Table of Contents

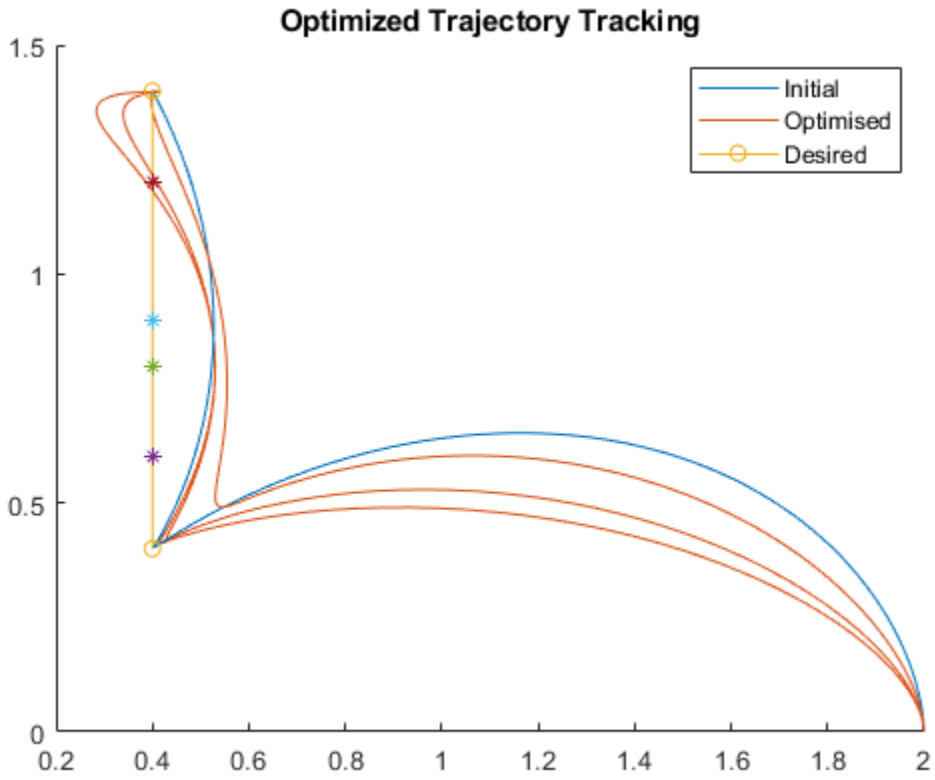

```
% Optimization of two-link robot arm tracking
clear; clc;

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

% Optimization setup
initParams = [10 20   1 20 45]; % 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   1    2  ];   % Lower bounds
ub = [3 6   50  200  500 ];  % Upper bounds

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

% 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    6.281479e+01    0.000e+00    3.626e+00
```

```
    1      12    6.111668e+01    0.000e+00    2.023e+00    2.840e+00
    2      18    6.107713e+01    0.000e+00    2.995e+06    6.470e-01
    3      26    6.109042e+01    0.000e+00    9.478e+00    7.679e-01
    4      34    6.098125e+01    0.000e+00    2.700e+00    6.771e-01
    5      47    6.070914e+01    0.000e+00    8.439e+00    1.665e-01
    6      58    6.045930e+01    0.000e+00    3.238e+00    1.857e-01
    7      80    6.045922e+01    0.000e+00    3.238e+00    8.809e-05
    8      95    6.045920e+01    0.000e+00    3.237e+00    1.928e-05
    9     113    6.045920e+01    0.000e+00    3.237e+00    5.599e-07
   10     127    6.045920e+01    0.000e+00    1.695e+06    2.450e-07

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.
```

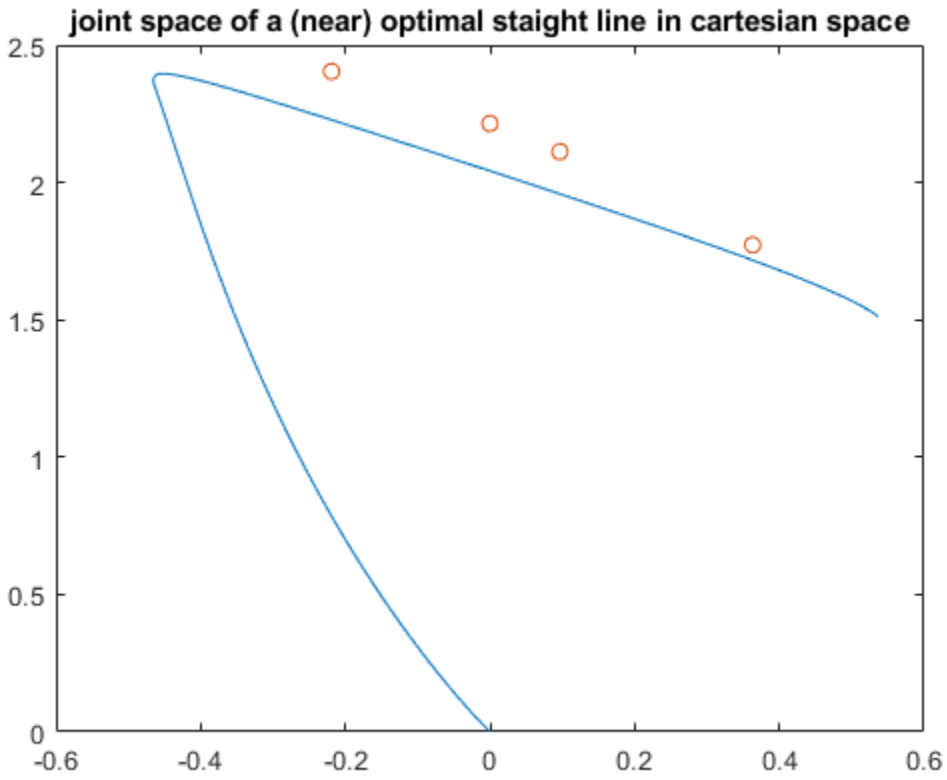
```
figure(1); hold on;
plot(xInit(:, 1), xInit(:, 2), '-');
plot(xAct(:, 1), xAct(:, 2), '-');
plot(xDes(:, 1), xDes(:, 2), 'o-');
plot(0.4,0.6, '*',0.4,0.8, '*',0.4,0.9, '*',0.4,1.2, '*');

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

```
Optimized Parameters :1.8607      5.89982      3.11489       19.203
46.0157
```



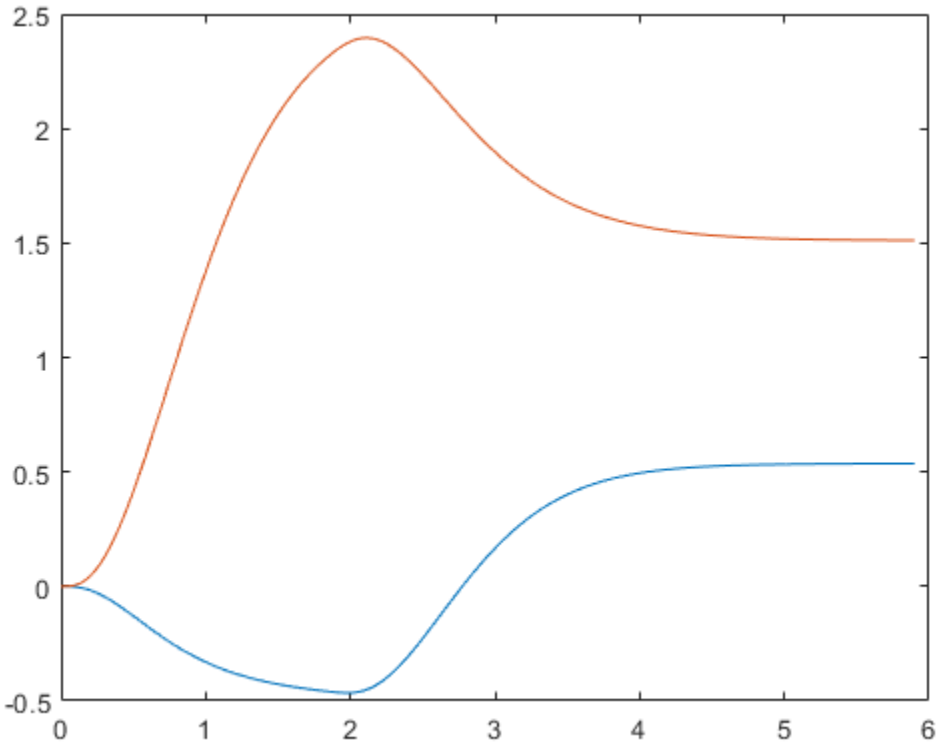
mid points in joint space

```
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')
```



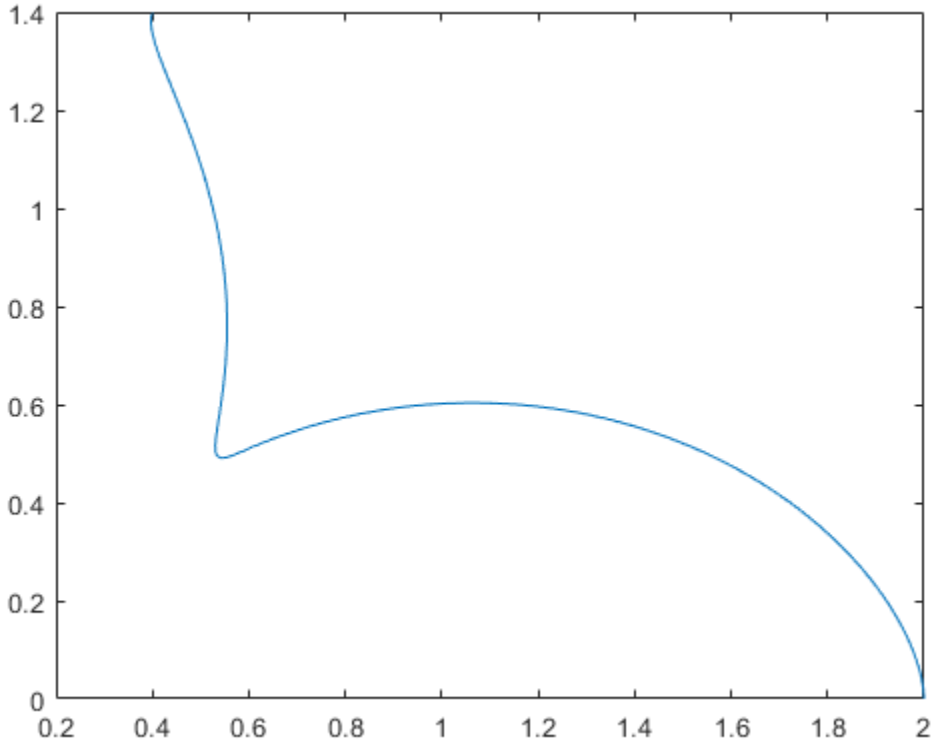
joint space plot

```
figure(5);plot(t,y(:,5:6));
```



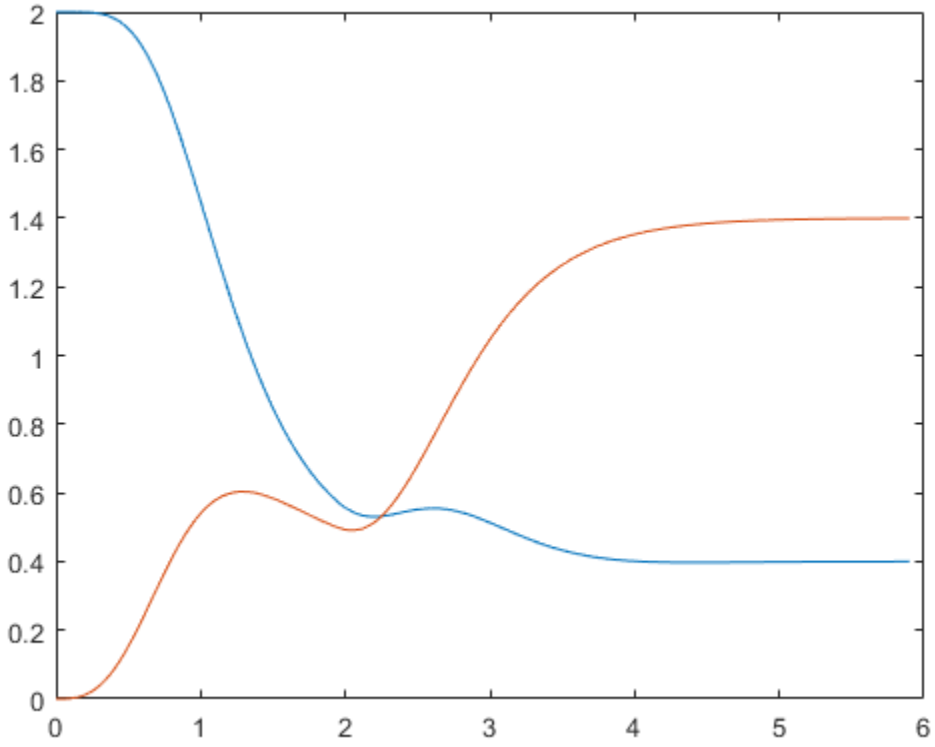
cartesian space plot

```
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)
    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,
kj), [0 time(end)], x0);

    % weights, could be done as a vector of weights
    w1 = 1000;
    w2 = 0;
    w3 = 2000;
    % w= [0.5, 1 , 5]; [qd_wt, time_wt, midpt_wt]
```

```
    % Middle Points
    qMid1 = inverse_kinematics(0.4, 0.6, 1, 1);
    qMid2 = inverse_kinematics(0.4, 0.8, 1, 1);
    qMid3 = inverse_kinematics(0.4, 0.9, 1, 1);
    qMid4 = inverse_kinematics(0.4, 1.2, 1, 1);

    % % Calculate the error metric
    % distto1 = w1 * sum((y(:, 5:6) - qDes(1,:)).^2,2) + w2 * (sum((time(1)
- t).^2,2));
    % distto3 = w3 * sum((y(:, 5:6) - qDes(2,:)).^2,2)    + w2 *
(sum((time(2) - t).^2,2));
    % distto4 = w3 * sum((y(:, 5:6) - qDes(3,:)).^2,2)       + w2 *
(sum((time(3) - t).^2,2));
    % distto5 = w3 * sum((y(:, 5:6) - qDes(4,:)).^2,2)       + w2 *
(sum((time(4) - t).^2,2));
    % distto6 = w3 * sum((y(:, 5:6) - qDes(5,:)).^2,2)       + w2 *
(sum((time(5) - t).^2,2));
    % distto2 = w1 * sum((y(:, 5:6) - qDes(6,:)).^2,2) + w2 * (sum((time(6)
- t).^2,2));
    % Calculate the error metric
    distto1 = min(sum((y(:, 5:6) - qDes(1,:)).^2,2));
    distto3 = min(sum((y(:, 5:6) - qMid1').^2,2));
    distto4 = min(sum((y(:, 5:6) - qMid2').^2,2));
    distto5 = min(sum((y(:, 5:6) - qMid3').^2,2));
    distto6 = min(sum((y(:, 5:6) - qMid4').^2,2));
    distto2 = min(sum((y(:, 5:6) - qDes(2,:)).^2,2));

    error   = w1*distto1 + w1*distto2+ w3*distto3+ w3*distto4 + w3*distto5+
w3*distto6;


    % error   = min(distto1) + min(distto2)+ min(distto5);
    % error   = min(distto1) + min(distto2);
    % 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
```

KY simOpt2 C:\Users\Koray\Documents\GitHub\twoLink_FrogExp\Control
30-Jan-2025 12:54:28

Published with MATLAB® R2023b