Jacobians and Cartesian Control

Jacobians, Joint Velocities, and Tip Velocities

A Jacobian

- Relates the tip forces to joint torques, and
- Converts from Cartesians (XYZ) motion to joint motion $(\theta_1, \theta_2, \theta_3)$

If the relationship between joint angles and tip position is

$$P = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f(\theta_1, \theta_2) \\ g(\theta_1, \theta_2) \end{bmatrix}$$

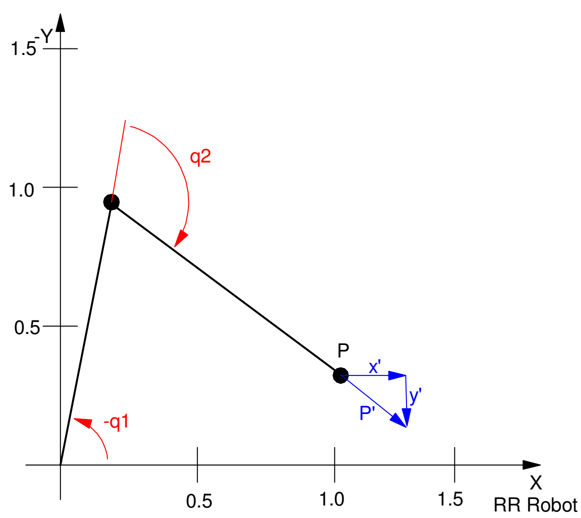
then the change in motion (velocity) is

$$\dot{P} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} & \frac{\partial f}{\partial \theta_2} \\ \frac{\partial g}{\partial \theta_1} & \frac{\partial g}{\partial \theta_2} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \text{n}$$

Jacobian is defined as

$$J(\theta_1, \theta_2) = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} & \frac{\partial f}{\partial \theta_2} \\ \frac{\partial g}{\partial \theta_1} & \frac{\partial g}{\partial \theta_2} \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$



Example: For the RR robot, the tip position is

$$x = c_1 + c_{12}$$

$$y = s_1 + s_{12}$$

The Jacobian is then

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} \end{bmatrix}$$

$$J = \begin{bmatrix} -s_1 - s_{12} & -s_{12} \\ c_1 + c_{12} & c_{12} \end{bmatrix}$$

Example: The tip position and joint angles of an RR robot is

$$P = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.0 \\ -0.3 \end{bmatrix}$$

$$Q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -1.3130 \\ 2.0432 \end{bmatrix}$$

If the joint velocities are

$$\dot{Q} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

then the tip velocity is

$$\dot{P} = J \cdot \dot{Q}$$

$$\dot{P} = \begin{bmatrix} 0.3 & -0.667 \\ 1 & 0.7451 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

$$\dot{P} = \begin{bmatrix} -0.1401 \\ 0.4253 \end{bmatrix}$$

If the tip velocity is

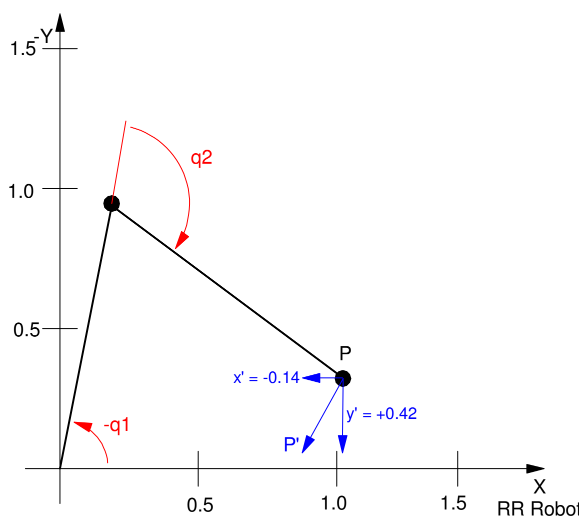
$$\dot{P} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

then the joint velocities must be

$$\dot{Q} = J^{-1} \dot{P}$$

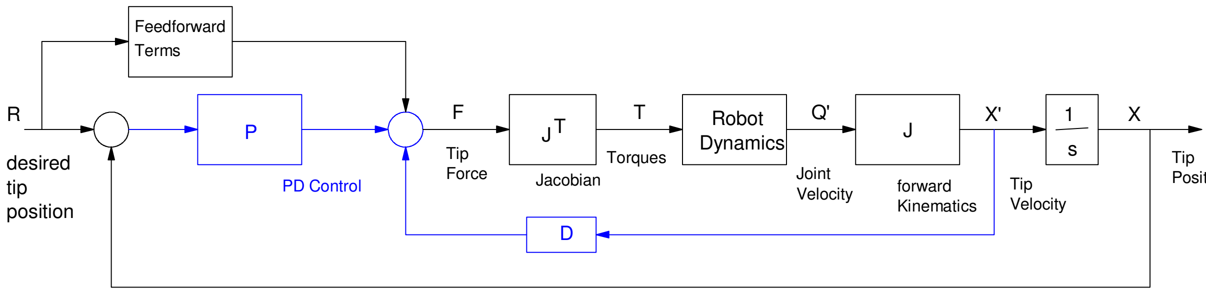
$$\dot{Q} = \begin{bmatrix} 0.3 & -0.667 \\ 1 & 0.7451 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\dot{Q} = \begin{bmatrix} 0.8367 \\ -1.1230 \end{bmatrix}$$

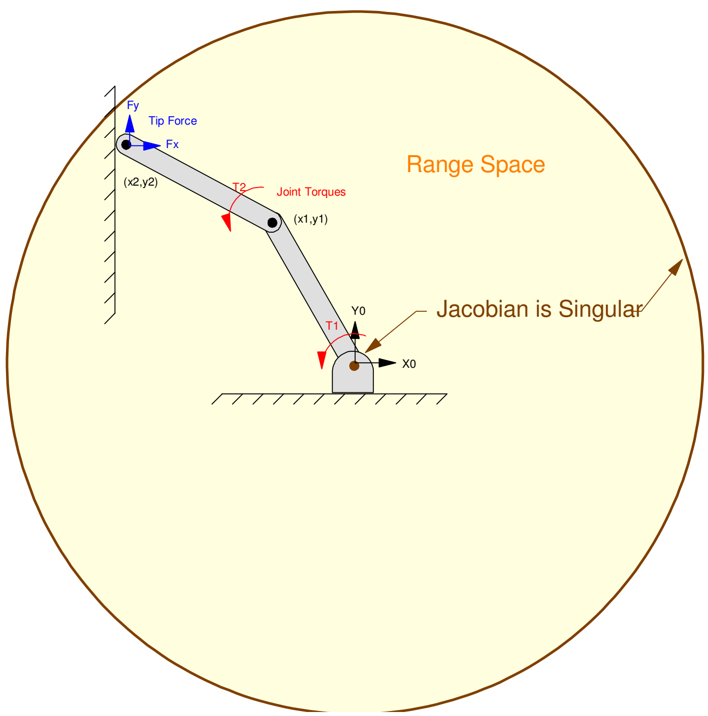


Jacobians and Cartesian Control

Jacobians also let you control the robot in tip (x,y) coordinates rather than joint angles.

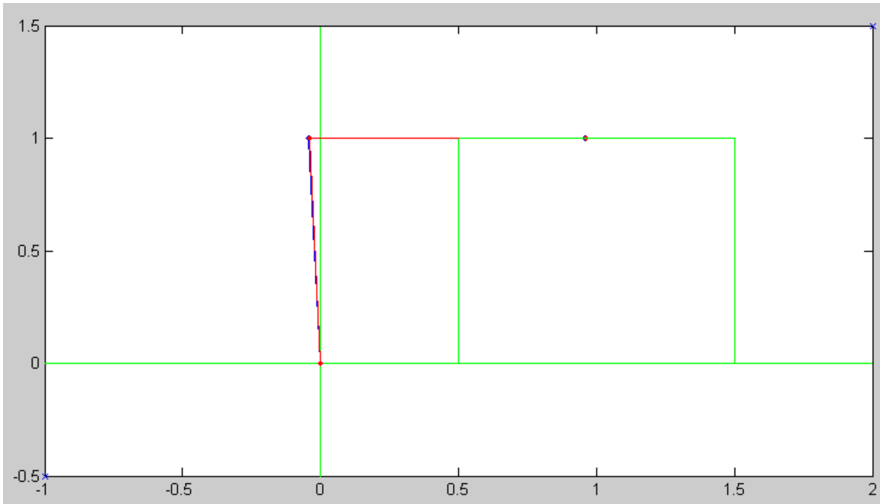


This has problems, however, if the Jacobian has singularities (i.e. the determinant is zero). At these points, you can't do XY control - i.e. avoid them.



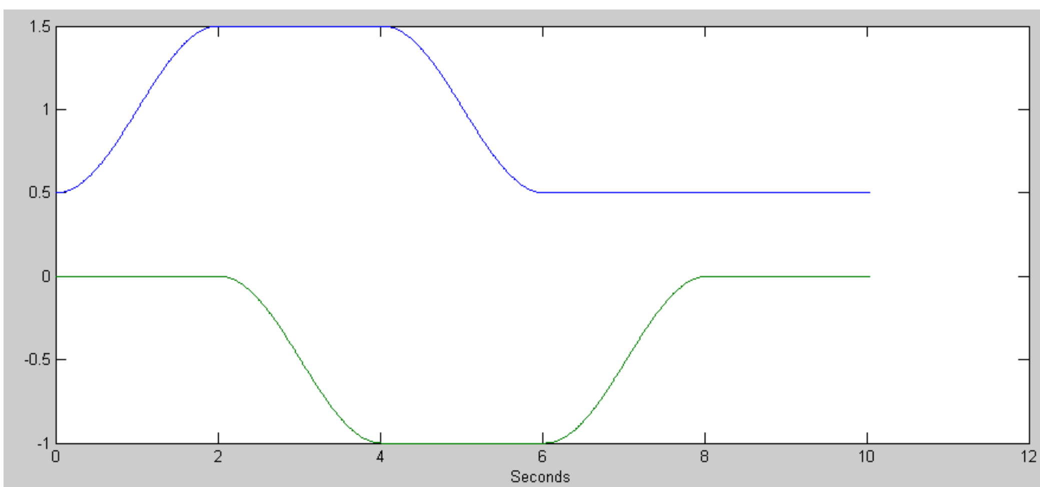
The Jacobian is singular at the edge of the range space (Q2 = 0 degrees) and at the origin (Q2 = 180 degrees) meaning that the joint velocities go to infinity as the tip position approaches these regions during path planning.

Example: Define the desired tip position to be the XY coordinates which trace out a square:



Tracing out a Square using Cartesian PD Control

Using cosine-interpolation with 2-second moves, the XY position vs. time should be:



Desired tip positions to trace out a square (blue = Xtip, green = Ytip)

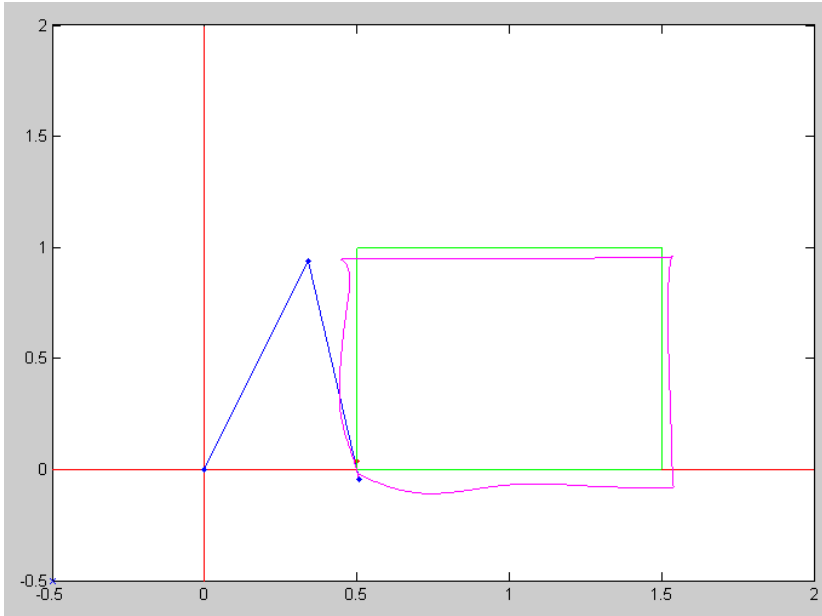
Case 1: PD Control

The force at the tip should be

$$T = J^T \left(P \begin{bmatrix} X_{ref} - X_{tip} \\ Y_{ref} - Y_{tip} \end{bmatrix} + D \begin{bmatrix} X_{ref} - \dot{X}_{tip} \\ Y_{ref} - \dot{Y}_{tip} \end{bmatrix} \right)$$

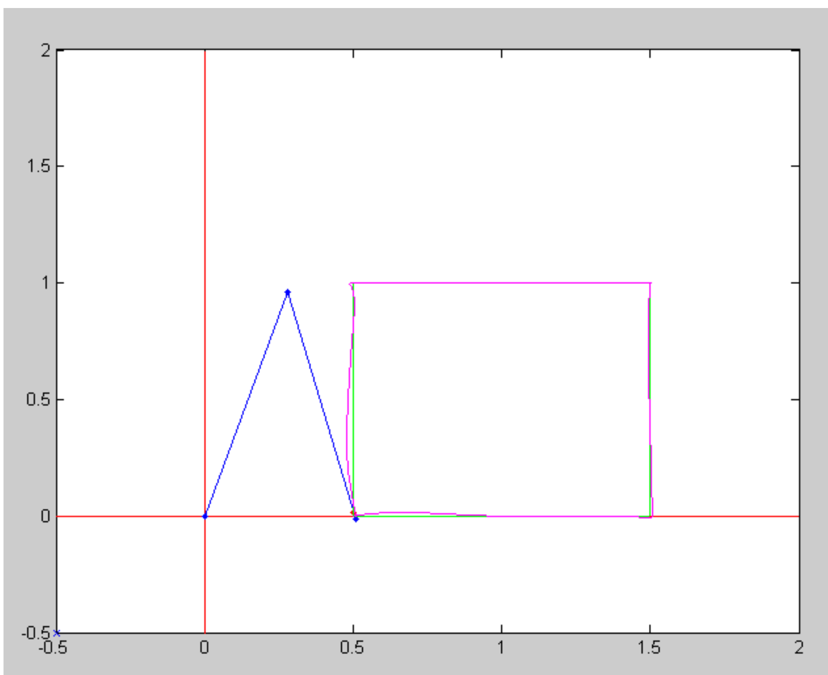
Pick P = 200 and D = 20 to place the closed-loop poles at $-10 \pm j10$

The resulting position vs. time is



Tracking with Cartesian Control using PD terms

Adding gravity compensation along with the acceleration in the tip position:



Cartesian Control with PD, Gravity, and Acceleration Terms

RR_XY_Control.m

```
% RR_XY_Control

 P0 = [0.5; 0];
 P1 = [1.5; 0];
 P2 = [1.5; 1];
 P3 = [0.5; 1];
 P4 = P0;

 T = 2;
 t = [0.01:0.01:T];
 a = (1 - cos(pi*t/T))/2;

 % for a step change in position, add the following line
 %a = 1*(a>0);

 X0 = P0*ones(1,50);
 X1 = P0*(1-a) + P1*a;
 X2 = P1*(1-a) + P2*a;
 X3 = P2*(1-a) + P3*a;
 X4 = P3*(1-a) + P4*a;
 X5 = P4*ones(1,50);
 Xr = [X0, X1, X2, X3, X4, X5];
 TIP = Xr;

 % tip velocity ( used for feedforward control )
 X2 = TIP(1,:);
 Y2 = TIP(2,:);

 dX2 = derivative(X2);
 dY2 = derivative(Y2);

 ddX2 = derivative(dX2);
 ddY2 = derivative(dY2);

 dXr = [dX2 ; dY2];
 ddXr = [ddX2 ; ddY2];

 Q = InverseRR(Xr(:,1));
 dQ = [0; 0];
 t = 0;
 dt = 0.001;

 % Start the simulation (dt = 0.001 for stability concerns)
 Xq = [];
 Tq = [];

 for i=1:length(Xr)
    Qr = InverseRR(Xr(:,i));
    for j=1:10
       c1 = cos(Q(1));
       s1 = sin(Q(1));
       c12 = cos(Q(1)+Q(2));
       s12 = sin(Q(1)+Q(2));

       X = [ c1 + c12 ;
             s1 + s12 ];
       J = [ -s1 - s12, -s12 ;
              c1 + c12,  c12 ];
       dX = J * dQ;
```

```
% Control Law and Feedforward Terms
       Facc = ddXr(:,i);
       Fpid = 100*(Xr(:,i) - X) + 20*(dXr(:,i)*0 - dX );

 % gravity
       Tg = -9.8 * [ 2*c1 + c12 ;  c12 ];

       T = (J') * ( Fpid + Facc*0 ) - Tg*0;

       ddQ = RRDynamics(Q, dQ, T);
       dQ = dQ + ddQ * dt;
       Q = Q + dQ*dt;
       t = t + dt;
       end

    RR(Q, Qr, TIP);

    Xq = [Xq, X];
    Tq = [Tq, T];
 end

 pause(5);

 t = [1:length(Xr)] * 0.01;
 clf
 subplot(211)
 plot(t,Xq,t,Xr);
 xlabel('Time (seconds)');
 ylabel('Tip (meters)');
 subplot(212)
 plot(t,Tq);
 xlabel('Time (seconds)');
 ylabel('Torque (Nm)');
```