Velocity Relationships

Jacobians

Motion and Force Control

Steven Dirven, Massey University, 2015

Reflection on Forward and Inverse Kinematics

So far, the following concepts have been discussed:

1. Assignment of Denavit-Hartenberg frames, and recording of the parameters $\theta_j, d_j, a_j, \alpha_j$ into a table for each joint
2. Forward Kinematics is the process of finding a robot's pose (location and orientation in space) based on the specified configuration of each joint
3. Inverse Kinematics is the process of finding the joint configurations (angles for revolute, length for prismatic) based a robot's specified pose

Reflection on Trajectory Generation

4. Trajectory Generation can be achieved by a number of methods. We visited traditional methods of:

- **Cartesian space** trajectory generation.
  - The pose matrix T was interpolated from the initial to final pose
  - The resulting joint angles were found by **inverse** kinematics
- **Joint space** trajectory generation.
  - The q joint vector was interpolated from the initial to final configurations
  - The resulting poses were found by **forward** kinematics

Velocity Relationships : Introduction

Forward and inverse kinematics are a great set of tools for describing the relationships between the pose of the robot (location and orientation of the end effector, T) and joint configurations (joint angles, q)

These describe the relationships between points expressed in different co-ordinate systems.

But what about if these points were not stationary, eg. they have some velocity. How can this behaviour be expressed in different frames?

Velocity Relationships : Introduction

This section develops a series of tools for investigating velocity within a frame, and then between frames. It consists of the following:

1. Developing relationships between linear and rotational velocities and their effect on that of the end-effector
2. Discovering the Jacobian matrix to transform velocity between frames
3. Using the inverse Jacobian to avoid inverse kinematics
4. Further investigation of the numerical method for inverse kinematics

Velocity Relationships : Introduction

In calculus, velocity with respect to time is the gradient of the displacement with respect to time.

It can be found by calculating the derivative of the displacement function with respect to time.

How do small variations in joint co-ordinates with respect to time affet the pose of the end-effector?

Is it possible to do something similar to the derivative?

Velocity Relationships : The First Order Difference

Suppose two different sets of joint configurations. Initially q, and then with some small change δ_q. For example: $(q + \delta_q)$

First Order Difference : Derivative Investigation

$$\frac{dT}{dq} \approx \frac{\text{Change in pose}}{\text{Change in joint angle}} \approx \frac{T(q + \delta_q) - T(q)}{\delta_q}$$

Next step is to calculate $T(q + \delta_q)$ and $T(q)$

Velocity Relationships : The First Order Difference

First Order Difference : Derivative Investigation

$$T(q) = \left[\begin{array}{c|c} \multirow{3}{*}{$R(q)$} & x_1 \\ & y_1 \\ & z_1 \\ \hline 000 & 1 \end{array}\right]$$

$$T(q+\delta_q) = \left[\begin{array}{c|c} \multirow{3}{*}{$R(q+\delta_q)$} & x_2 \\ & y_2 \\ & z_2 \\ \hline 000 & 1 \end{array}\right]$$

Substitute into:

$$\frac{dT}{dq} \approx \frac{T(q+\delta_q) - T(q)}{\delta_q}$$

Velocity Relationships : The First Order Difference

First Order Difference : Final Substitution

$$\frac{dT}{dq} \approx \frac{1}{\delta_q}\left(\left[\begin{array}{c|c} R(q+\delta_q) & \begin{array}{c} x_2 \\ y_2 \\ z_2 \end{array} \\ \hline 000 & 1 \end{array}\right] - \left[\begin{array}{c|c} R(q) & \begin{array}{c} x_1 \\ y_1 \\ z_1 \end{array} \\ \hline 000 & 1 \end{array}\right] \right)$$

where $x_2 - x_1 = \delta_x \quad y_2 - y_1 = \delta_y \quad z_2 - z_1 = \delta_z$

$$\frac{dT}{dq} \approx \frac{1}{\delta_q}\left[\begin{array}{c|c} R(\delta_q) & \begin{array}{c} \delta_x \\ \delta_y \\ \delta_z \end{array} \\ \hline 000 & 0 \end{array}\right]$$

and $(\delta_x, \delta_y, \delta_z)$ is the translational displacement of the next coordinate system

Velocity Relationships : The First Order Difference Example (Linear)

This process will be demonstated in MATLAB.

Consider the puma robot that moves from a nominal joint configuration to a new joint configuration

Joint 1 will be moved by a small angle $\delta_q = 1X10^{-6}$ rad.

First Order Difference : Example

$$\begin{aligned}
q &= [0 \quad 0.7854 \quad 3.1416 \quad 0 \quad 0.7854 \quad 0] \\
\delta_q &= [1x10^{-6} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \\
q + \delta_q &= [1x10^{-6} \quad 0.7854 \quad 3.1416 \quad 0 \quad 0.7854 \quad 0]
\end{aligned}$$

Velocity Relationships : The First Order Difference Example (Linear)

Velocity Relationships : The First Order Difference MATLAB Example

```
mdl_puma560
Ti = p560.fkine(qn)
dq = 1e-6
Tf = p560.fkine(qn+[dq 0 0 0 0 0])
dTdq1 = (Tf-Ti)/dq
```

```
dTdq1 =
         0   -1.0000   -0.0000    0.1500
   -0.0000   -0.0000    1.0000    0.5963
         0         0         0         0
         0         0         0         0
```

Observations: The matrix is no longer a homogeneous transform

- The bottom right element is now zero, not a one
- The upper left 3X3 matrix is no longer orthonormal

Velocity Relationships : The First Order Difference Example (Linear)

Example matrix from previous slide

First Order Difference : Evaluating the $\delta_x, \delta_y, \delta_z$ components

$$\frac{dT}{dq_1} \approx \frac{1}{\delta_q} \left[\begin{array}{c|c} & \delta_x \\ R(\delta_q) & \delta_y \\ & \delta_z \\ \hline 000 & 0 \end{array} \right] = \left[\begin{array}{ccc|c} 0 & -1 & 0 & 0.1500 \\ 0 & 0 & 1 & 0.5963 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array} \right]$$

Equate the components $\delta_x, \delta_y, \delta_z$ in the matrices. Realise that each of these components has already been divided by δ_q so each element must be multiplied by this factor.

$$\therefore \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} = \begin{bmatrix} 0.1500 \\ 0.5963 \\ 0 \end{bmatrix} \delta_{q1}$$

Velocity Relationships : First Order Difference Example 2 (Linear)

What happens if joint 2 was perturbed by the same small finite amount of $1x10^{-6}$ rad?

Notice that $1x10^{-6}$ is in the second column now.

First Order Difference : Example

$$\begin{aligned}
q &= [0 \quad 0.7854 \quad 3.1416 \quad 0 \quad 0.7854 \quad 0] \\
\delta_q &= [0 \quad 1x10^{-6} \quad 0 \quad 0 \quad 0 \quad 0] \\
q + \delta_q &= [0 \quad 0.7854 + 1x10^{-6} \quad 3.1416 \quad 0 \quad 0.7854 \quad 0]
\end{aligned}$$

Velocity Relationships : First Order Difference Example 2 (Linear)

Velocity Relationships : First Order Difference MATLAB Example 2

```
mdl_puma560
Ti = p560.fkine(qn)
dq = 1e-6
Tf = p560.fkine(qn+[0 dq 0 0 0 0])
dTdq1 = (Tf-Ti)/dq
```

```
dTdq2 =
    1.0000   -0.0000   -0.0000    0.0144
    0.0000         0    0.0000         0
    0.0000    0.0000    1.0000    0.5963
         0         0         0         0
```

Notice that this component rotates about the global y axis.

Velocity Relationships : First Order Difference Example 2 (Linear)

Example matrix from previous slide

First Order Difference : Evaluate the $\delta_x, \delta_y, \delta_z$ Components

$$\frac{dT}{dq_2} \approx \frac{1}{\delta_q} \left[\begin{array}{c|c} & \delta_x \\ R(\delta_q) & \delta_y \\ & \delta_z \\ \hline 000 & 0 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0.0144 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.5963 \\ \hline 0 & 0 & 0 & 0 \end{array}\right]$$

Equate the components $\delta_x, \delta_y, \delta_z$ in the matrices. Realise that each of these components has already been divided by δ_q so each element must be multiplied by this factor.

$$\therefore \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} = \begin{bmatrix} 0.0144 \\ 0 \\ 0.5963 \end{bmatrix} \delta_{q2}$$

Velocity Relationships : First Order Difference Example 1 and 2 : Divide by small finite time step (Linear)

Suppose that the small angle δ_q was turned through in a finite time δ_t. If we divide these matrices element-wise we can get linear velocity components in terms of rotational velocity components : $v = k\omega$

First Order Difference : Small Time Step

$$\frac{1}{\delta_t}\begin{bmatrix}\delta_x\\ \delta_y\\ \delta_z\end{bmatrix} = \begin{bmatrix}0.1500\\ 0.5963\\ 0\end{bmatrix}\frac{\delta_{q1}}{\delta_t} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}0.1500\\ 0.5963\\ 0\end{bmatrix}\dot{q}_1$$

$$\frac{1}{\delta_t}\begin{bmatrix}\delta_x\\ \delta_y\\ \delta_z\end{bmatrix} = \begin{bmatrix}0.0144\\ 0\\ 0.5963\end{bmatrix}\frac{\delta_{q2}}{\delta_t} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}0.0144\\ 0\\ 0.5963\end{bmatrix}\dot{q}_2$$

Velocity Relationships : First Order Difference (Rotation)

First order difference of T from earlier.

Notice that the top left 3x3 matrix is equal to : $\frac{R(\delta_q)}{\delta_q}$

Can investigate the behaviour of this matrix in a similar manner to the linear component as before

First Order Difference : Investigating Rotation

$$\frac{dT}{dq_1} \approx \frac{1}{\delta_q} \left[\begin{array}{c|c} & \delta_x \\ R(\delta_q) & \delta_y \\ & \delta_z \\ \hline 000 & 0 \end{array} \right] = \left[\begin{array}{ccc|c} 0 & -1 & 0 & 0.1500 \\ 0 & 0 & 1 & 0.5963 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array} \right]$$

Velocity Relationships : First Order Difference (Rotation)

First Order Difference : Investigating Rotation

$$\frac{dR}{dq_1} \approx \frac{1}{\delta_q}\left[\; R(\delta_q) \;\right] = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{dR}{dt_1} \approx \frac{1}{\delta_t}\left[\; R(\delta_q) \;\right] = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \frac{\delta_{q1}}{\delta_t}$$

Substitute $\dot{R} = S(\omega)R$

$$S(\omega)R \approx \frac{1}{\delta_t}\left[\; R(\delta_q) \;\right] = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \frac{\delta_{q1}}{\delta_t}$$

Velocity Relationships : First Order Difference (Rotation)

First Order Difference : Investigating Rotation

$$S(\omega)R \approx \frac{1}{\delta_t}\left[\; R(\delta_q) \;\right] = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \frac{\delta_{q1}}{\delta_t}$$

Post Multiply by R^T

$$S(\omega) \approx \frac{1}{\delta_t}\left[\; R(\delta_q) \;\right]\left[\; R_{initial} \;\right]^T = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \frac{\delta_{q1}}{\delta_t}$$

where $\frac{\delta_{q1}}{\delta_t} = \dot{q}_1$

$$S(\omega) \approx \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{q}_1 \approx \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ \omega_y & \omega_x & 0 \end{bmatrix} \qquad \therefore \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{q}_1$$

Velocity Relationships : First Order Difference (Rotation)

First Order Difference : Investigating Rotation

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{q}_1$$

Observations:

- It makes sense that the rotation of joint 1 only exhibits rotation in the global Z axis, as this is the axis of the first joint.
- Because q_1 is a revolute joint measured in radians, its derivative is the rate of rotation eg. $\dot{q}_1$ can be thought of as ω_1, a rotational velocity about joint 1

Velocity Relationships : Introduction to the Jacobian

The **Jacobian** is:

- a matrix of partial derivatives of linear and rotational velocities with respect to the joint variables.
- an **instantaneous** snapshot of the contribution that each joint plays in making up the robots global linear and rotational velocities.
- a useful tool for investigating dynamic aspects of serial robotic motion

The rows correspond to cartesian degrees of freedom, and the columns correspond to the joints.

Velocity Relationships : The Jacobian

The process that has been demonstrated for first order differences has been used to find 2 vectors. They describe the instantaneous linear velocity and rotational velocity ω (using the skew matrix definition).

The Jacobian : Find Global Linear and Rotational Velocity

$$\nu = J(q)\dot{q}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{\delta x_1}{\delta q_1} & \frac{\delta x_2}{\delta q_2} & \frac{\delta x_3}{\delta q_3} & \frac{\delta x_4}{\delta q_4} & \frac{\delta x_5}{\delta q_5} & \frac{\delta x_6}{\delta q_6} \\ \frac{\delta y_1}{\delta q_1} & \frac{\delta y_2}{\delta q_2} & \frac{\delta y_3}{\delta q_3} & \frac{\delta y_4}{\delta q_4} & \frac{\delta y_5}{\delta q_5} & \frac{\delta y_6}{\delta q_6} \\ \frac{\delta z_1}{\delta q_1} & \frac{\delta z_2}{\delta q_2} & \frac{\delta z_3}{\delta q_3} & \frac{\delta z_4}{\delta q_4} & \frac{\delta z_5}{\delta q_5} & \frac{\delta z_6}{\delta q_6} \\ \frac{\delta \alpha_1}{\delta q_1} & \frac{\delta \alpha_2}{\delta q_2} & \frac{\delta \alpha_3}{\delta q_3} & \frac{\delta \alpha_4}{\delta q_4} & \frac{\delta \alpha_5}{\delta q_5} & \frac{\delta \alpha_6}{\delta q_6} \\ \frac{\delta \beta_1}{\delta q_1} & \frac{\delta \beta_2}{\delta q_2} & \frac{\delta \beta_3}{\delta q_3} & \frac{\delta \beta_4}{\delta q_4} & \frac{\delta \beta_5}{\delta q_5} & \frac{\delta \beta_6}{\delta q_6} \\ \frac{\delta \gamma_1}{\delta q_1} & \frac{\delta \gamma_2}{\delta q_2} & \frac{\delta \gamma_3}{\delta q_3} & \frac{\delta \gamma_4}{\delta q_4} & \frac{\delta \gamma_5}{\delta q_5} & \frac{\delta \gamma_6}{\delta q_6} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix}$$

Velocity Relationships : The Jacobian : Meaning

Multiplying the Jacobian by the derivatives (joint velocities) of each of the joints expresses the instantaneous relationship between the joint velocities and their effects on linear and rotaional velocities in the base frame.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{\delta x_1}{\delta q_1} & \frac{\delta x_2}{\delta q_2} & \frac{\delta x_3}{\delta q_3} & \frac{\delta x_4}{\delta q_4} & \frac{\delta x_5}{\delta q_5} & \frac{\delta x_6}{\delta q_6} \\ \frac{\delta y_1}{\delta q_1} & \frac{\delta y_2}{\delta q_2} & \frac{\delta y_3}{\delta q_3} & \frac{\delta y_4}{\delta q_4} & \frac{\delta y_5}{\delta q_5} & \frac{\delta y_6}{\delta q_6} \\ \frac{\delta z_1}{\delta q_1} & \frac{\delta z_2}{\delta q_2} & \frac{\delta z_3}{\delta q_3} & \frac{\delta z_4}{\delta q_4} & \frac{\delta z_5}{\delta q_5} & \frac{\delta z_6}{\delta q_6} \\ \frac{\delta \alpha_1}{\delta q_1} & \frac{\delta \alpha_2}{\delta q_2} & \frac{\delta \alpha_3}{\delta q_3} & \frac{\delta \alpha_4}{\delta q_4} & \frac{\delta \alpha_5}{\delta q_5} & \frac{\delta \alpha_6}{\delta q_6} \\ \frac{\delta \beta_1}{\delta q_1} & \frac{\delta \beta_2}{\delta q_2} & \frac{\delta \beta_3}{\delta q_3} & \frac{\delta \beta_4}{\delta q_4} & \frac{\delta \beta_5}{\delta q_5} & \frac{\delta \beta_6}{\delta q_6} \\ \frac{\delta \gamma_1}{\delta q_1} & \frac{\delta \gamma_2}{\delta q_2} & \frac{\delta \gamma_3}{\delta q_3} & \frac{\delta \gamma_4}{\delta q_4} & \frac{\delta \gamma_5}{\delta q_5} & \frac{\delta \gamma_6}{\delta q_6} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix}$$

Velocity Relationships : The Jacobian

Refreshing the Earlier Expressions

Earlier, the following expressions were derived:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}0.1500\\ 0.5963\\ 0\end{bmatrix}\dot{q}_1 \quad where \quad \begin{bmatrix}0.1500\\ 0.5963\\ 0\end{bmatrix} = \begin{bmatrix}\frac{\delta x_1}{\delta q_1}\\ \frac{\delta y_1}{\delta q_1}\\ \frac{\delta z_1}{\delta q_1}\end{bmatrix}$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}0.0144\\ 0\\ 0.5963\end{bmatrix}\dot{q}_2 \quad where \quad \begin{bmatrix}0.0144\\ 0\\ 0.5963\end{bmatrix} = \begin{bmatrix}\frac{\delta x_2}{\delta q_2}\\ \frac{\delta y_2}{\delta q_2}\\ \frac{\delta z_2}{\delta q_2}\end{bmatrix}$$

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}0\\ 0\\ 1\end{bmatrix}\dot{q}_1 \quad where \quad \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} = \begin{bmatrix}\frac{\delta \alpha_1}{\delta q_1}\\ \frac{\delta \beta_1}{\delta q_1}\\ \frac{\delta \gamma_1}{\delta q_1}\end{bmatrix}$$

These can be substituted into the Jacobian matrix

Velocity Relationships : Filling the Jacobian Matrix

Substitute Vectors from Previous Slide into Jacobian

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0.1500 & 0.0144 & \frac{\delta x_3}{\delta q_3} & \frac{\delta x_4}{\delta q_4} & \frac{\delta x_5}{\delta q_5} & \frac{\delta x_6}{\delta q_6} \\ 0.5963 & 0 & \frac{\delta y_3}{\delta q_3} & \frac{\delta y_4}{\delta q_4} & \frac{\delta y_5}{\delta q_5} & \frac{\delta y_6}{\delta q_6} \\ 0 & 0.5963 & \frac{\delta z_3}{\delta q_3} & \frac{\delta z_4}{\delta q_4} & \frac{\delta z_5}{\delta q_5} & \frac{\delta z_6}{\delta q_6} \\ 0 & \frac{\delta \alpha_2}{\delta q_2} & \frac{\delta \alpha_3}{\delta q_3} & \frac{\delta \alpha_4}{\delta q_4} & \frac{\delta \alpha_5}{\delta q_5} & \frac{\delta \alpha_6}{\delta q_6} \\ 0 & \frac{\delta \beta_2}{\delta q_2} & \frac{\delta \beta_3}{\delta q_3} & \frac{\delta \beta_4}{\delta q_4} & \frac{\delta \beta_5}{\delta q_5} & \frac{\delta \beta_6}{\delta q_6} \\ 1 & \frac{\delta \gamma_2}{\delta q_2} & \frac{\delta \gamma_3}{\delta q_3} & \frac{\delta \gamma_4}{\delta q_4} & \frac{\delta \gamma_5}{\delta q_5} & \frac{\delta \gamma_6}{\delta q_6} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix}$$

To fill the rest of the Jacobian, the process demonstrated from slide 10-19 would need to be repeated for each joint.

Velocity Relationships : Filling the Jacobian Matrix

Now that the principle has been shown, how can this be done automatically using the Peter Corke Toolbox in MATLAB

Finding the Jacobian in the Base Frame MATLAB PCT

```
mdl_puma560
j = p560.jacob0(qn)
```

```
j =
    0.1501    0.0144    0.3197         0         0         0
    0.5963    0.0000    0.0000         0         0         0
         0    0.5963    0.2910         0         0         0
         0    0.0000    0.0000    0.7071    0.0000    1.0000
    0.0000   -1.0000   -1.0000   -0.0000   -1.0000   -0.0000
    1.0000    0.0000    0.0000   -0.7071    0.0000   -0.0000
```

Velocity Relationships : Filling the Jacobian Matrix

Notice that the 3x3 matrix in the top right is all zeros. This is because the PUMA 560 robot has a spherical wrist. This means that there are no translational velocity components from these joints in the global cartesian space.

```
j =
    0.1501    0.0144    0.3197         0         0         0
    0.5963    0.0000    0.0000         0         0         0
         0    0.5963    0.2910         0         0         0
         0    0.0000    0.0000    0.7071    0.0000    1.0000
    0.0000   -1.0000   -1.0000   -0.0000   -1.0000   -0.0000
    1.0000    0.0000    0.0000   -0.7071    0.0000   -0.0000
```

Velocity Relationships : Transforming Velocities Between Co-ordinate Systems

The Jacobian in Velocity Transformation

It is known that two frames {A} and {B} can be related by:

$$^{A}T_{B} = \begin{bmatrix} ^{A}R_{B} & ^{A}t_{B} \\ 0 & 1 \end{bmatrix}$$

To transform the velocity from frame {A} to {B} the Jacobian is used in the following manner:

$$^{B}\nu =^{B} J_{A} \, ^{A}\nu$$

A substitution is made for $^{B}J_{A}$ as follows:

$$^{B}J_{A} = J_{v}(^{A}T_{B}) = \begin{bmatrix} ^{B}R_{A} & 0_{3x3} \\ 0_{3x3} & ^{B}R_{A} \end{bmatrix}$$

Velocity Relationships : The Jacobian Expressed in the End-effector Frame

The formula on the previous slide is used to relate the spatial velocity of a point expressed in frame {A} in the frame {B}

However, if the frames are on the same rigid body, then the following substitution is made:

The Jacobian in Velocity Transformation

$$^{B}J_{A} = \bar{J}_{v}(^{A}T_{B}) = \begin{bmatrix} ^{B}R_{A} & -^{B}R_{A}S(^{A}t_{B}) \\ 0_{3x3} & ^{B}R_{A} \end{bmatrix}$$

Velocity Relationships : Transforming Velocities Between Co-ordinate Systems

When calculating the Jacobian by hand earlier, it was found relative to the base frame

Jacobian in Base Frame

```
mdl_puma560
p560.Jacob0(qn).
```

The Jacobian can also be found in the end effector frame:

Jacobian in End-effector Frame

```
mdl_puma560
p560.Jacobn(qn)
```

Velocity Relationships : Numerical Jacobian Summary

The **Jacobian is a matrix of partial derivatives**, expressing the capability of each joint to achieve **linear and rotational velocity** in the base frame based on joint variable rates. (In this case a 6R PUMA robot. However, q_i could be a planar joint)

The Jacobian can be found by the first difference method.

It is only instantaneously true, as soon as the robot moves, the Jacobian needs to be recalculated.

Velocity Relationships : Analytical Jacobian

Earlier, spatial velocity was expressed in terms of translation and angular velocity. Sometimes, it is more intuitive to consider roll, pitch, yaw angles, which can be done analytically.

Finding the Analytical Jacobian

$$\Gamma = (\theta_r, \theta_p, \theta_y)$$

$$R = R_x(\theta_r) R_y(\theta_p) R_z(\theta_y) =$$

$$\begin{bmatrix} c\theta_p c\theta_y & -c\theta_p s\theta_y & s\theta_p \\ c\theta_r s\theta_y + c\theta_y s\theta_p s\theta_r & -s\theta_p s\theta_r s\theta_y + c\theta_r c\theta_y & -c\theta_p s\theta_r \\ s\theta_r s\theta_y - c\theta_r c\theta_y s\theta_p & c\theta_r s\theta_p s\theta_y + c\theta_y s\theta_r & c\theta_p c\theta_r \end{bmatrix}$$

Velocity Relationships : Analytical Jacobian

Velocity Relationships : Analytical Jacobian

$$\dot{R} = s(\omega)R$$

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} s\theta_p\dot{\theta}_y + \dot{\theta}_r \\ -c\theta_p s\theta_r\dot{\theta}_y + c\theta_r\dot{\theta}_p \\ c\theta_p c\theta_r\dot{\theta}_y + s\theta_r\dot{\theta}_p \end{bmatrix}$$

which can be factored as:

$$\omega = \begin{bmatrix} 1 & 0 & s\theta_p \\ 0 & c\theta_r & -c\theta_p s\theta_r \\ 0 & s\theta_r & c\theta_p c\theta_r \end{bmatrix} \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_p \\ \dot{\theta}_y \end{bmatrix}$$

and expressed as:

$$\omega = B(\Gamma)\dot{\Gamma}$$

Velocity Relationships : Analytical Jacobian

Velocity Relationships : Analytical Jacobian J_a

$$J_a(q) = \begin{bmatrix} I_{3x3} & 0_{3x3} \\ 0_{3x3} & B^{-1}(\Gamma) \end{bmatrix} J(q)$$

This is only true when B is not singular. It does however become singular when cos ϕ = 0 or pitch angle ϕ = $\pm\pi/2$

Velocity Relationships : Jacobian Condition

The process of investigating the Jacobian has been demonstrated for finding cartesian velocity components based on joint angle rates.

The inverse problem is equally interesting : How can a cartesian velocity relationship be specified, and the joint rates calculated?

That is, can the expression $\nu = J(q)\dot{q}$ be rearranged to find $\dot{q}$ based on knowing the jacobian and cartesian velocity components?

Velocity Relationships : Jacobian Condition

Rearrangement of Cartesian Velocities to Find Joint Velocities

$$\nu = J(q)\dot{q}$$

Premultiply by $J(q)^{-1}$

$$J(q)^{-1}\nu = J(q)^{-1}J(q)\dot{q} = I\dot{q} = \dot{q}$$

$$\dot{q} = J(q)^{-1}\nu$$

To find the inverse Jacobian, the Jacobian must be square and non-singular.

Velocity Relationships : Jacobian Condition

An n-link robot's Jacobian is a 6xn matrix, so a square Jacobian requires a robot with 6 joints. A robotic configuration q, at which $det(J(Q)) = 0$ is described as singular or degenerate.

Eg. When PUMA joints 4&6 one DOF is lost. The Jacobian has 2 equal columns. It is rank deficient:

Finding the Jacobian in the Base Frame at Pose qr

```
J = p560.jacob0(qr)
```

Velocity Relationships : Jacobian Condition

An n-link robot's Jacobian is a 6xn matrix, so a square Jacobian requires a robot with 6 joints. A robotic configuration q, at which $det(J(Q)) = 0$ is described as singular or degenerate.

Eg. When PUMA joints 4&6 one DOF is lost. The Jacobian has 2 equal columns. It is rank deficient:

```
J =
    0.1500   -0.8636   -0.4318         0         0         0
    0.0203    0.0000    0.0000         0         0         0
         0    0.0203    0.0203         0         0         0
         0         0         0         0         0         0
         0   -1.0000   -1.0000         0   -1.0000         0
    1.0000    0.0000    0.0000    1.0000    0.0000    1.0000
```

Velocity Relationships : Jacobian Condition

When a robot is near a singularity, but not at it, it can result in high joint rates being required for low velocity cartesian motion.

Examination of Jacobian : PUMA Example MATLAB PCT

```
q = qr
q(5) = 5*pi/180 % investigate at 5 degrees
J = p560.jacob0(q)
qd = inv(J) * [0 0 0.1 0 0 0]'
qd'
```

```
qd' =

    -0.0000   -4.9261    9.8522    0.0000   -4.9261         0
```

eg. 9.85 $rads^{-1}$ is approx $600^o s^{-1}$

Velocity Relationships : Jacobian Condition

The **determinant** and **condition score** are two methods to evaluate how close a robot is to a singularity, where there is poor condition and manipulability.

At a **singularity**, the **determinant of the Jacobian is zero**. Thus, the determinant tends to zero, as singularities are approached.

On the other hand, the **condition score** is high when the Jacobian is poorly conditioned.

Velocity Relationships : Jacobian Condition

What are the magnitudes of the determinant and condition score for the PUMA robot when in the joint configuration approaches a singularity?

Examination of Jacobian : Determinant and Condition Score Example MATLAB PCT

```
q = qr
q(5) = 5*pi/180 % investigate at 5 degrees
J = p560.jacob0(q)
det(J)
cond(J)
```

$det(J) = -1.5509 * 10^{-5}$ (very small)
$cond(J) = 235.2498$ (poorly conditioned)

A magnitude near 1 for the Jacobian condition means that each DOF are equally dextrous.

Velocity Relationships : Jacobians for Under Actuated and Redundantly Actuated Robots

For a Jacobian matrix to have an inverse, it needs to be square.

However, if the robot is under actuated or redundantly actuated, the jacobian is rectangular
(either too few columns, or too many columns eg. not 6)

$$\begin{bmatrix} \nu \\ \\ X \\ X \\ X \end{bmatrix} = \begin{bmatrix} & J(q) & \\ \\ X & X & X \\ X & X & X \\ X & X & X \end{bmatrix} \begin{bmatrix} \dot{q} \end{bmatrix} \qquad \begin{bmatrix} \\ \nu \\ \\ \end{bmatrix} = \begin{bmatrix} & & X \\ & & X \\ J(q) & & X \\ & & X \\ & & X \\ & & X \end{bmatrix} \begin{bmatrix} \dot{q} \\ \\ \\ X \end{bmatrix}$$

What does this mean for the Jacobian, and how can this be dealt with?

Velocity Relationships : Jacobians for Under Actuated and Redundantly Actuated Robots

For a Jacobian matrix to have an inverse, it needs to be square.

However, if the robot is under actuated or redundantly actuated, the jacobian is rectangular
(either too few columns, or too many columns eg. not 6)

$$\begin{bmatrix} \nu \\ \\ X \\ X \\ X \end{bmatrix} = \begin{bmatrix} & J(q) & \\ & & \\ X & X & X \\ X & X & X \\ X & X & X \end{bmatrix} \begin{bmatrix} \dot{q} \end{bmatrix} \qquad \begin{bmatrix} \\ \nu \\ \\ \end{bmatrix} = \begin{bmatrix} & & X \\ & & X \\ J(q) & & X \\ & & X \\ & & X \\ & & X \end{bmatrix} \begin{bmatrix} \dot{q} \\ \\ \\ X \end{bmatrix}$$

What does this mean for the Jacobian, and how can this be dealt with? Ignore parts of matrix eg (X’s)

Velocity Relationships : Jacobians for Under Actuated Robots : Investigation

For under-actuated robots there are fewer than 6 DOF, thus, some DOF need to be ignored.

If all DOF are specified, then the MATLAB solver will find a solution with mimimum squared error in each of the 6 spatial DOF.

However, the Jacobian can be used to force a prescribed number of them to be perfectly accurate. The technique is to specify the number of independant DOF that the robot has, and allow the resultant location and orientation components to take an arbitrary (uncontrolled) magnitude.

Velocity Relationships : Jacobians for Under Actuated Robots : 2-Link Example MATLAB PCT

First, investigate what happens if all DOF are attempted to be controlled

Jacobians for Under Actuated Robots : 2-Link Example MATLAB PCT

```
mdl_twolink
qn = [1 1]
J = jacob0(twolink,qn)
qd = pinv(J)*[0.1 0 0 0 0 0]'
xd = J*qd
```

```
J =
   -1.7508   -0.9093
    0.0000   -0.0000
    0.1242   -0.4161
         0         0
   -1.0000   -1.0000
    0.0000    0.0000
```

```
qd =
   -0.0698
    0.0431
```

```
xd =
    0.0829
   -0.0000
   -0.0266
         0
    0.0266
   -0.0000
```

Velocity Relationships : Jacobians for Under Actuated Robots : 2-Link Example MATLAB PCT

This has motion in x, but not enough

It also has motion in y and z, which were not commanded.

This can be overcome if two degrees of freedom are commanded, and the others are left to take an arbitrary magnitude.

```
J =                               qd =                    xd =
   -1.7508   -0.9093                 -0.0698                   0.0829
    0.0000   -0.0000                  0.0431                  -0.0000
    0.1242   -0.4161                                          -0.0266
         0         0                                                0
   -1.0000   -1.0000                                           0.0266
    0.0000    0.0000                                          -0.0000
```

Velocity Relationships : Jacobians for Under Actuated Robots : 2-Link Example MATLAB PCT

What about if only 2 DOF are specified to be controlled?

Deriving Jacobian for v_x and v_y. Finding $\dot{q}_1$ and $\dot{q}_2$.

$$\begin{bmatrix} v_x \\ v_y \\ \hline v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} J_{xy} \\ J_0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

Take the top matrix

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = J_x y \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

$$J_{xy}^{-1} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = J_{xy}^{-1} J_x y \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

$$J_{xy}^{-1} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = I_{2x2} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

$$J_{xy}^{-1} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = J_{xy}^{-1} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

Velocity Relationships : Jacobians for Under Actuated Robots : 2-Link Example MATLAB PCT

Jacobians for Under Actuated Robots : 2-Link Example MATLAB PCT

```
Jxy = J(1:2,:)
qd = inv(Jxy)*[0.1 0]'
xd = J*qd
```

```
xd=
    0.1000
    0.0000
         0
         0
         0
   -0.0642
```

Observe correct motion for v_x and v_y. Rotational velocity about Z is unavoidable.

Velocity Relationships : Jacobians for Redundantly Actuated Robots : 8-Link Example MATLAB PCT

For a redundantly-actuated robot, the Jacobian is wider than it is tall.

There are infinite solutions, so MATLAB returns the one for which the norm is minimum $|\dot{q}|$.

$$\dot{q} = J(q)^{+}\nu$$

Where $J(q)^{+}$ is the left pseudo-inverse.

Consider an 8 axis p8 robot at a nominal pose

```
qn8 = [0 0 qn]
```

Velocity Relationships : Jacobians for Redundantly Actuated Robots : 8-Link Example MATLAB PCT

Jacobians for Redundantly Actuated Robots : 8-Link Example MATLAB PCT

```
mdl_p8
p8.base = eye(4)
qn8 = [0 0 qn]
J = jacob0(p8,qn8)
xd = [0.2 0.2 0.2 0 0 0]
q = pinv(J)*xd'
```

```
q =
      0.1859
      0.1799
      0.0338
     -0.3569
      0.0441
      0.0478
      0.3128
     -0.0338
```

All 8 joints have non-zero velocity (q)

The Jacobian has 8 columns and 6 rows

Velocity Relationships : Jacobians for Redundantly Actuated Robots : 8-Link Example MATLAB PCT

Because the robot is redundantly actuated, some joint angles can be used to achieve additional constraints (eg. obstacle avoidance, or moving away from joint limits.)

Null-space motion. Two orthogonal vectors, where any linear combination of them results in no displacement or change in orientation in cartesian space.

```
N = null(J)
n =
    0.0849    0.2419
    0.2843   -0.0205
   -0.4768    0.0343
    0.0209    0.3855
   -0.0428   -0.7900
   -0.6742    0.0486
    0.0219    0.4045
    0.4768   -0.0343
```

Velocity Relationships : Jacobians for Redundantly Actuated Robots : 8-Link Example MATLAB PCT

Null-space motion can be used to avoid collisions between links and obstacles.

Because null-space motion does not affect the final pose in terms of end-effector pose or orientation, these joint angle components can be added to the general solution method.

That is:

$$\dot{q} = J(q)^{+}\nu + NN^{+}\dot{q}_{ns}$$

where $NN^{+}\dot{q}_{ns}$ is the null space motion

Velocity Relationships : Jacobians for Redundantly Actuated Robots : 8-Link Example MATLAB PCT

Jacobians for Redundantly Actuated Robots : 8-Link Example MATLAB PCT : Null Space

```
N = null(J)
qd_ns = [0 0 0 0 -0.1 0 0 0]'
qp = N*pinv(N)*qd_ns
qp = qp/qp(5)*qd_ns(5)
the_norm = norm(J*qp)
```

```
the_norm =

    3.2718e-17
```

This is a very small number, which confirms that the end effector did not move much at all.

Velocity Relationships : Jacobian Singularity

For the case where the Jacobian is square, but singular (eg. where det(J) = 0), the equation $\dot{q} = J(q)^{-1}\nu$ cannot be solved.

However, if a small value p is added to all diagonal elements, the inverse can be found with some small error in $\dot{q}$

$$\dot{q} = (J(q) + pI)^{-1}\nu$$

where

$$pI = \begin{bmatrix} p & 0 & 0 & 0 & 0 & 0 \\ 0 & p & 0 & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 \\ 0 & 0 & 0 & p & 0 & 0 \\ 0 & 0 & 0 & 0 & p & 0 \\ 0 & 0 & 0 & 0 & 0 & p \end{bmatrix}$$

Adding *pI* immediately makes each column unique, and the determinant can be found.

Force Relationships : Introduction and Relationship to Velocity Technique

Forces and wrenches are going to be discussed further in the next lecture series. This is a preliminary introduction as the techniques are based on very similar matrix principles as for linear and rotational velocities.

Equivalent concept between:

$$\nu = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \qquad g = \begin{bmatrix} f_x \\ f_y \\ f_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$

g is a matrix of force and moment components in X, Y and Z axes.

Force Relationships : Transforming Wrenches Between Frames

What is wrench?

It is a coupling of both force and moment transfer, acting at a joint. For a 6 DOF robot, wrench can be calculated in each of the 6 joint frames, as well as transformed between them.

Two wrenches attached to the same rigid body can be transformed in the following manner:

$$^{A}g = (^{B}J_{A})^{T} \;\; ^{B}g$$

Notice, this time, that the Jacobian has been transposed, and the mapping is from frame {B} to frame {A}.

Force Relationships : Transforming Wrenches Between Frames : MATLAB PCT

Finding the ${}^{B}J_{A}$ matrix (From Earlier)

Frame {A} and {B} attached to same body

$$ {}^{B}J_{A} = J_{v}({}^{A}T_{B}) = \begin{bmatrix} {}^{B}R_{A} & 0_{3x3} \\ 0_{3x3} & {}^{B}R_{A} \end{bmatrix} $$

Frame {A} and {B} attached to different bodies

$$ {}^{B}J_{A} = \bar{J}_{v}({}^{A}T_{B}) = \begin{bmatrix} {}^{B}R_{A} & -{}^{B}R_{A}S({}^{A}t_{B}) \\ 0_{3x3} & {}^{B}R_{A} \end{bmatrix} $$

For a robot, co-ordinate systems are on different bodies - use the second technique

Force Relationships : Transforming Wrenches Between Frames : MATLAB PCT

Suppose a frame that is displaced from another by a translation in x of 2 metres with a force applied of 3N in the y direction

Transforming Wrenches Between Frames : MATLAB PCT

```
J = tr2jac(transl(2,0,0))
F = J'*[0 3 0 0 0 0]'
```

```
F =
       0
       3
       0
       0
       0
       6
```

Second row, 3N in the y direction

Sixth rox, 6Nm in the z axis

Force Relationships : Transforming Wrenches Into Joint Space : MATLAB PCT

Similar to how the manipulator Jacobian transforms joint velocity to the end-effector spatial velocity, the Jacobian transpose transforms a wrench applied at the end-effector to torques and forces experienced by the joints.

$Q =^0 J(q)^+ \; {}^0g$ expressed in the base space

$Q =^N J(q)^+ \; {}^Ng$ expressed in the end-effector space

The great thing about this Jacobian is that it can never be singular!

Force Relationships : Transforming Wrenches Into Joint Space : MATLAB PCT

Transforming Wrenches Between Frames : MATLAB PCT

```
tau = p560.jacob0(qn)'*[0 20 0 0 0 0]'
```

```
tau =
   11.9261
    0.0000
    0.0000
         0
         0

         0
```

The first joint needs to restrain 11.9261Nm of torque for a 20N external load

This vector contains magnitudes of torque and force for revolute and prismatic joints respectively

Force Relationships : Summary

Wrenches consist of forces and moments

Wrenches can be transformed between frames in much the same way as linear and rotational velocities by way of the Jacobian transpose.

The matrix technique of $Q =^0 J(q)^+ \ {}^0g$ can be used to find the torque vector for robot consisting of revolute joints. For prismatic joints, the vector contains force requirements of the joint.