# Matlab Chapter 8
# Velocity Relationships

**Shuhao Liang**
Graduate Institute of Intelligent Manufacturing Technology
National Taiwan University of Science and Technology
2024

# Review – Chapter 3 Time and Motion

## 3.1 Time-Varying Pose

In this section we discuss how to describe the rate of change of pose which has both a translational and rotational velocity component. The translational velocity is straightforward: it is the rate of change of the position of the origin of the coordinate frame. Rotational velocity is a little more complex.

### 3.1.1 Derivative of Pose

There are many ways to represent the orientation of a coordinate frame but most convenient for present purposes is the exponential form

$$ {}^{A}\boldsymbol{R}_{B}(t) = \mathrm{e}^{\left[{}^{A}\hat{\omega}(t)\right]_{\times}\theta(t)} \in \mathbf{SO}(3) $$

where the rotation is described by a rotational axis ${}^{A}\hat{\omega}(t)$ defined with respect to frame $\{A\}$ and a rotational angle $\theta(t)$, and where $[\cdot]_{\times}$ is a skew-symmetric matrix.

At an instant in time $t$ we will assume that the axis has a fixed direction and the frame is rotating around the axis. The derivative with respect to time is

Rotation

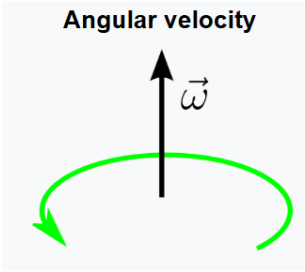
$$ {}^{A}\dot{\boldsymbol{R}}_{B}(t) = \left[{}^{A}\hat{\omega}(t)\right]_{\times}\dot{\theta}\,\mathrm{e}^{\left[{}^{A}\hat{\omega}(t)\right]_{\times}\theta(t)} \in \mathbb{R}^{3\times 3} $$

$$ = \left[{}^{A}\hat{\omega}(t)\right]_{\times}\dot{\theta}\,{}^{A}\boldsymbol{R}_{B}(t) $$

**Velocity**

A derivative represents the rate of change of something. In the case of position, the first derivative therefore would be the **rate of change of position**, otherwise known as velocity.

${}^{A}\hat{\omega}(t)$ : angular velocity




$\theta(t)$ : Rotational angle

$\left[{}^{A}\hat{\omega}(t)\right]_{\times}$ :Skew-symmetric matrix

# skew-symmetric matrix

https://en.wikipedia.org/wiki/Skew-symmetric_matrix

- a square matrix whose transpose equals its negative.

$$A \text{ skew-symmetric} \iff A^{\mathsf{T}} = -A.$$

- if $a_{ij}$ denotes the entry in the i-th row and j-th column, then the skew-symmetric condition is equivalent to

$$A \text{ skew-symmetric} \iff a_{ji} = -a_{ij}.$$

## Example [ edit ]

The matrix

$$A = \begin{bmatrix} 0 & 2 & -45 \\ -2 & 0 & -4 \\ 45 & 4 & 0 \end{bmatrix}$$

is skew-symmetric because

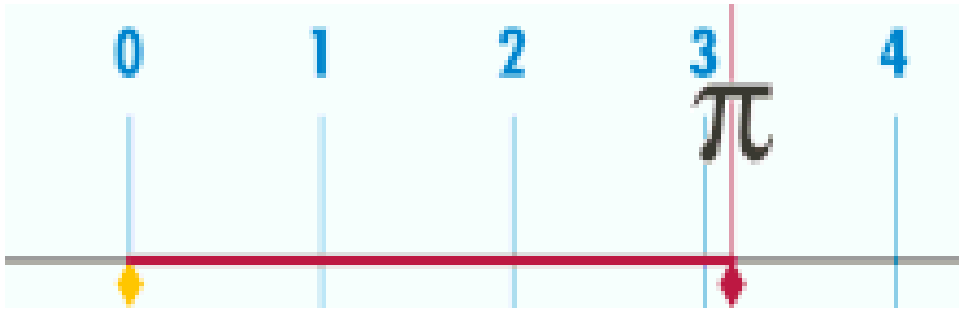
$$-A = \begin{bmatrix} 0 & -2 & 45 \\ 2 & 0 & 4 \\ -45 & -4 & 0 \end{bmatrix} = A^{\mathsf{T}}.$$

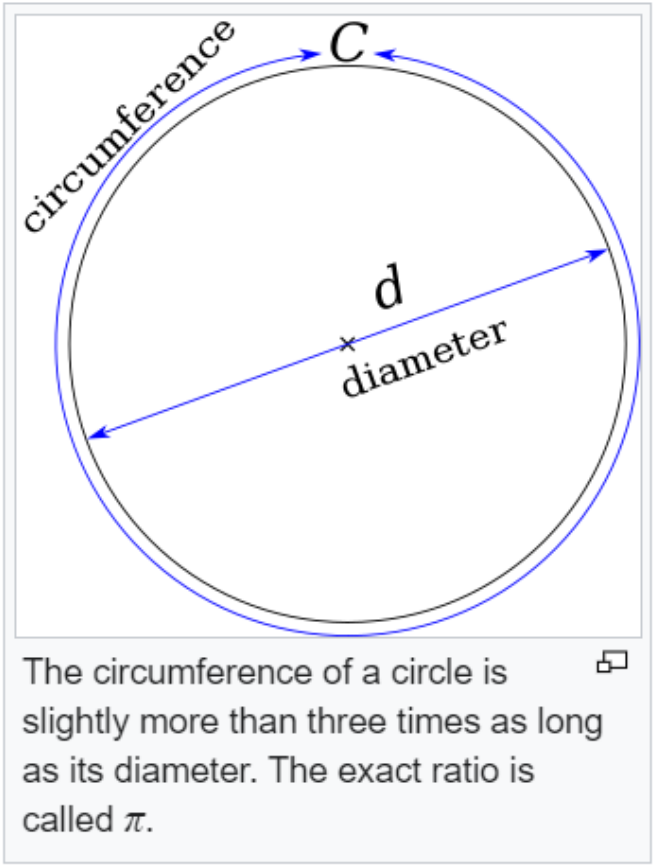
# Pi - π

https://en.wikipedia.org/wiki/Pi

- **π= 3.14159**
- **Definition**
  - π is commonly defined as the ratio of a circle‘s circumference 圓周長 **C** to its diameter d

$$\pi = \frac{C}{d}$$




The circumference of a circle is slightly more than three times as long as its diameter. The exact ratio is called $\pi$.

# Radian (rad)

https://en.wikipedia.org/wiki/Radian

- The arc of a complete circle is 2 π, so 180° equal to π 。

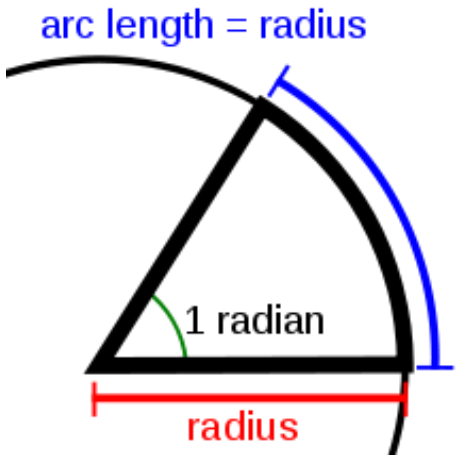
以度數表示的角，把數字乘以$\frac{\pi}{180^\circ}$便轉換成弧度；以弧度表示的角，乘以$\frac{180^\circ}{\pi}$便轉換成度數。

$$\pi = 180^\circ$$

$$1 = \frac{180^\circ}{\pi} \approx 57.2958^\circ$$

$$\frac{\pi}{3} = \frac{\pi}{3} \cdot \frac{180^\circ}{\pi} = 60^\circ$$

$$1^\circ = 1^\circ \cdot \frac{\pi}{180^\circ} \approx 0.0175$$



# Angular velocity

https://en.wikipedia.org/wiki/Angular_velocity

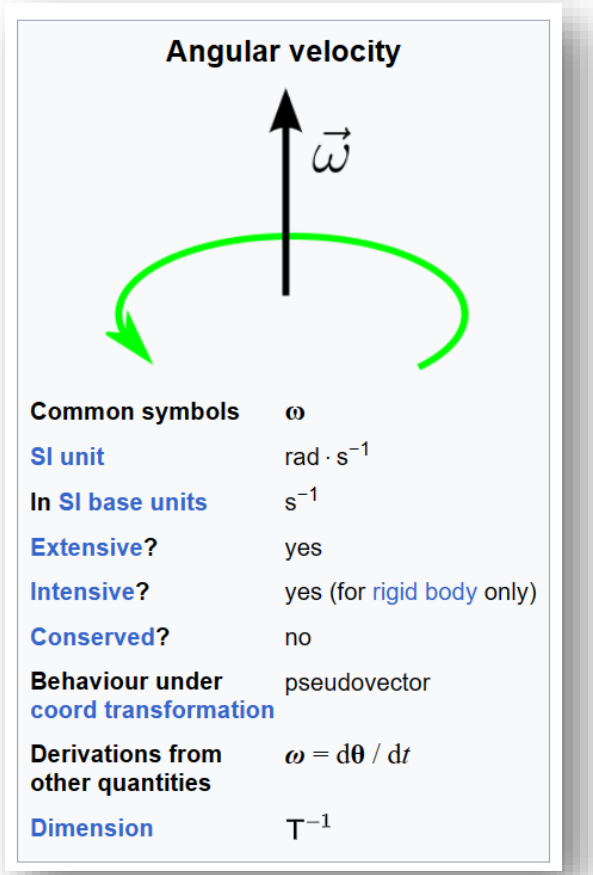
https://en.wikipedia.org/wiki/Angular_frequency

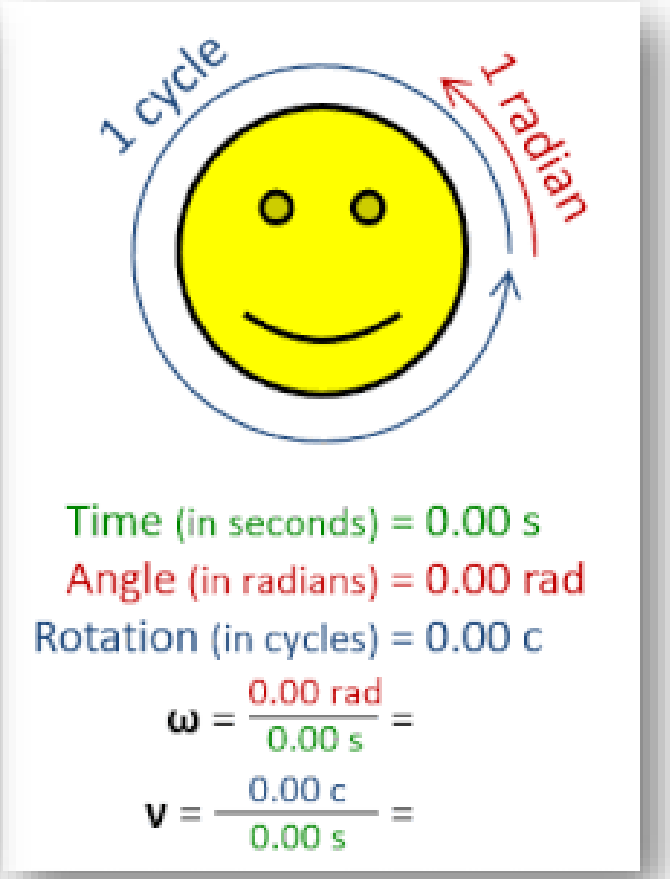
- **Angular frequency** can be obtained multiplying *rotational frequency, ν* (or ordinary frequency, f) by a *full turn (2π radians):*

- $\omega = 2\pi \text{ rad} \cdot \nu$.

It can also be formulated as **$\omega = d\theta/dt$**, the instantaneous rate of change of the angular displacement, θ, with respect to time, t.

| | |
|---|---|
| 轉速 rpm | ν (or ordinary frequency, f) |



# Reference

ETH zürich

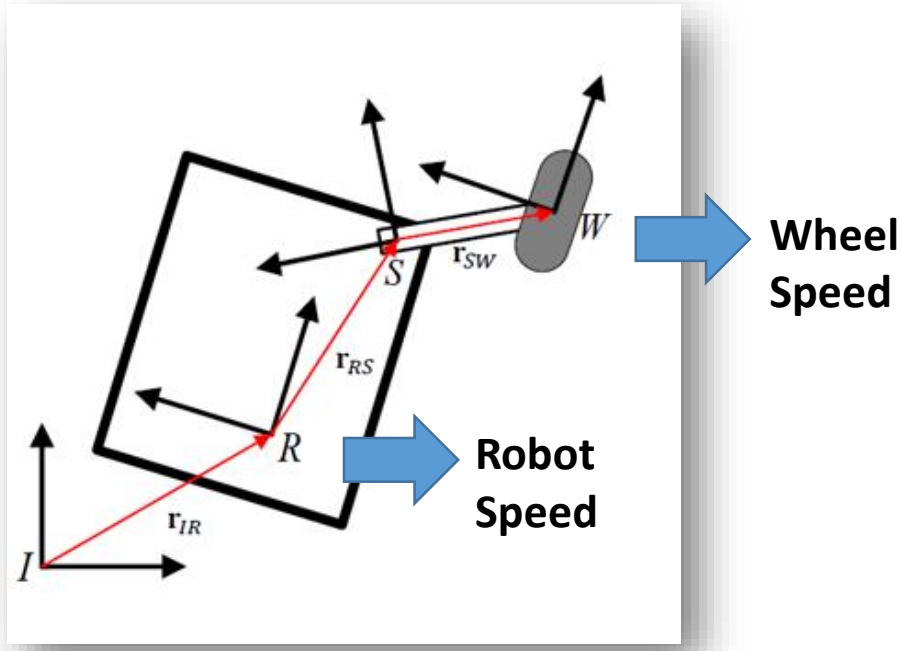
ASL
Autonomous Systems Lab

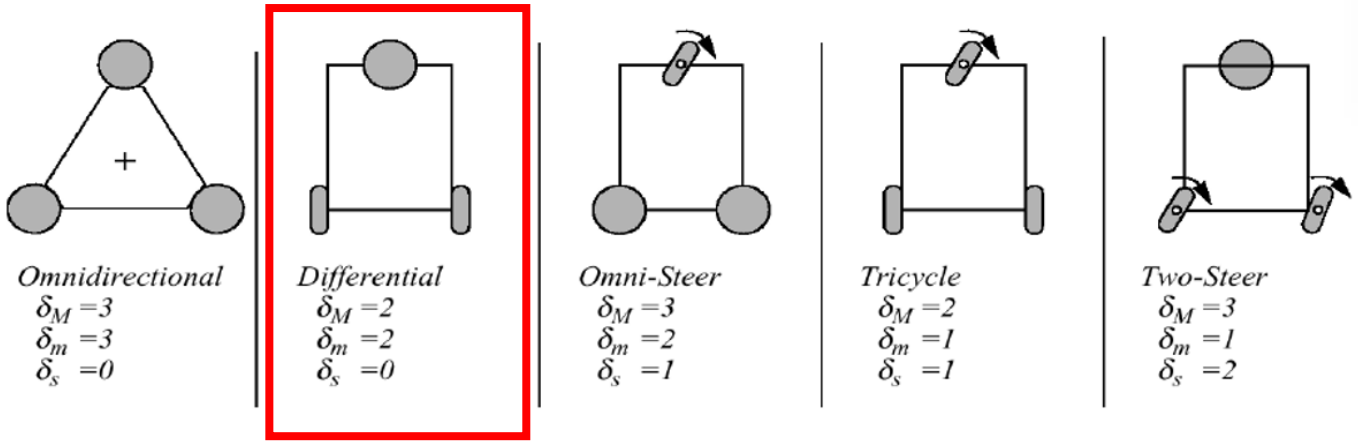
## Wheeled Locomotion | Introduction
Autonomous Mobile Robots

**Paul Furgale**
Margarita Chli, Marco Hutter, Martin Rufli, Davide Scaramuzza, Roland Siegwart

# Introduction to Wheeled Locomotion:

- ✓ Introduction
- ✓ Differential Kinematics
- ✓ Wheeled Kinematics
- ✓ Worked Exercise | A Differential Drive Robot



# Differential Drive

## A Differential Drive Robot | Summary

機動性

- Degree of Maneuverability

$$\delta_m = 2, \qquad \delta_s = 0, \qquad \delta_M = 2$$

- Forward differential kinematics

Robot speed, angle

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ r/2b & -r/2b \end{bmatrix} \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix}$$

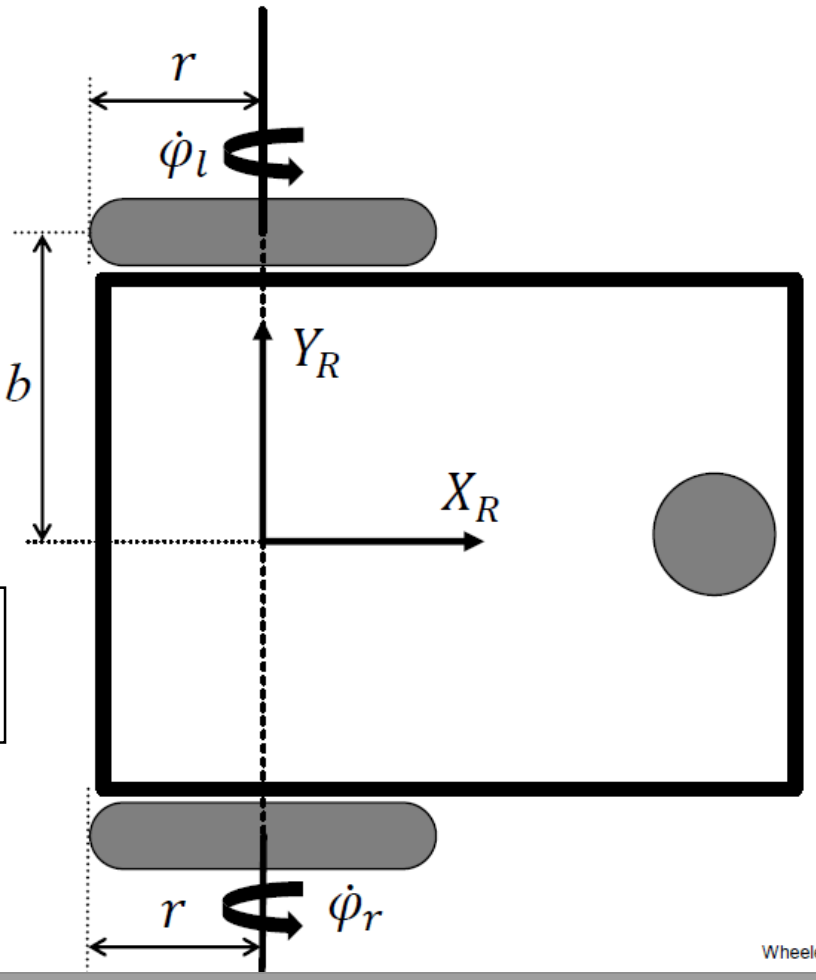
已知右輪、左輪轉速

- Inverse differential kinematics

Wheel speed, angle

$$\begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} = \begin{bmatrix} 1/r & 0 & b/r \\ 1/r & 0 & -b/r \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

預計車子行走的速度x,y及轉向角 θ



# Part III Arm-Type Robots

Chapter 7 Robot Arm Kinematics

Chapter 8 Velocity Relationships

Chapter 9 Dynamics and Control

Chapter

# 8 Manipulator Velocity

8.1 Manipulator **Jacobian**

8.2 **Jacobian** Condition and Manipulability

8.3 Resolved-Rate Motion Control

8.4 **Under- and Over-Actuated** Manipulators

8.5 Force Relationships

8.6 Inverse Kinematics: a General Numerical Approach

8.7 Advanced Topics

8.8 Wrapping Up

# Jacobians

- Jacobians are an important concept in robotics, relating changes in **one space to changes in another**.

- Reference
  - Derive the Jacobian of u and v with respect to x and y
  - What is Jacobian?
  - Jacobian and number of robot joints

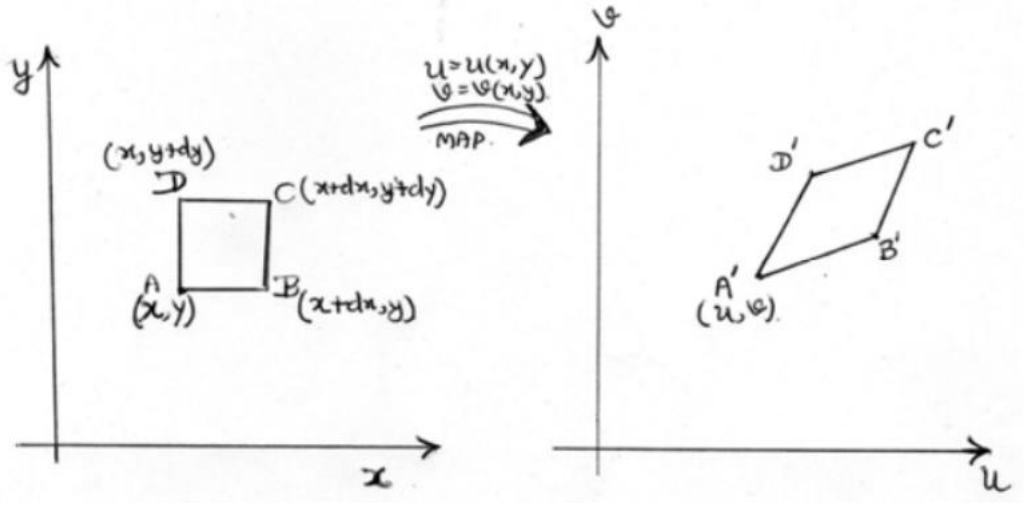
# Derive the Jacobian of u and v with respect to x and y

https://i.sstatic.net/UU2dV.png

Consider a small differential rectangular element ABCD in the x-y coordinate system as shown below. This shape is mapped to quadrilateral A′ B′ C′ D′ under the mapping $u = u(x,y)$ and $v = v(x,y)$. Show that under this mapping

$$\frac{Area\ of\ A'B'C'D'}{Area\ of\ ABCD} = \begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}$$

The ratio is called as the Jacobian of $u$ and $v$ with respect to $x$ and $y$ and denoted by $\frac{\partial(u,v)}{\partial(x,y)}$.



# What is Jacobian?

https://www.goseeko.com/blog/what-is-jacobian/

## Overview(Jacobian)

Jacobian is defined as- If u and v are functions of the two independent variables x and y , then the determinant,

$$\begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}$$

is known as the jacobian of u and v with respect to x and y.

We can write it as below

$$\frac{\partial(u,v)}{\partial(x,y)} \quad or\ simply\ J(u,v)$$

Suppose there are three functions u, v and w of three independent variables x , y and z then,

Suppose there are three functions u , v and w of three independent variables x , y and z then,

The Jacobian can be defined as,

$$\frac{\partial(u,v,w)}{\partial(x,y,z)} = \begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{vmatrix}$$

# Important properties of the Jacobians

- **Property-1**
  - **If u and v are the functions of x and y , then**

$$\frac{\partial(u,v)}{\partial(x,y)}\cdot\frac{\partial(x,y)}{\partial(u,v)}=1$$

- **Property-2:**
  - **Second property is also known as chain rule.**

$$\frac{\partial(u,v)}{\partial(x,y)}=\frac{\partial(u,v)}{\partial(r,s)}\cdot\frac{\partial(r,s)}{\partial(x,y)}$$

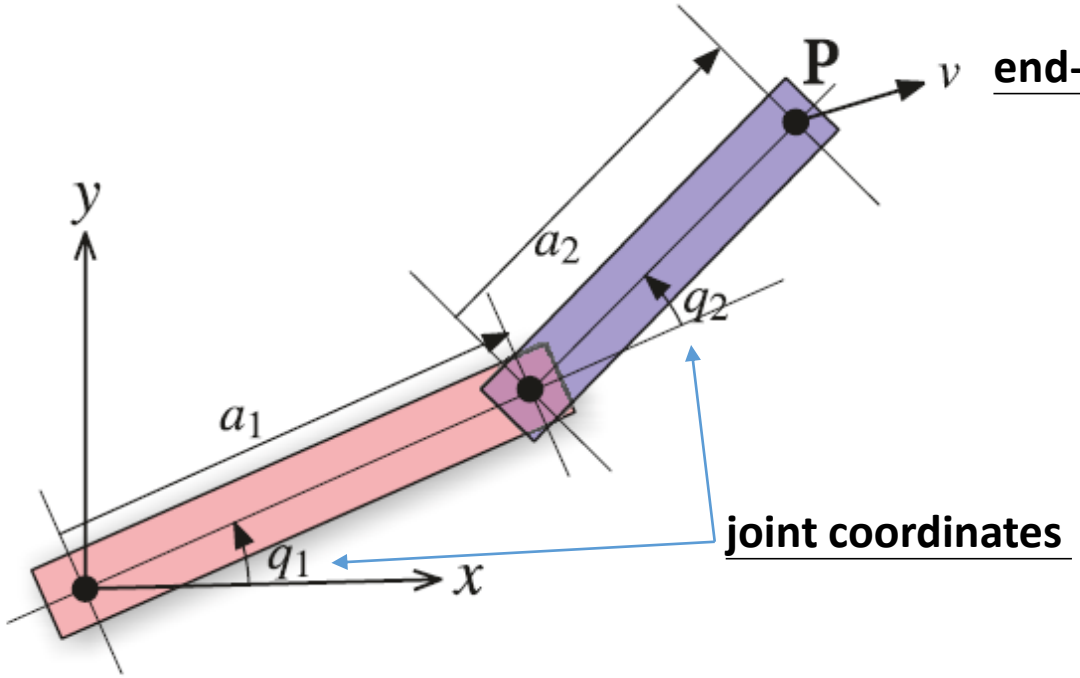
$$\frac{\partial(u,v)}{\partial(r,s)}\cdot\frac{\partial(r,s)}{\partial(x,y)}=\begin{vmatrix}\frac{\partial u}{\partial r} & \frac{\partial u}{\partial s}\\ \frac{\partial v}{\partial r} & \frac{\partial v}{\partial s}\end{vmatrix}\begin{vmatrix}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\\ \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y}\end{vmatrix}$$

- **Property-3**
  - If u,v,w are the functions of three independent variables x,y,z are not independent , then,

$$\frac{\partial(u,v,w)}{\partial(x,y,z)}=0$$

# 8.1 Manipulator Jacobian

- The **manipulator Jacobian** which describes the relationship between the rate of change of **joint coordinates** and the spatial velocity of the **end-effector** expressed in either the world frame or the end-effector frame.




**end-effector**

$$\begin{cases} p = (x, y) \\ \nu = \mathrm{d}p \,/\, \mathrm{d}t \end{cases}$$

**joint coordinates**

Fig. 8.1.
Two-link robot showing the end-effector position $p = (x, y)$ and the Cartesian velocity vector $\nu = \mathrm{d}p \,/\, \mathrm{d}t$

and define two real-valued symbolic variables to represent the joint angles

```
>> syms q1 q2 real
```

and then compute the forward kinematics using these

```
>> TE = p2.fkine( [q1 q2] );
```

The position of the end-effector $\boldsymbol{p} = (x, y) \in \mathbb{R}^2$ is

```
>> p = TE.t;  p = p(1:2)
p =
 a2*cos(q1 + q2) + a1*cos(q1)
 a2*sin(q1 + q2) + a1*sin(q1)
```

The Toolbox considers robot pose in 3-dimensions using SE(3). This robot operates in a plane, a subset of SE(3), so we select $p = (x, y)$.

and we compute the derivative of $\boldsymbol{p}$ with respect to the joints variables $\boldsymbol{q}$. Since $\boldsymbol{p}$ and $\boldsymbol{q}$ are both vectors the derivative

$$\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q}) \tag{8.1}$$

will be a matrix – a Jacobian matrix

```
>> J = jacobian(p, [q1 q2])
J =
[ - a2*sin(q1 + q2) - a1*sin(q1), -a2*sin(q1 + q2)]
[   a2*cos(q1 + q2) + a1*cos(q1),  a2*cos(q1 + q2)]
```

Physical insight comes from Fig. 8.2 which shows the joint axes in space. Alternatively you could use the `teach` method

```
>> p560.teach(qn)
```

and jog the various joint angles and observe the change in end-effector pose.

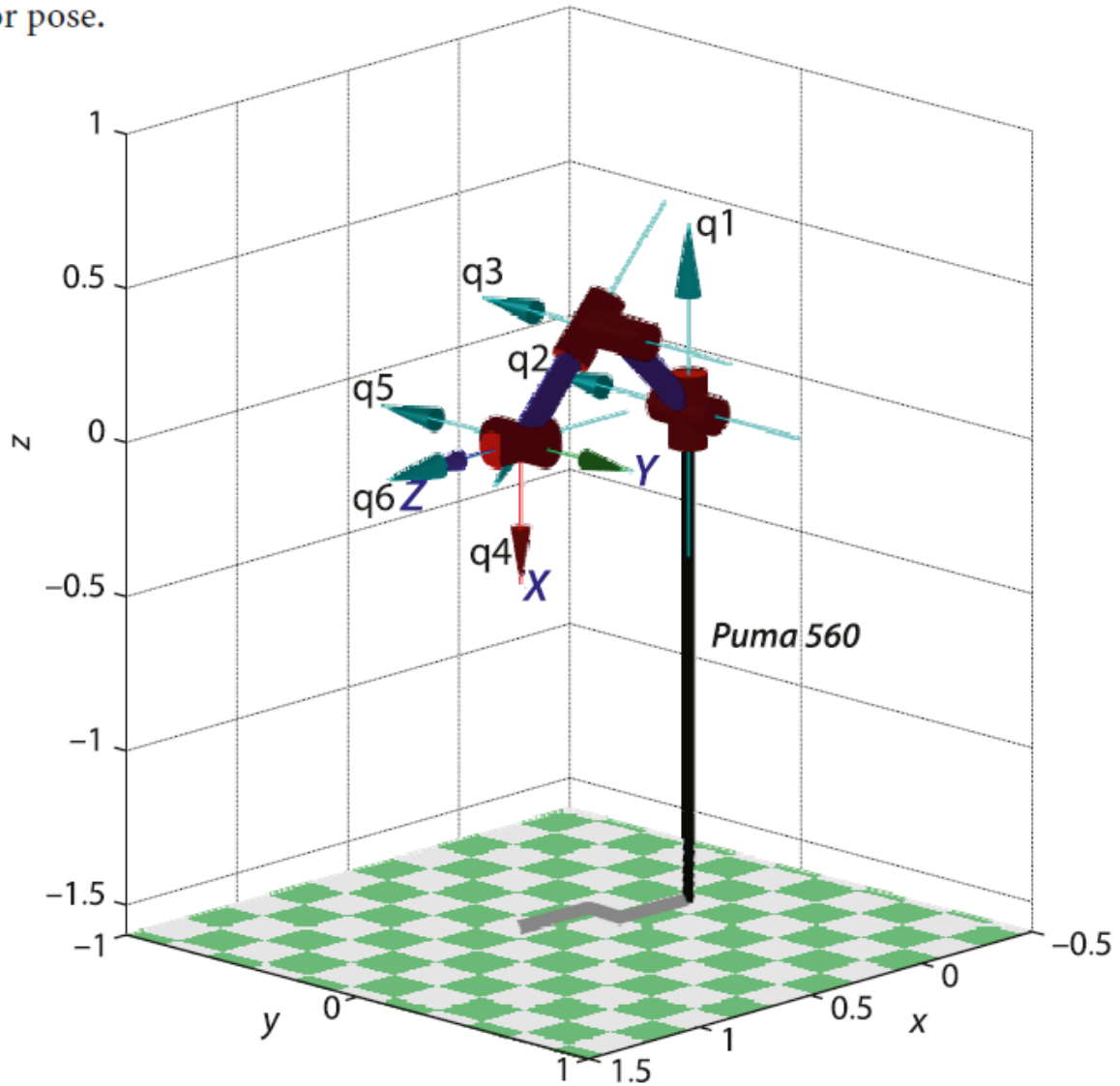



**Fig. 8.2.**
Puma robot in its nominal pose `qn`. The end-effector $z$-axis points in the world $x$-direction, and the $x$-axis points downward

# 8.2 Jacobian Condition and Manipulability

## 8.2 Jacobian Condition and Manipulability

We have discussed how the Jacobian matrix maps joint rates to end-effector Cartesian velocity but the inverse problem has strong practical use – what joint velocities are needed to achieve a required end-effector Cartesian velocity? We can invert Eq. 8.2 and write

$$\dot{q} = J(q)^{-1}\nu \tag{8.3}$$

provided that $J$ is square and nonsingular. The Jacobian is a dim $\mathcal{T} \times$ dim $\mathcal{C}$ matrix so in order to achieve a square Jacobian matrix a robot operating in the task space $\mathcal{T} \subset \mathbf{SE}(3)$, which has 6 spatial degrees-of-freedom, requires a robot with 6 joints.

# Inverse Jacobian

- The inverse Jacobian can be used to resolve desired **Cartesian velocity** into **joint velocity** as an alternative means of generating Cartesian paths for under- and over-actuated robots.

### 8.2.1 Jacobian Singularities

A robot configuration $q$ at which $\det(J(q)) = 0$ is described as singular or degenerate. Singularities occur when the robot is at maximum reach or when one or more axes become aligned resulting in the loss of degrees of freedom – the gimbal lock problem again.

For example at the Puma's *ready* pose the Jacobian

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

is singular

```
>> det(J)
ans =
     0
```

Digging a little deeper we see that the Jacobian rank is only

```
>> rank(J)
ans =
     5
```

### 8.2.2 Manipulability

Consider the set of generalized joint velocities with a unit norm

$$\dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}} = 1$$

which lie on the surface of a hypersphere in the $N$-dimensional joint velocity space. Substituting Eq. 8.3 we can write

$$\boldsymbol{\nu}^T \left( \boldsymbol{J}(\boldsymbol{q}) \boldsymbol{J}(\boldsymbol{q})^T \right)^{-1} \boldsymbol{\nu} = 1 \tag{8.4}$$

which is the equation of points on the surface of an ellipsoid within the dim $\mathcal{T}$-dimensional end-effector velocity space. If this ellipsoid is close to spherical, that is, its radii are of the same order of magnitude then all is well – the end-effector can achieve arbitrary Cartesian velocity. However if one or more radii are very small this indicates that the end-effector cannot achieve velocity in the directions corresponding to those small radii.

We will load the numerical, rather than symbolic model, for the planar robot arm of Fig. 8.1

```
>> mdl_planar2
```

which allows us to plot the velocity ellipse for an arbitrary pose

```
>> p2.vellipse([30 40], 'deg')
```

We can also interactively explore how its shape changes with configuration by

```
>> p2.teach('callback', @(r,q) r.vellipse(q), 'view', 'top')
```

which is shown in Fig. 8.3.

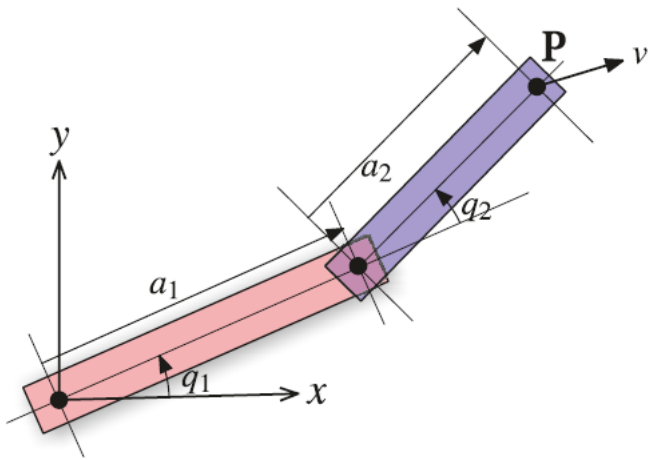



Fig. 8.1.
Two-link robot showing the end-effector position $p = (x, y)$ and the Cartesian velocity vector $\nu = \mathrm{d}p / \mathrm{d}t$

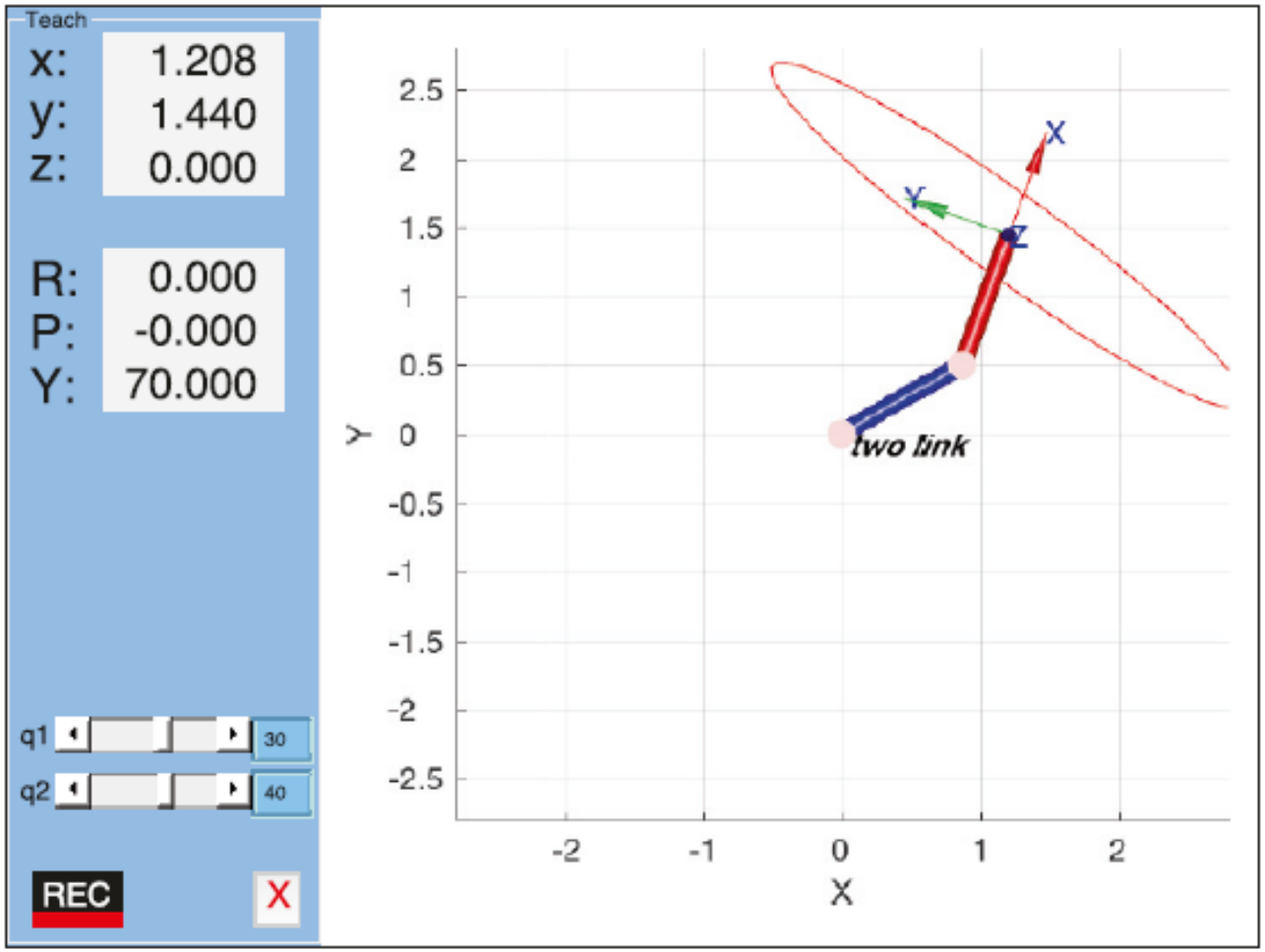



Fig. 8.3.
Two-link robot with overlaid velocity ellipse

# 8.3 Resolved-Rate Motion Control

## 8.3 Resolved-Rate Motion Control

Resolved-rate motion control is a simple and elegant algorithm to generate straight line motion by exploiting Eq. 8.3

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

to map or *resolve* desired Cartesian velocity to joint velocity without explicitly requiring inverse kinematics as we used earlier. For now we will assume that the Jacobian is square ($6 \times 6$) and nonsingular but we will relax these constraints later.

The motion control scheme is typically implemented in discrete-time form as

$$\dot{q}^*\langle k\rangle = J(q\langle k\rangle)^{-1}\nu^* \tag{8.5}$$

$$q^*\langle k+1\rangle \leftarrow q\langle k\rangle + \delta_t \dot{q}^*\langle k\rangle$$

where $\delta_t$ is the sample interval. The first equation computes the required joint velocity as a function of the current joint configuration and the desired end-effector velocity $\nu^*$. The second performs forward rectangular integration to give the desired joint angles for the next time step, $q^*\langle k+1\rangle$.

# An example of the algorithm is implemented by the Simulink® model

```
>> sl_rrmc
```

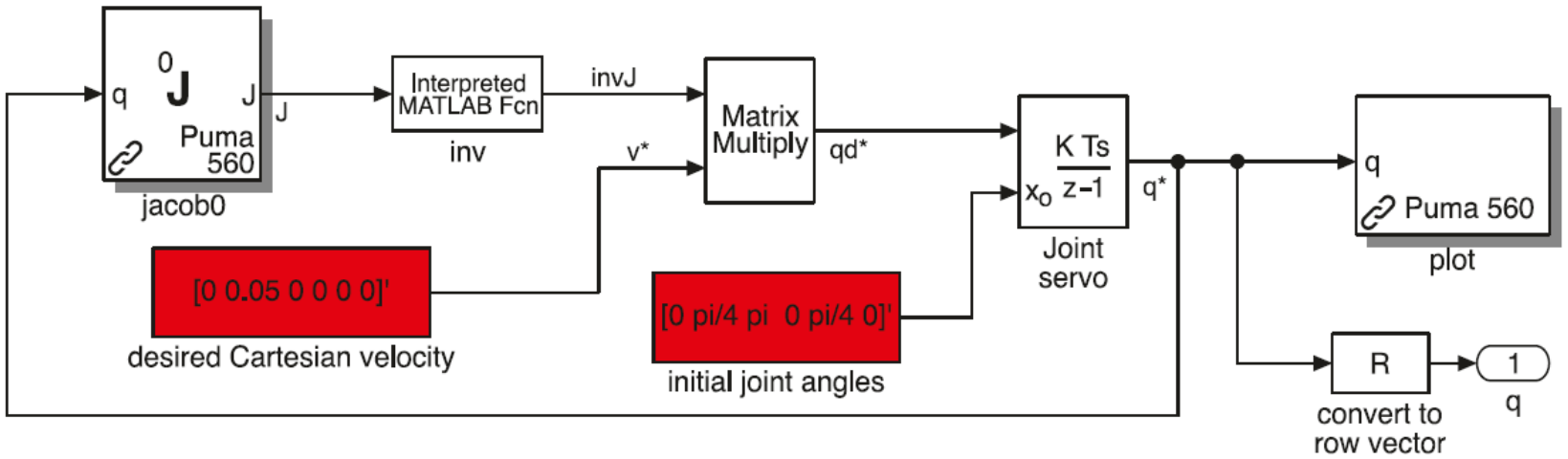



Fig. 8.5. The Simulink® model `sl_rrmc` for resolved-rate motion control for constant end-effector velocity

Running the simulation

```
>> r = sim('sl_rrmc');
```

we see an animation of the manipulator end-effector moving at constant velocity in Cartesian space. Simulation results are returned in the simulation object `r` from which we extract time and joint coordinates

```
>> t = r.find('tout');
>> q = r.find('yout');
```

We apply forward kinematics to determine the end-effector position

```
>> T = p560.fkine(q);
>> xyz = transl(T);
```

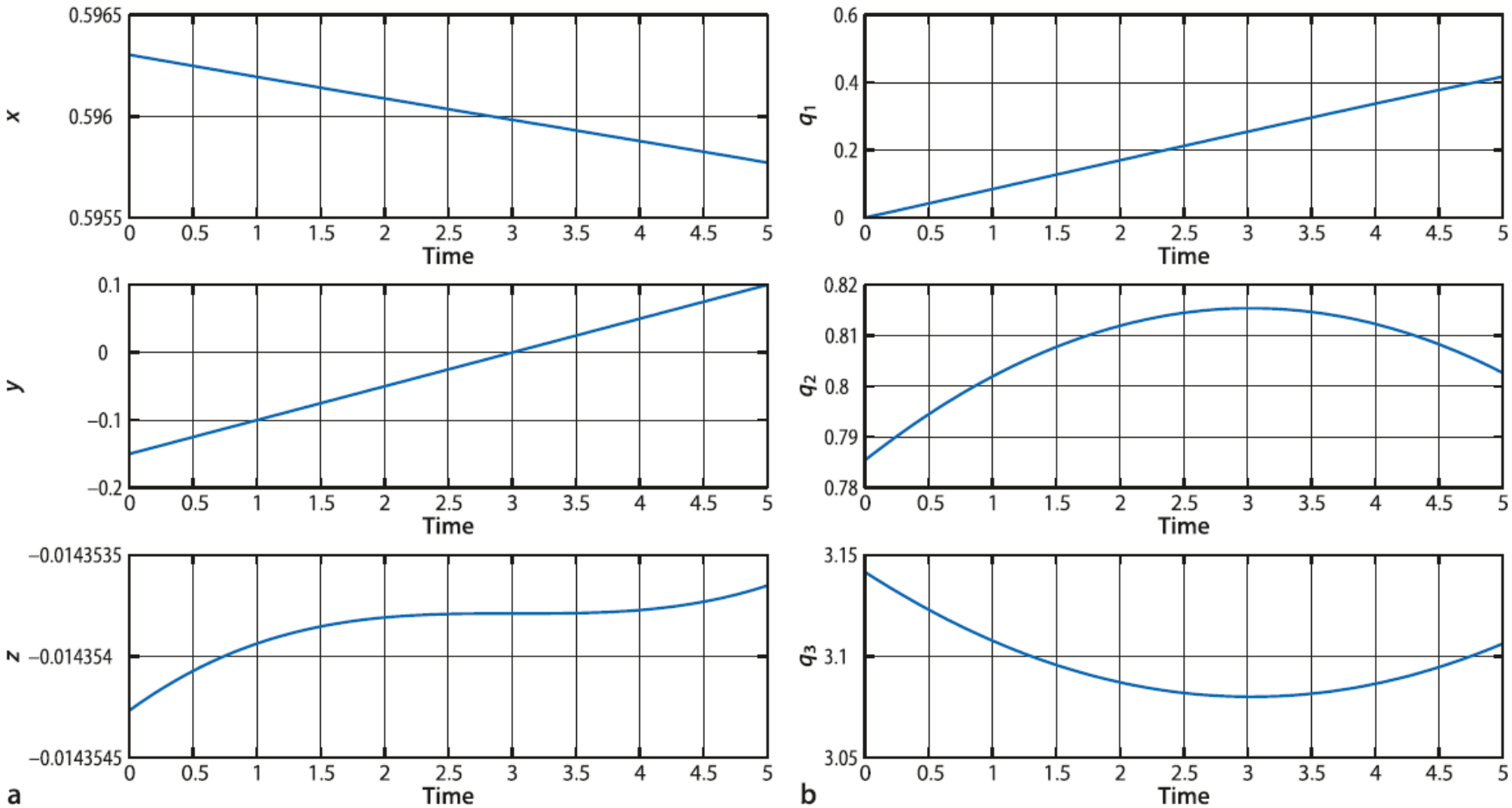



Fig. 8.6. Resolved-rate motion control, Cartesian and joint coordinates versus time. a Cartesian end-effector motion. Note the small, but unwanted motion in the $x$- and $z$-directions; b joint motion

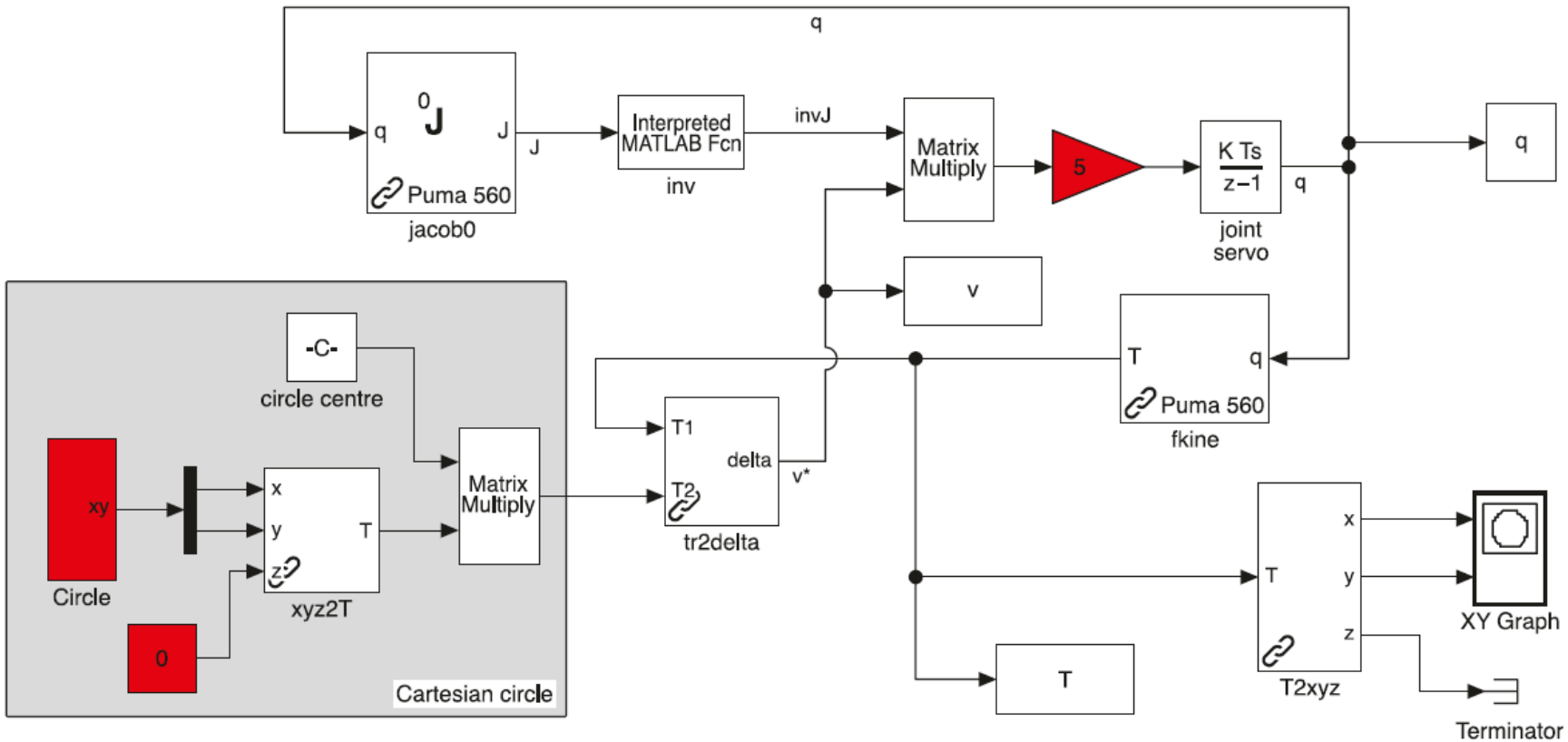



Fig. 8.7. The Simulink® model `sl_rrmc2` for closed-loop resolved-rate motion control with circular end-effector motion

# 8.4 Under- and Over-Actuated Manipulators

## 8.4 Under- and Over-Actuated Manipulators

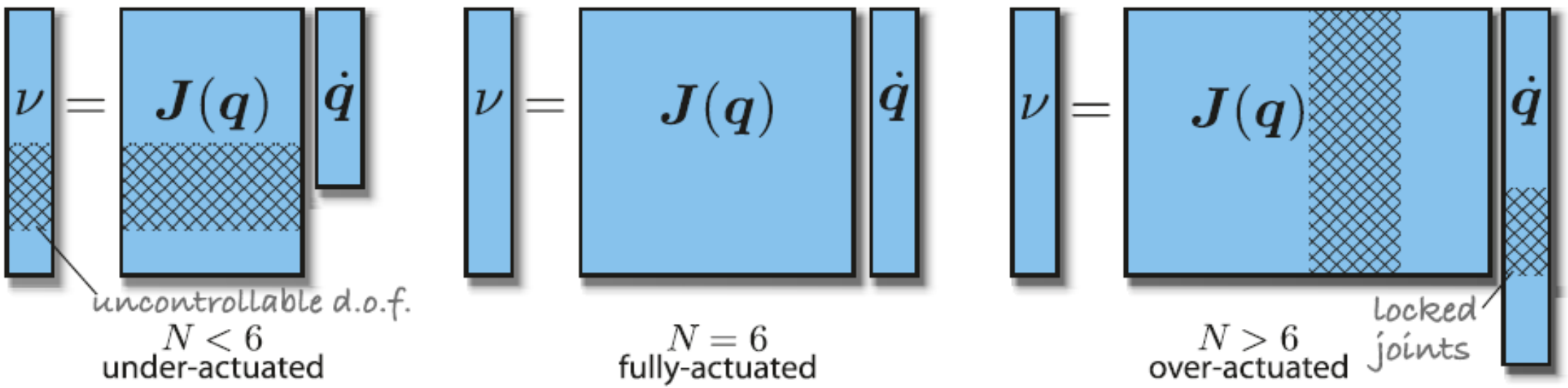
So far we have assumed that the Jacobian is square. For the nonsquare cases it is helpful to consider the velocity relationship

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

in the diagrammatic form shown in Fig. 8.8. The Jacobian is a $6 \times N$ matrix, the joint velocity is an $N$-vector, and $\boldsymbol{\nu}$ is a 6-vector.

The case of $N < 6$ is referred to as an under-actuated robot, and $N > 6$ is over-actuated or redundant. The under-actuated case cannot be solved because the system of equations is under-constrained but the system can be *squared up* by deleting some rows of $\boldsymbol{\nu}$ and $\boldsymbol{J}$ – accepting that some Cartesian degrees of freedom are not controllable given the low number of joints. For the over-actuated case the system of equations is under-constrained and the best we can do is find a least-squares solution as described in the previous section. Alternatively we can *square up* the Jacobian to make it invertible by deleting some columns – effectively *locking* the corresponding joints.

**Fig. 8.8.**
Schematic of Jacobian, $\boldsymbol{\nu}$ and $\dot{\boldsymbol{q}}$ for different cases of $N$. The *hatched* areas represent matrix regions that could be deleted in order to create a square sub-system capable of solution



### 8.4.2 Jacobian for Over-Actuated Robot

An over-actuated or redundant robot has $N > 6$, and a Jacobian that is wider than it is tall. In this case we rewrite Eq. 8.3 to use the left pseudo-inverse

$$\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})^{+} \boldsymbol{\nu} \tag{8.7}$$

which, of the infinite number of solutions possible, will yield the one for which $\|\dot{\boldsymbol{q}}\|$ is smallest – the minimum-norm solution.

We will demonstrate this for the left arm of the Baxter robot from Sect. 7.2.2.4 at a nominal pose

```
>> mdl_baxter
>> TE = SE3(0.8, 0.2, -0.2) * SE3.Ry(pi);
>> q = left.ikine(TE)
```

and its Jacobian

```
>> J = jacob0(left, q);
>> about J
J [double] : 6x7 (336 bytes)
```

# Over-actuated robots

- An **over-actuated robot** is a type of robotic system that has **more actuators** (motors or drives) than necessary to control its **degrees of freedom**. This redundancy allows the robot to perform tasks more flexibly and robustly, as it can adapt to various conditions and compensate for failures in individual actuator.

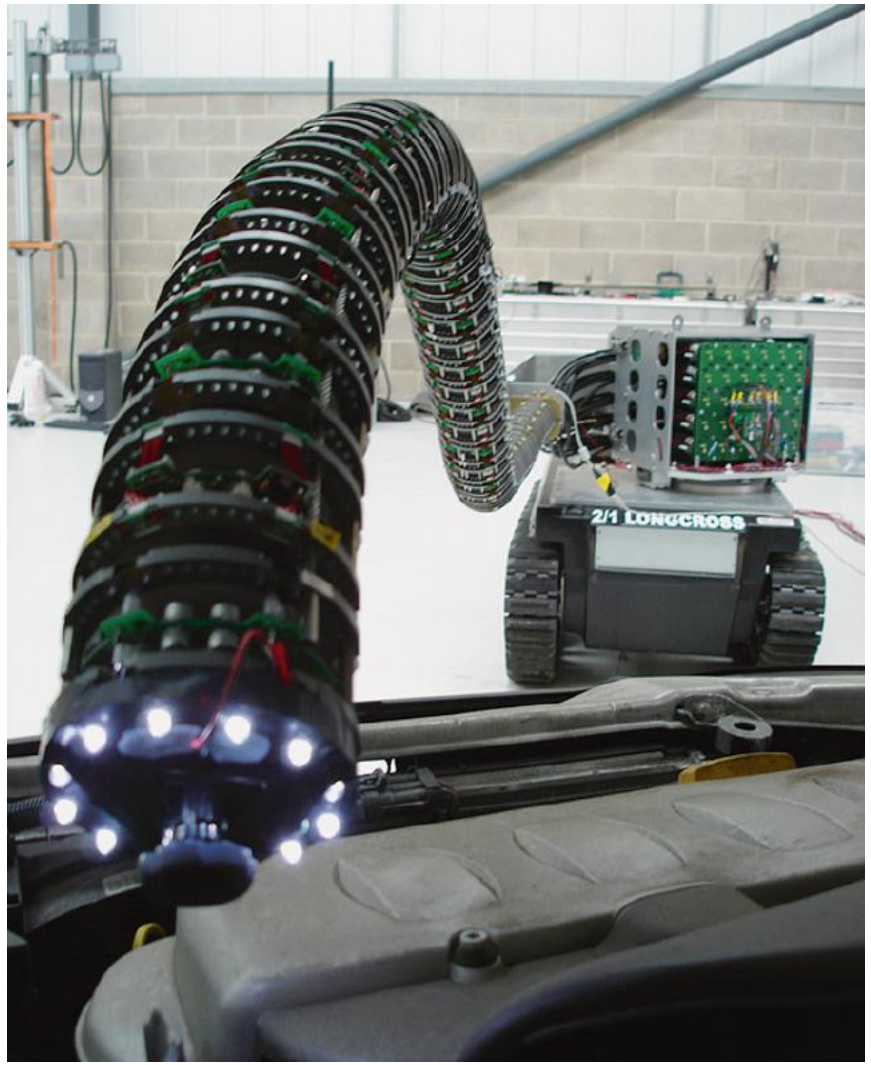



**Fig. 8.9.**
20-DOF snake-robot arm:
2.5 m reach, 90 mm diameter
and payload capacity of 25 kg
(image courtesy of OC Robotics)

# Jacobian and number of robot joints

https://robotacademy.net.au/lesson/jacobian-and-number-of-robot-joints/

## Jacobian shape summary

Used with permission of ABB

J(q)

- Fully actuated (6x6)
  - ➥ full access to SE(3)
  - ➥ Jacobian is square

Copyright © 2014 Adept Technology, Inc. The usage of this image in no manner is to be construed as Adept's endorsement of QUT or any QUT product or service

J(q)

- Under actuated (6xN), N < 6
  - ➥ limited access to SE(3)
  - ➥ remove some spatial degrees of freedom

J(q)

- Over actuated (6xN), N > 6
  - ➥ full access to SE(3)
  - ➥ use pseudo inverse
  - ➥ has **spare** joints
  - ➥ can do null-space motion

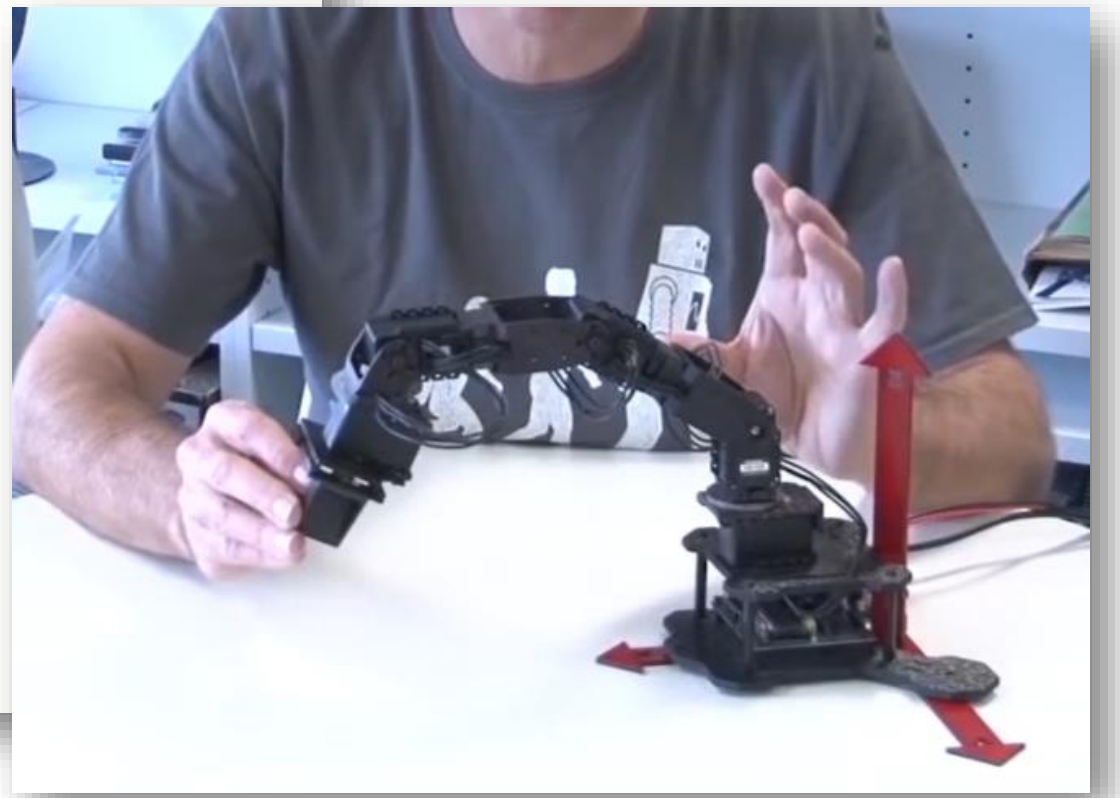
# Shape of the Jacobian

## Shape of the Jacobian

- The Jacobian has
  - one column per joint ☞ **N** columns
  - one row per spatial velocity component ☞ 6 rows for motion in 3D

$$\begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \mathbf{J}(q) \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_N \end{pmatrix}$$

6xN



# Underactuated arm

- Consider a robot with only 4 joints

$$\begin{pmatrix} v_x \\ v_y \\ v_z \\ \cancel{\omega_x} \\ \cancel{\omega_y} \\ \omega_z \end{pmatrix} = \begin{pmatrix} J_{1,1} & J_{1,2} & J_{1,2} & J_{1,4} \\ J_{2,1} & J_{2,2} & J_{1,3} & J_{2,4} \\ J_{3,1} & J_{3,2} & J_{3,3} & J_{3,4} \\ J_{4,1} & J_{4,2} & J_{4,3} & J_{4,4} \\ J_{5,1} & J_{5,2} & J_{5,3} & J_{5,4} \\ J_{6,1} & J_{6,2} & J_{6,3} & J_{6,4} \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{pmatrix}$$

6x4

- Jacobian is not square – cannot be inverted
- Choose velocity that we don't want to control

$N < 6$


Copyright © 2014 Adept Technology, Inc.
The usage of this image in no manner is to be construed as Adept's endorsement of QUT or any QUT product or service

# Overactuated arm

- Consider a robot with 100 joints

6x100

$$\begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} J_{1,1} & J_{1,2} & \cdots & J_{1,100} \\ J_{2,1} & J_{2,2} & \cdots & J_{2,100} \\ \vdots & \vdots & \ddots & \cdots \\ J_{6,1} & J_{6,2} & \cdots & J_{6,100} \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_{100} \end{pmatrix}$$

$N > 6$

- It cannot be inverted
- But we can take the **pseudo inverse**

100x100

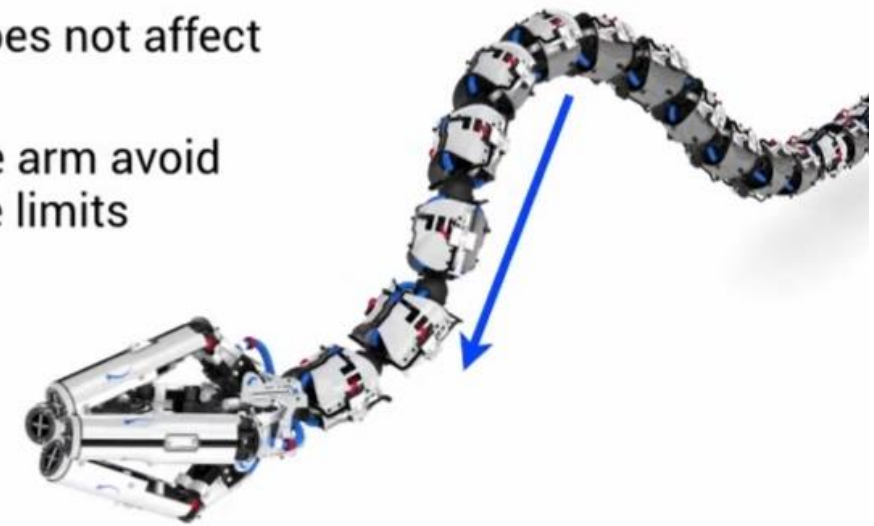
$$\dot{\boldsymbol{q}} = \mathbf{J}(\boldsymbol{q})^{+}\boldsymbol{\nu} \qquad \mathbf{J}^{+} = \left(\mathbf{J}^{T}\mathbf{J}\right)^{-1}\mathbf{J}^{T}$$

# Null space motion

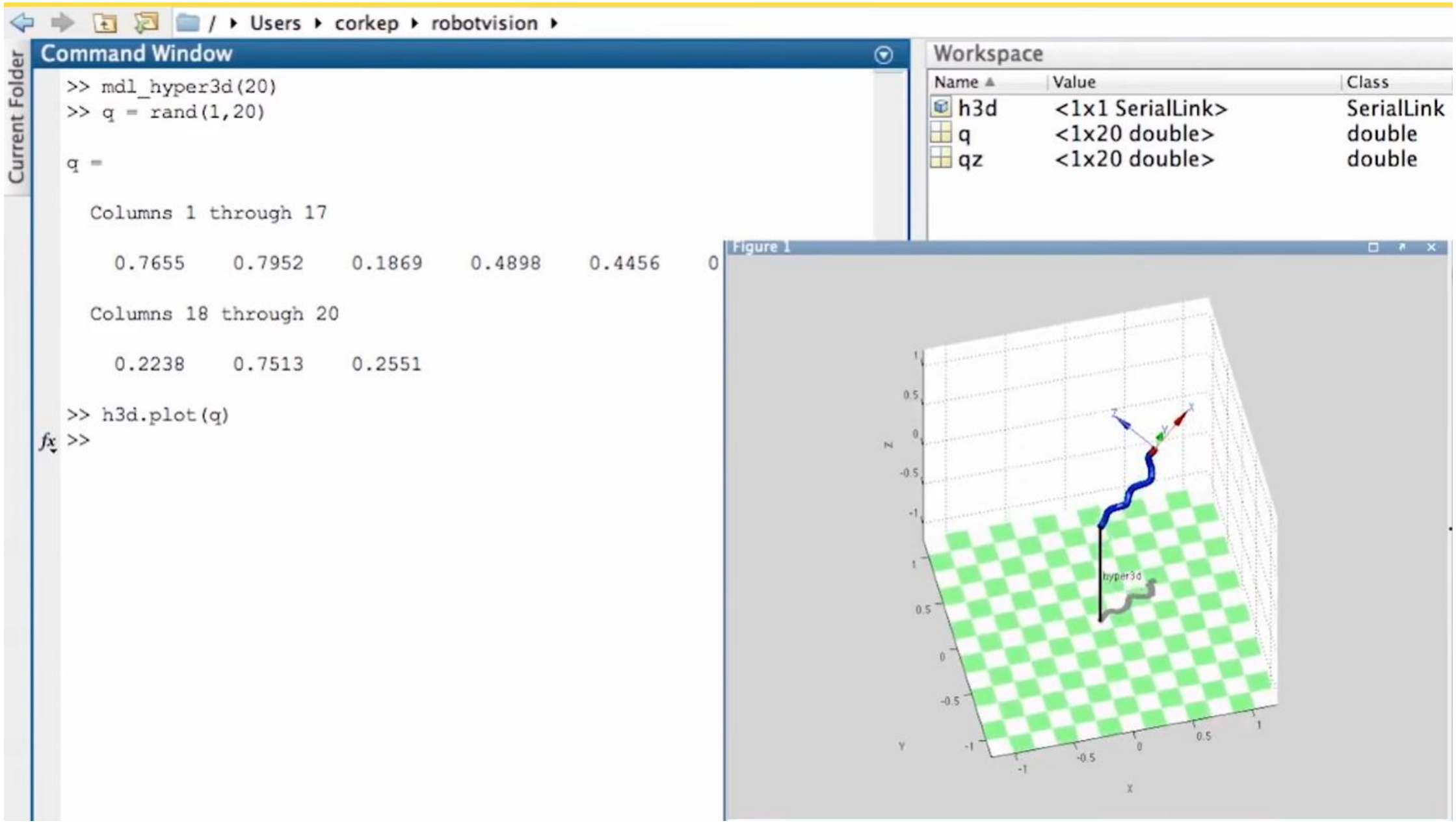
$$\dot{q} = \mathbf{J}(q)^{+} v + \mathbf{N}\mathbf{N}^{+}\dot{q}_{\text{ns}}$$

$N > 6$

- Motion of joints that does not affect the end-point pose
- Can be used to help the arm avoid obstacles or joint angle limits



- $\mathbf{N}$ is the null-space of $\mathbf{J}(q)$
  - the set of vectors $\mathbf{N} = \{n_1, n_2, \cdots n_M\}$
  - such that $\mathbf{J}(q)n_i = \mathbf{o}$


/ ▸ Users ▸ corkep ▸ robotvision ▸
Current Folder
Command Window
>> mdl_hyper3d(20)
>> q = rand(1,20)
q =
Columns 1 through 17
0.7655 0.7952 0.1869 0.4898 0.4456 0
Columns 18 through 20
0.2238 0.7513 0.2551
>> h3d.plot(q)
>>
Workspace
Name ▲ Value Class
h3d <1x1 SerialLink> SerialLink
q <1x20 double> double
qz <1x20 double> double
Figure 1
hyper3d
Z
Y
X

# 8.5 Force Relationships

## 8.5 Force Relationships

In Sect. 3.2.2 we introduced wrenches $\boldsymbol{W} = (f_x, f_y, f_z, m_x, m_y, m_z) \in \mathbb{R}^6$ which are a vector of forces and moments.

### 8.5.1 Transforming Wrenches to Joint Space

The manipulator Jacobian transforms joint velocity to an end-effector spatial velocity according to Eq. 8.2 and the Jacobian transpose transforms a wrench applied at the end-effector to torques and forces experienced at the joints▸

$$\boldsymbol{Q} = {}^0\boldsymbol{J}(\boldsymbol{q})^T \, {}^0\boldsymbol{W} \tag{8.9}$$

Derived through the principle of virtual work, see for instance Spong et al. (2006, sect. 4.10).

where $\boldsymbol{W}$ is a wrench in the world coordinate frame and $\boldsymbol{Q}$ is the generalized joint force vector. The elements of $\boldsymbol{Q}$ are joint torque or force for revolute or prismatic joints respectively.

The mapping for velocity, from end-effector to joints, involves the inverse Jacobian which can potentially be singular. The mapping of forces and torques, from end-effector to joints, is different – it involves the transpose of the Jacobian which can never be singular. We exploit this property in the next section to solve the inverse-kinematic problem numerically.

For the Puma 560 robot in its nominal pose, see Fig. 8.2, a force of 20 N in the world *y*-direction results in joint torques of

```
>> tau = p560.jacob0(qn)' * [0 20 0 0 0 0]';
>> tau'
ans =
   11.9261    0.0000    0.0000         0         0         0
```

The force pushes the arm *sideways* and only the waist joint will rotate in response – experiencing a torque of 11.93 N m due to a lever arm effect. A force of 20 N applied in the world *x*-direction results in joint torques of

```
>> tau = p560.jacob0(qn)' * [20 0 0 0 0 0]';
>> tau'
ans =
    3.0010    0.2871    6.3937         0         0         0
```

which is pulling the end-effector away from the base which results in torques being applied to the first three joints.

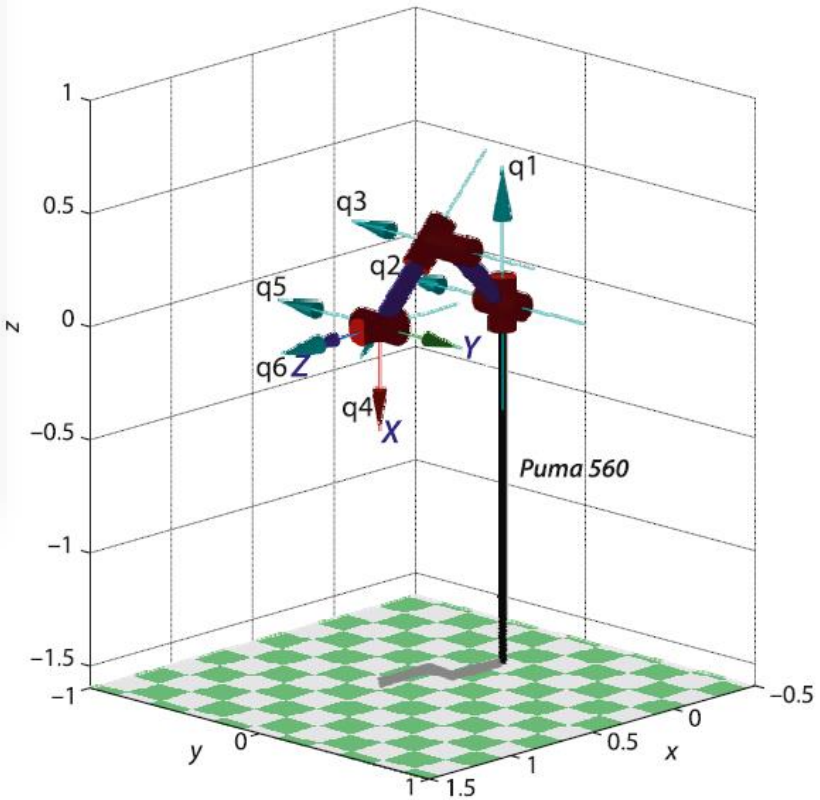



Fig. 8.2.
Puma robot in its nominal pose qn. The end-effector *z*-axis points in the world *x*-direction, and the *x*-axis points downward

# Analytic Jacobian matrix

- Created Jacobians to map angular velocity to roll-pitch-yaw or Euler angle rates, and these were used to form the **analytic Jacobian matrix**.

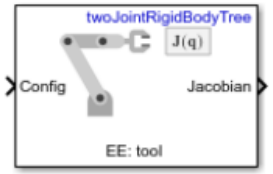
https://www.mathworks.com/help/robotics/ref/getjacobian.html

## Get Jacobian

Geometric Jacobian for robot configuration

R2024b

expand all in page




**Libraries:**

Robotics System Toolbox / Manipulator Algorithms

# 8.6 Inverse Kinematics: a General Numerical Approach

## 8.6 Inverse Kinematics: a General Numerical Approach

In Sect. 7.2.2.1 we solved the inverse kinematic problem using an explicit solution that required the robot to have 6 joints and a spherical wrist. For the case of robots which do not meet this specification, for example those with more or less than 6 joints, we need to consider a numerical solution. Here we will develop an approach based on the forward kinematics and the Jacobian transpose which we can compute for any manipulator configuration since these functions have no singularities.

### 8.6.1 Numerical Inverse Kinematics

The principle is shown in Fig. 8.10 where the robot in its current configuration is drawn solidly and the desired configuration is faint. From the overlaid pose graph the error between actual $\xi_E$ and desired pose $\xi_E^*$ is $\xi_\Delta$ which can be described by a spatial displacement as discussed in Sect. 3.1.4

$$^E\boldsymbol{\Delta} = \Delta\left(\xi_E, \xi_E^*\right) = \left(\boldsymbol{t}, \hat{\boldsymbol{v}}\theta\right) \in \mathbb{R}^6$$

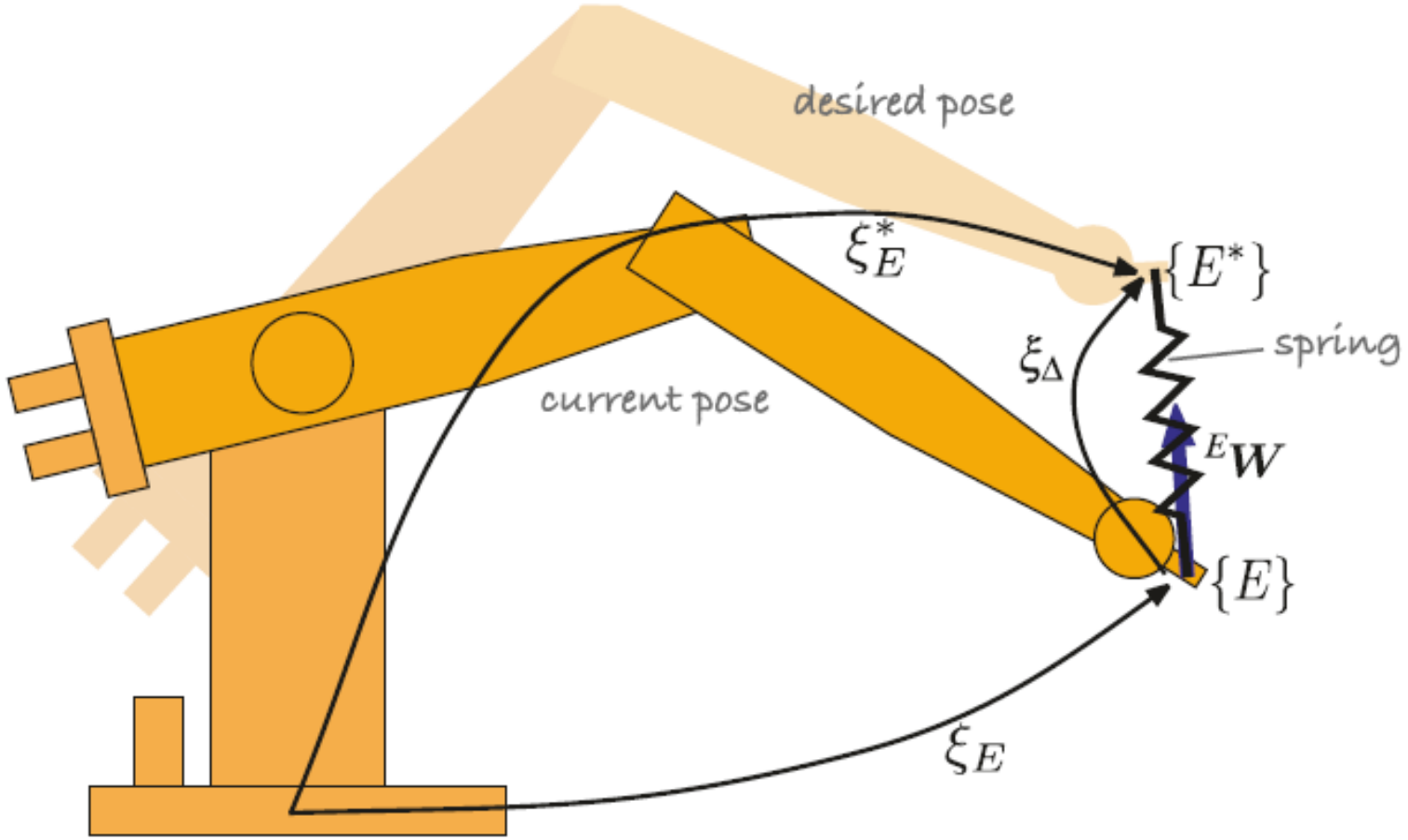



Fig. 8.10.
Schematic of the numerical inverse kinematic approach, showing the current $\xi_E$ and the desired $\xi_E^*$ manipulator pose

# 8.7 Advanced topic

## 8.7 Advanced Topics

### 8.7.1 Computing the Manipulator Jacobian Using Twists

In Sect. 7.1.2.2 we computed the forward kinematics as a product of exponentials based on the screws representing the joint axes in a zero-joint angle configuration. It is easy to differentiate the product of exponentials with respect to motion about each screw axis which leads to the Jacobian matrix

$$ {}^0\boldsymbol{J}^{\mathcal{V}} = \left( S_1 \;\; \mathrm{Ad}\left(e^{[S_1]q_1}\right) S_2 \;\; \cdots \;\; \mathrm{Ad}\left(e^{[S_1]q_1} \cdots e^{[S_{N-1}]q_{N-1}}\right) S_N \right) $$

for velocity in the world coordinate frame. The Jacobian is very elegantly expressed and can be easily built up column by column. Velocity in the end-effector coordinate frame is related to joint velocity by the Jacobian matrix

$$ {}^E\boldsymbol{J}^{\mathcal{V}} = \mathrm{Ad}\left({}^E\xi_0\right) {}^0\boldsymbol{J}^{\mathcal{V}} $$

where Ad $(\cdot)$ is the adjoint matrix introduced in Sect. 3.1.2.

> However, compared to the Jacobian of Sect. 8.1, these Jacobians give the velocity of the end-effector as a *velocity twist*, not a spatial velocity as defined on page 65.

To obtain the Jacobian that gives spatial velocity as described in Sect. 8.1 we must apply a velocity transformation

$$ {}^0\boldsymbol{J} = \begin{pmatrix} \boldsymbol{I}_{3\times3} & -\left[{}^0\boldsymbol{t}_E\right]_\times \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{pmatrix} {}^0\boldsymbol{J}^{\mathcal{V}} $$

**Robotics, Vision and Control** – FUNDAMENTAL ALGORITHMS IN **MATLAB**

Course No. M4609301
Introduction to Robotics
機器人學概論

# Matlab_Chapter 2
# Representing Position and Orientation

SHUHAO LIANG

GRADUATE INSTITUTE OF INTELLIGENT MANUFACTURING TECHNOLOGY

NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

2024

# twists

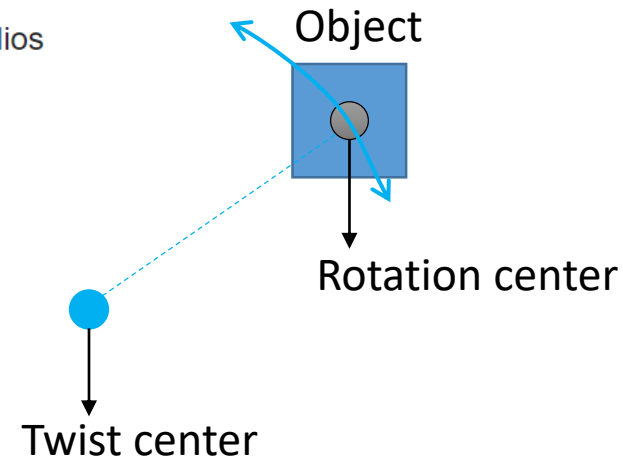
# Week05

https://en.wikipedia.org/wiki/Twist

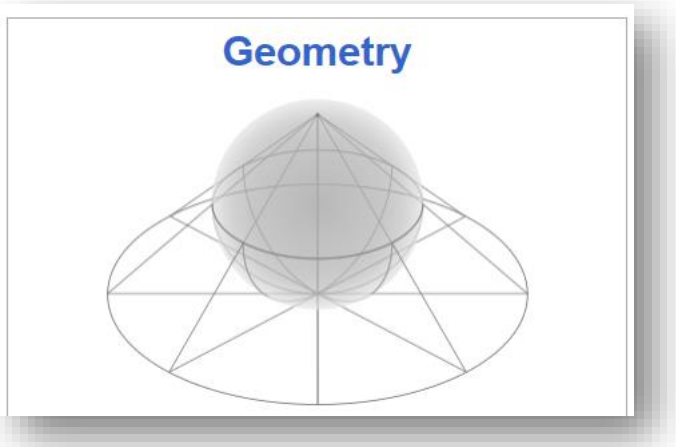
## Mathematics, science, and technology [ edit ]

- Twist (mechanics), the torsion of an object
- Twist (differential geometry), a geometric quantity associated with a ribbon
- Twists of curves, a method of deriving related curves
- Twist (rational trigonometry), in Wildberger's *Divine Proportions: Rational Trigonometry to Universal Geometry*
- Twist (screw theory), in applied mathematics and physics
- Twist (software), a test automation solution by ThoughtWorks Studios
- Ellipse Twist, a French hang glider
- Twist fungus (*Dilophospora alopecuri*)
- Twisting properties, in statistics
- Twist transcription factor, a gene protein

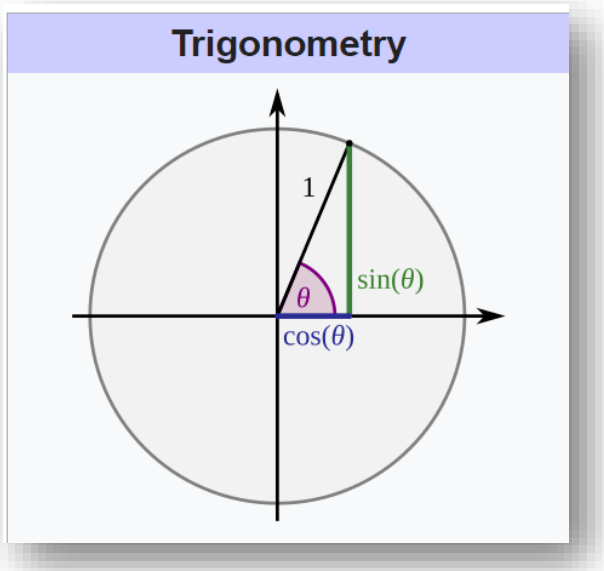
Twist (rational trigonometry)




Euclidean geometry




trigonometry



# 8.8 Wrapping Up

## 8.8 Wrapping Up

Jacobians are an important concept in robotics, relating changes in one space to changes in another. We previously encountered Jacobians for estimation in Chap. 6 and will use them later for computer vision and control.

In this chapter we have learned about the manipulator Jacobian which describes the relationship between the rate of change of joint coordinates and the spatial velocity of the end-effector expressed in either the world frame or the end-effector frame. We showed how the inverse Jacobian can be used to resolve desired Cartesian velocity into joint velocity as an alternative means of generating Cartesian paths for under- and over-actuated robots. For over-actuated robots we showed how null-space motions can be used to move the robot's joints without affecting the end-effector pose. The numerical properties of the Jacobian tell us about manipulability, that is how well the manipulator is able to move, or exert force, in different directions. At a singularity, indicated by linear dependence between columns of the Jacobian, the robot is unable to move in certain directions. We visualized this by means of the velocity and force ellipsoids.

We also created Jacobians to map angular velocity to roll-pitch-yaw or Euler angle rates, and these were used to form the analytic Jacobian matrix. The Jacobian transpose is used to map wrenches applied at the end-effector to joint torques, and also to map wrenches between coordinate frames. It is also the basis of numerical inverse kinematics for arbitrary robots and singular poses.