdata('data6_1.txt');                % Import data
global t D vel
t   = data(:,1); % time
vel = data(:,2); % velocity
D   = data(:,3); % actuation

%% Task 3.a
% Estimation of theta=|C2|C3|vx(0)|. C1 = 10 is known. formulate a residual
% function [res]=residual(vel, theta) that computes the misfit M(theta)-y
% between the model predictions and the actual measurements.

theta0 = [0.1,0.6,0];C1 = 10;        % true value
res = residuals(vel,[C1, theta0]);figure(2);plot(res);
title('residual');grid on;                                 % check function
theta0 = [pi,34,23];                                       % initial condition
[theta,resnorm,residual] = lsqnonlin(@(theta)residuals(vel,[C1,theta]),theta0);
C2_est = theta(1);C3_est = theta(2);V0_est = theta(3);

str = '%s* = %f \n';C = {'C2','C3','V0';C2_est,C3_est,V0_est};
disp(sprintf(str,C{:}));

%% TASK 3.b
% Estimate the confidence ellipsoid around the estimate of [C2;C3;vX(0)],
% The matrix N by the Jacobian dM/dtheta.

% Calculating the Jacobian after its derivation from the ODEs
j1 = (exp(-C3_est*t)-1)/C3_est;
j2 = (C1*D-C2_est)/C3_est.*(t.*exp(-C3_est*t))+(exp(-C3_est*t)-1)/C3_est-V0_est*exp(-C3_est*t).*t;
j3 = exp(-C3_est*t);
Jacobian = [j1,j2,j3];
```

```
[N,d]=size(Jacobian);          % N=Number of experiments, d=N of regressors
variance_res = sum(residual.^2)/(N-d);                 % estimated variance
covariance_est = variance_res*inv(Jacobian'*Jacobian); % Covariance Estimator

% ELLIPSES PLOTTING
[V,Diag] = eig(covariance_est);    % get eigenvalues and eigenvector of covariances
xy = [cos(linspace(0,2*pi,50));sin(linspace(0,2*pi,50))]; % generate coordinates of 50 points unit cir
xy_ellipse_C2_C3 = [C2_est;C3_est]*ones(1,50) + V([1 2],[1 2]) * sqrt(Diag([1 2],[1 2]))*xy;
xy_ellipse_C2_V0 = [C2_est;V0_est]*ones(1,50) + V([1 3],[1 3]) * sqrt(Diag([1 3],[1 3]))*xy;
xy_ellipse_C3_V0 = [C3_est;V0_est]*ones(1,50) + V([2 3],[2 3]) * sqrt(Diag([2 3],[2 3]))*xy;

% plotting the ellipses
figure(2);
subplot(3,1,1);
plot(xy_ellipse_C2_C3(1,:), xy_ellipse_C2_C3(2,:),'-g','LineWidth',2);grid on;
title('2D 1-sigma confidence ellipsoid for C2-C3')
xlabel('C2*');ylabel('C3*');

subplot(3,1,2);
plot(xy_ellipse_C2_V0(1,:),xy_ellipse_C2_V0(2,:),'-','LineWidth',2);grid on;
title('2D 1-sigma confidence ellipsoid for C2-V0')
xlabel('C2*');ylabel('V0*');

subplot(3,1,3);
plot (xy_ellipse_C3_V0(1,:), xy_ellipse_C3_V0(2,:), '-y', 'LineWidth',2);grid on;
title('2D 1-sigma confidence ellipsoid for C3-V0')
xlabel('C3*');ylabel('V0*');

%% Task 3.c
% Try to estimate the four parameters [C1;C2;C3;vX(0)] simultaneously.
theta0 = [pi,3,2,1.5];                                 % initial condition
[theta_3c,resnorm,residual] = lsqnonlin(@(theta)residuals(vel,theta),theta0);
disp(theta_3c)
% We obtain wrong value due to a constant input

%% TASK 4
% Load data6 2.txt. Now, the dutycycle is not constant over the given interval.

data = importdata('data6_2.txt');
t   = data(:,1);    % time
vel = data(:,2);    % velocity
D   = data(:,3);    % actuation

%% task 4.a
% Estimate the four parameters [C1;C2;C3;vx(0)] using the new data.
theta0_c = [10,0.1,0.6,0]; % Initial guess theta = [C1,C2,C3,V0]
[theta_c,resnorm,residual] = lsqnonlin(@(theta_c)residuals(vel,theta_c),theta0_c);
C1_est = theta_c(1);C2_est = theta_c(2);
C3_est = theta_c(3);V0_est = theta_c(4);

str_c = '%s* = %f \n';     % print values
C = {'C1','C2','C3','V0';C1_est,C2_est,C3_est,V0_est};
disp(sprintf(str_c,C{:}))
```