```
% Optimization of two-link robot arm tracking
clear; clc;

% Define desired trajectory
qDes = [ -0.4986    2.5681;
          0.5371    1.5108 ];

% Middle Points
qMid = [inverse_kinematics(0.4, 0.6, 1, 1), ...
        inverse_kinematics(0.4, 0.8, 1, 1), ...
        inverse_kinematics(0.4, 0.9, 1, 1), ...
        inverse_kinematics(0.4, 1.2, 1, 1)];


%  weights
wt = [1000, 0, 0]; % [qd_wt, time_wt, midpt_wt]

% Optimization setup
initParams = [10 20   .1 20 25]; % Initial guess for [time, wn, bj, kj]

[init_T, init_Y] = ode45(@(t, x) myTwolinkwithprefilter(t, x, initParams(3),
initParams(1:2), qDes, initParams(4), initParams(5)),   [0 initParams(2)],
zeros(8, 1));

% Lower and upper boundaries
lb = [0 0    1.5   10    2   ];    % Lower bounds
ub = [3 6    50  200    500 ];   % Upper bounds

% Objective Function
objectiveFunc = @(params) objectiveFunction(params, qDes, wt,qMid);

% Run optimization
options = optimset('Display', 'iter', 'TolFun', 1e-6, 'MaxIter', 200);
optimalParams = fmincon(objectiveFunc, initParams, [], [], [], [], lb, ub,
[], options);

% Simulate with optimal parameters and plot results
[t, y] = ode45(@(t, x) myTwolinkwithprefilter(t, x, optimalParams(3),
optimalParams(1:2), qDes, optimalParams(4), optimalParams(5)), [0
optimalParams(2)], zeros(8, 1));

% Output
xAct = forward_kinematics(y(:, 5), y(:, 6), 1, 1);
xDes = forward_kinematics(qDes(:, 1), qDes(:, 2), 1, 1);
xInit = forward_kinematics(init_Y(:, 5), init_Y(:, 6), 1, 1);
```

```
Initial point X0 is not between bounds LB and UB;
FMINCON shifted X0 to strictly satisfy the bounds.

                                            First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       6    3.578872e+01    0.000e+00    1.797e+01
    1      12    2.405328e-01    0.000e+00    1.763e+01    1.095e+01
    2       18    2.041177e-01    0.000e+00    1.515e+03    3.182e-01
    3       25    1.293050e-01    0.000e+00    3.555e-01    1.705e+00
    4       31    2.071935e-02    0.000e+00    7.369e-02    3.083e+00
    5       37    1.247150e-02    0.000e+00    7.246e-02    9.940e-01
    6       43    7.260020e-03    0.000e+00    6.586e-02    1.460e-01
    7       49    6.155053e-03    0.000e+00    2.428e-02    1.039e-01
    8       55    7.183270e-03    0.000e+00    2.022e-02    2.040e-01
    9       61    6.253474e-03    0.000e+00    5.026e-03    1.300e-01
   10       68    6.123198e-03    0.000e+00    1.537e-03    4.917e-02
   11       74    6.122282e-03    0.000e+00    6.142e-04    1.202e-03
   12       80    6.122177e-03    0.000e+00    6.279e-04    2.519e-04
   13       87    6.121915e-03    0.000e+00    6.189e-04    4.715e-04
   14       93    6.120280e-03    0.000e+00    3.447e-04    4.322e-03
   15       99    6.119917e-03    0.000e+00    2.000e-04    6.062e-04
   16      105    6.118512e-03    0.000e+00    8.784e-05    2.162e-03
   17      113    6.118476e-03    0.000e+00    3.967e+01    3.450e-04
   18      119    1.414872e-03    0.000e+00    1.759e+02    9.820e-01
   19      136    1.404879e-03    0.000e+00    4.454e-02    3.819e-04
   20      142    6.004111e-04    0.000e+00    2.793e-02    2.319e-02
   21      149    1.711579e-04    0.000e+00    2.889e-03    2.750e-01
   22      161    1.685538e-04    0.000e+00    5.754e-04    6.215e-03
   23      177    1.670781e-04    0.000e+00    3.712e-04    5.964e-04
   24      188    1.660642e-04    0.000e+00    2.855e-04    2.427e-03
   25      195    1.647741e-04    0.000e+00    6.085e-04    3.723e-03
   26      201    1.645227e-04    0.000e+00    5.564e-04    7.768e-03
   27      207    1.658844e-04    0.000e+00    1.122e-03    2.225e-02
   28      214    1.719960e-04    0.000e+00    2.665e-03    2.892e-01
   29      220    1.927017e-04    0.000e+00    3.779e-03    5.502e-01
   30      230    1.759056e-04    0.000e+00    2.122e-03    4.075e-03

                                            First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
   31      241    1.737980e-04    0.000e+00    2.840e-04    1.602e-01
   32      262    1.737880e-04    0.000e+00    1.724e+02    1.779e-05
   33      272    1.737880e-04    0.000e+00    2.824e-04    3.314e-08
   34      288    1.737870e-04    0.000e+00    1.723e+02    1.940e-06
   35      298    1.737869e-04    0.000e+00    1.723e+02    1.513e-11

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.
```

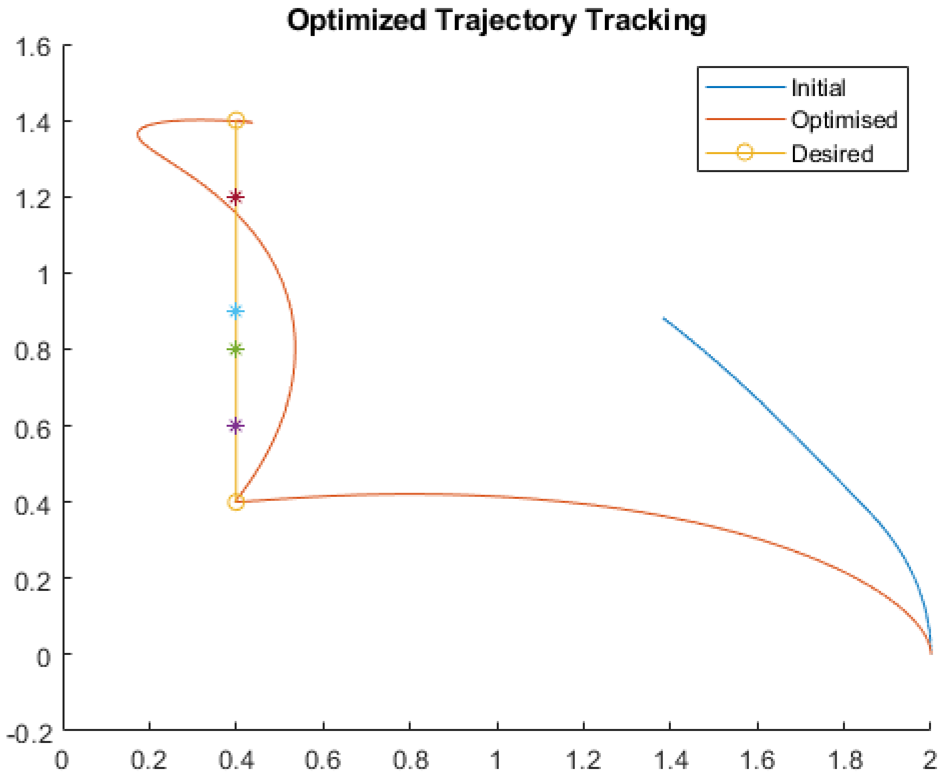
```
figure(1); hold on;
plot(xInit(:, 1), xInit(:, 2), '-');
plot(xAct(:, 1), xAct(:, 2), '-');
plot(xDes(:, 1), xDes(:, 2), 'o-');
plot(0.4,0.6, '*',0.4,0.8, '*',0.4,0.9, '*',0.4,1.2, '*');

legend('Initial','Optimised', 'Desired');
title('Optimized Trajectory Tracking');
disp(['Optimized Parameters :', num2str(optimalParams)])
```

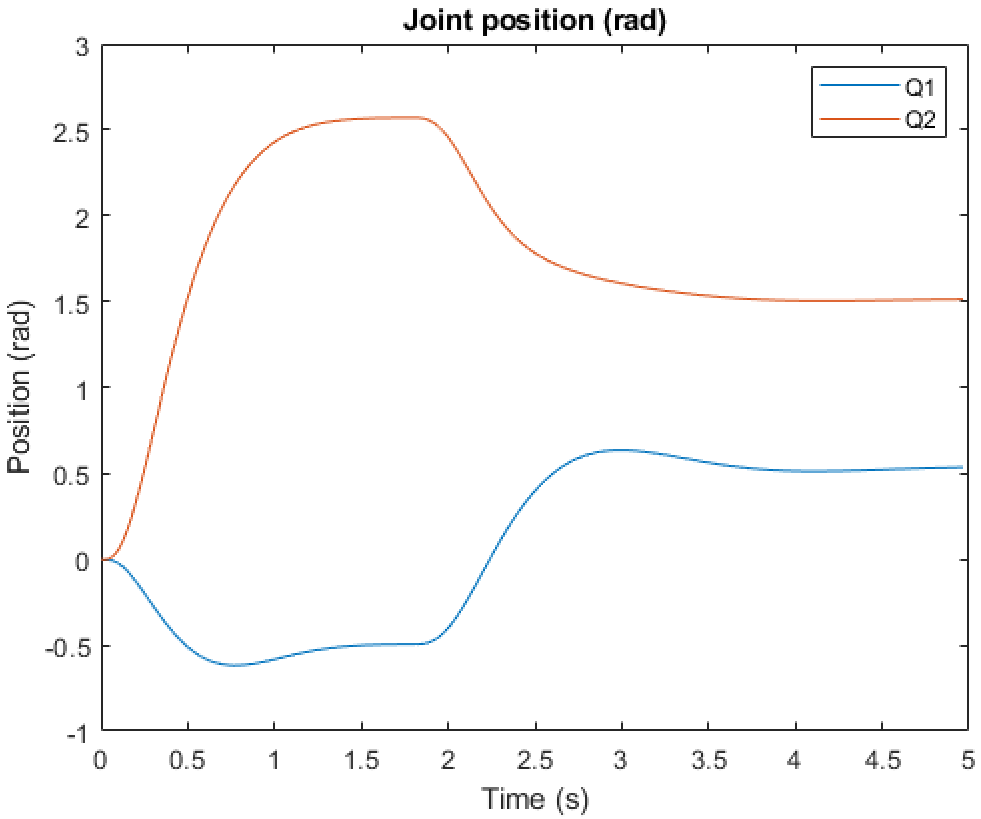
```
% mid points in joint space
figure(2);plot(y(:,5),y(:,6),qMid(1,:),qMid(2,:),'o');
xlabel('Joint 1 position')
ylabel('Joint 2 position')
title('joint space of a (near) optimal staight line in cartesian space')


% joint space plot
figure(2); grid on;
plot(t,y(:,5:6));
xlabel('Time (s)')
ylabel('Position (rad)')
title('Joint position (rad)')
legend('Q1','Q2')

%  cartesian space plot
figure(3); hold on; grid on;
plot(xAct(:,1),xAct(:,2))
xlabel('X axis')
ylabel('Y axis')
title('Cartesian Position (m)')


% x/y vs time
figure(4); grid on; hold on;
plot(t,xAct(:,1:2))
xlabel('Time (s)')
ylabel('Position')
legend('X','Y')
title('Cartesian Position vs Time')
% %% publish('simOpt2.m','pdf');
% disp(sprintf('KY %s \t %s \t %s',mfilename,pwd,datetime("now")));

% Objective function
function error = objectiveFunction(params, qDes,wt,qMid)
    time = [params(1) params(2)];
    wn = params(3);
    bj = params(4);
    kj = params(5);

    % Initial conditions
    x0 = zeros(8, 1);
    x0(1:2) = [qDes(1, 1); qDes(1, 2)];

    % Simulate the system
    [t, y] = ode45(@(t, x) myTwolinkwithprefilter(t, x, wn, time, qDes,
bj(1), kj(1)), [0 time(end)], x0);

    % Calculate the error metric
    distto1 = min(sum((y(:, 5:6) - qDes(1,:)).^2,2));
    distto2 = min(sum((y(:, 5:6) - qDes(2,:)).^2,2));

    distMid1 = min(sum((y(:, 5:6) - qMid(:,1)').^2,2));
    distMid2 = min(sum((y(:, 5:6) - qMid(:,2)').^2,2));
    distMid3 = min(sum((y(:, 5:6) - qMid(:,3)').^2,2));
    distMid4 = min(sum((y(:, 5:6) - qMid(:,4)').^2,2));

    time1 = min(sum((time(1) - t).^2,2));
    time2 = min(sum((time(2) - t).^2,2));

    error   = wt(1) * distto1  + wt(1) * distto2  + ...
              wt(2) * time1    + wt(2) * time2    + ...
              wt(3) * distMid1 + wt(3) * distMid2 + ...
              wt(3) * distMid3 + wt(3) * distMid4;


    % distto5 = 5000 * sum((y(:, 5:6) - qMid3'),2) + w2 *
(sum(  (   (time(1) + (time(2) - time(1))/2 ) - t).^2   ,2));

end

% myTwolinkwithprefilter function
function dxdt = myTwolinkwithprefilter(t, x, wn, time, qDes, bj, kj)
    zeta = 1.0;
    A = [zeros([2 2]) eye(2); -eye(2)*wn^2 -eye(2)*2*zeta*wn];
    B = [0 0; 0 0; wn^2 0; 0 wn^2];

    % Actual position and velocity
    q = x(5:6);
    qd = x(7:8);
    q1p = x(7); q2p = x(8);
    q1 = x(5); q2 = x(6);

    % Robot constants
    L_1 = 1; L_2 = 1; m_1 = 1; m_2 = 1;
    ka = L_2^2 * m_2;
    kb = L_1 * L_2 * m_2;
    kc = L_1^2 * (m_1 + m_2);

    M = [ka + 2*kb*cos(q2) + kc, ka + kb*cos(q2);
         ka + kb*cos(q2), ka];
    V = ka*sin(q2)*([0 -1; 1 0] * [q1p^2; q2p^2] + [-2*q1p*q2p; 0]);

    Numerator = V + [-bj 0; 0 -bj]*qd + [-kj 0; 0 -kj]*(q - x(1:2));
    qdd = M\Numerator;
    if t < time(1)
        dotx = A*x(1:4) + B*qDes(1, :)';
    else
      dotx = A*x(1:4) + B*qDes(2, :)';
    end
    dxdt = [dotx; qd; qdd];
end



% % Optimization of two-link robot arm tracking
% clear; clc;
```

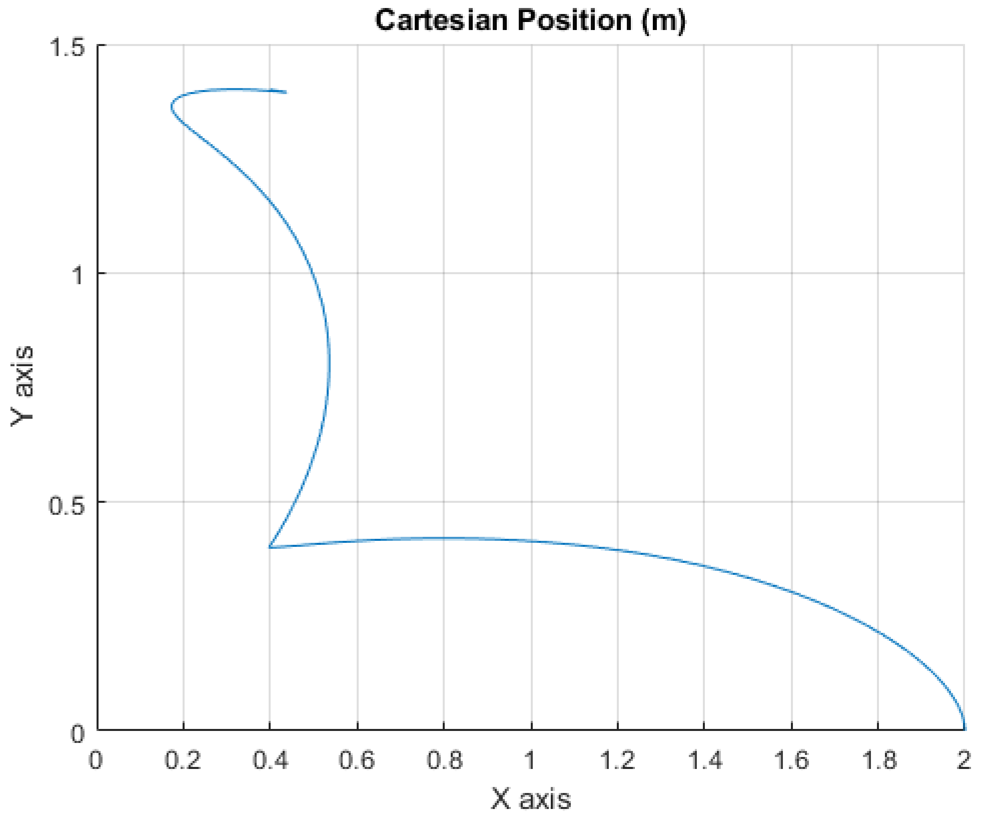
```
%
% % Define desired trajectory
% qDes = [ -0.4986    2.5681;
%           0.5371    1.5108 ];
% % Define the paramaters as a vector
% wn = [ 2 2]; % weights
% B  = [20 20];  % Damping
% K  = [45 45];  % Spring
%
% % Middle Points for error calculation
% qMidx = [ -0.2191     0          0.0967     0.3630];
% qMidy = [ 2.4039     2.2143     2.1118     1.7722];
%
% % Optimization setup
% initParams = [10 20   wn(1) wn(2) B(1) B(2) K(1) K(2)]; % Initial guess
for [time, wn, bj, kj]
%
% [init_T, init_Y] = ode45(@(t, x) myTwolinkwithprefilter(t, x, wn,
initParams(1:2), qDes, B, K), [0 initParams(2)], zeros(8, 1));
%
% % Lower and upper boundaries
% lb = [0 0    1  1   1   1   2   2  ];   % Lower bounds
% ub = [3 6   50 50 200 200 500 500 ];  % Upper bounds
%
% % Objective Function
% objectiveFunc = @(params) objectiveFunction(params,
qDes,qMidx,qMidy,wn,B,K);
%
% % Run optimization
% options = optimset('Display', 'iter', 'TolFun', 1e-6, 'MaxIter', 200);
% optimalParams = fmincon(objectiveFunc, initParams, [], [], [], [], lb, ub,
[], options);
%
% % Simulate with optimal parameters and plot results
% [t, y] = ode45(@(t, x) myTwolinkwithprefilter(t, x, optimalParams(3:4),
optimalParams(1:2), qDes, optimalParams(5:6), optimalParams(7:8)), [0
optimalParams(2)], zeros(8, 1));
%
% % Output
% xAct = forward_kinematics(y(:, 5), y(:, 6), 1, 1);
% xDes = forward_kinematics(qDes(:, 1), qDes(:, 2), 1, 1);
% xInit = forward_kinematics(init_Y(:, 5), init_Y(:, 6), 1, 1);
% %%
% figure(1); hold on;
% plot(xInit(:, 1), xInit(:, 2), '-');
% plot(xAct(:, 1), xAct(:, 2), '-');
% plot(xDes(:, 1), xDes(:, 2), 'o-');
% plot(0.4,0.6, '*',0.4,0.8, '*',0.4,0.9, '*',0.4,1.2, '*');
%
% legend('Initial','Optimised', 'Desired');
% title('Optimized Trajectory Tracking');
% disp(['Optimized Parameters :', num2str(optimalParams)])
%
% %% mid points in joint space
% %
% qMidx = [ -0.2191 0      0.0967     0.3630];
% qMidy = [ 2.4039     2.2143     2.1118     1.7722];
%
% % figure(6);plot(y(:,5),y(:,6),qMidx,qMidy,'o');
% % title('joint space of a (near) optimal staight line in cartesian space')
% %% joint space plot
% % figure(5);plot(t,y(:,5:6));
%
% %%  cartesian space plot
% % figure(3); plot(xAct(:,1),xAct(:,2))
% %% x/y vs time
% % figure(4); plot(t,xAct(:,1:2))
%
% %% publish('simOpt2.m','pdf');
% disp(sprintf('KY %s \t %s \t %s',mfilename,pwd,datetime("now")));
%
% % Objective function
% function error = objectiveFunction(params, qDes,qMidx,qMidy,wn,B,K)
%     time = [params(1) params(2)];
%
%     % Initial conditions
%     x0 = zeros(8, 1);
%     x0(1:2) = [qDes(1, 1); qDes(1, 2)];
%
%     % Simulate the system
%     [t, y] = ode45(@(t, x) myTwolinkwithprefilter(t, x, wn, time, qDes, B,
K), [0 time(end)], x0);
%
%     % weights, could be done as a vector of weights
%     w1 = 1000;
%     w2 = 0;
%     w3 = 0;
%
%
%     % Calculate the error metric
%     distto1 = min(sum((y(:, 5:6) - qDes(1,:)).^2,2));
%     distto3 = min(sum((y(:, 5:6) - [qMidx(1) qMidy(1)]).^2,2));
%     distto4 = min(sum((y(:, 5:6) - [qMidx(2) qMidy(2)]).^2,2));
%     distto5 = min(sum((y(:, 5:6) - [qMidx(3) qMidy(3)]).^2,2));
%     distto6 = min(sum((y(:, 5:6) - [qMidx(4) qMidy(4)]).^2,2));
%     distto2 = min(sum((y(:, 5:6) - qDes(2,:)).^2,2));
%     tErr1   = min(sum((time(1) - t).^2,2));
%     tErr2   = min(sum((time(2) - t).^2,2));
%     error   = w1*distto1 + w1*distto2+ w3*distto3+ w3*distto4 +
w3*distto5+ w3*distto6 + w2*tErr1 + w2*tErr2;
% end
%
% % myTwolinkwithprefilter function
% function dxdt = myTwolinkwithprefilter(t, x, wn, time, qDes, bj, kj)
%     zeta = 1.0;
%     A1 = [zeros([2 2]) eye(2); -eye(2)*wn(1)^2 -eye(2)*2*zeta*wn(1)];
%     B1 = [0 0; 0 0; wn(1)^2 0; 0 wn(1)^2];
%
```

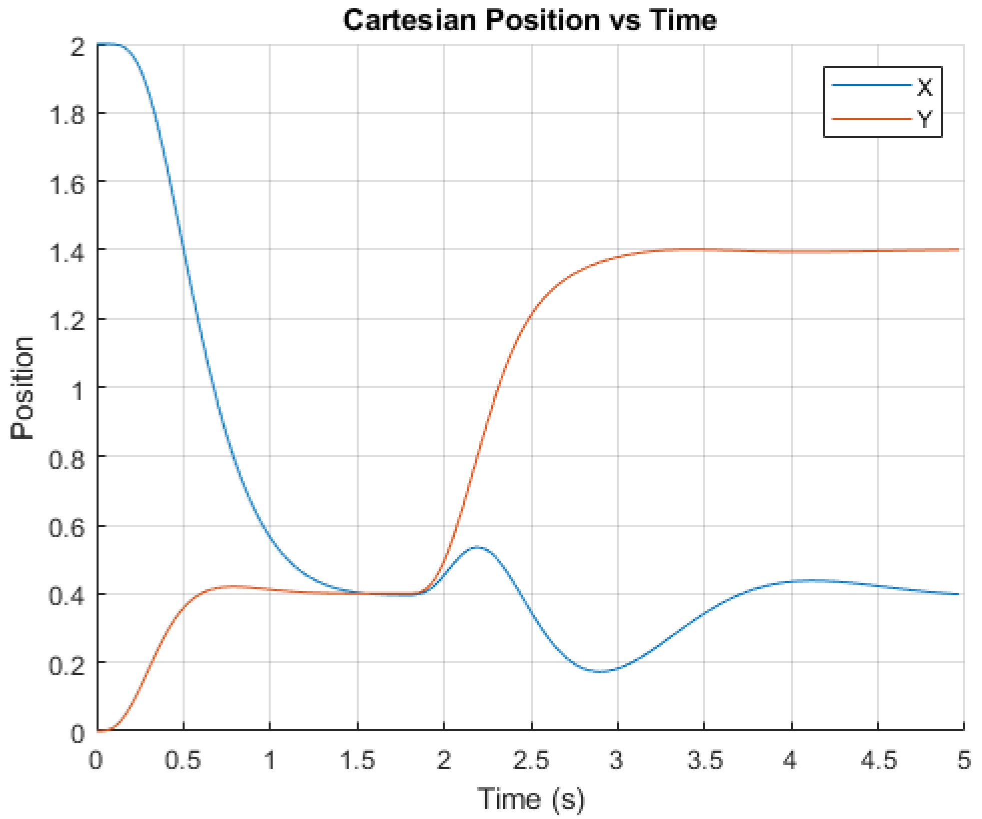
```
%     A2 = [zeros([2 2]) eye(2); -eye(2)*wn(2)^2 -eye(2)*2*zeta*wn(2)];
%     B2 = [0 0; 0 0; wn(2)^2 0; 0 wn(2)^2];
%
%     % Actual position and velocity
%     q = x(5:6);
%     qd = x(7:8);
%     q1p = x(7); q2p = x(8);
%     q1 = x(5); q2 = x(6);
%
%     % Robot constants
%     L_1 = 1; L_2 = 1; m_1 = 1; m_2 = 1;
%     ka = L_2^2 * m_2;
%     kb = L_1 * L_2 * m_2;
%     kc = L_1^2 * (m_1 + m_2);
%
%     M = [ka + 2*kb*cos(q2) + kc, ka + kb*cos(q2);
%          ka + kb*cos(q2), ka];
%     V = ka*sin(q2)*([0 -1; 1 0] * [q1p^2; q2p^2] + [-2*q1p*q2p; 0]);
%
%     Numerator = V + [-bj(1) 0; 0 -bj(2)]*qd + [-kj(1) 0; 0 -kj(2)]*(q -
x(1:2));
%
%     qdd = M\Numerator;
%
%     if t < time(1)
%         dotx = A1*x(1:4) + B1*qDes(1, :)';
%     else
%        dotx = A2*x(1:4) + B2*qDes(2, :)';
%     end
%     dxdt = [dotx; qd; qdd];
% end
```

```
Optimized Parameters :1.7772      4.96475      11.2111      10.2623
31.0696
```


Optimized Trajectory Tracking
Initial
Optimised
Desired
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Joint position (rad)
Q1
Q2
Position (rad)
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time (s)


Cartesian Position (m)


Cartesian Position vs Time

Published with MATLAB® R2023b