Aalto University
School of Electrical
Engineering

Velocity relationships, part 2

ELEC-C1320 Robotiikka

Pekka Forsman

Topics

- Manipulator jacobian (*continues from lecture 7*)
  - Jacobian condition and manipulability
  - Jacobian singularity
  - Jacobian for under-actuated robot
  - Jacobian for over-actuated robot
- Transforming Wrenches to Joint Space

Jacobian condition and manipulability

Jacobian matrix maps joint rates to end-effector Cartesian velocity. The **inverse problem** has strong practical use (*as was the case for inverse kinematics solution, i.e. what are the joint* ***position*** *values that result in the desired Cartesian tool-frame pose*). In the inverse problem the task is to determine the joint **v**elocities that are needed to achieve the required end-effector Cartesian velocity.

We can invert Eq. 8.2, p. 231 Corke's text book, and write (provided that $\boldsymbol{J}$ is square and non-singular):

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

For a N-link robot the Jacobian is a $6 \times N$ matrix so a square Jacobian requires a robot with 6 joints (as are most of the industrial robot manipulators)

A robot configuration $\boldsymbol{q}$ at which, the determinant of the Jacobian matrix equals zero, i.e. $\det(J(\boldsymbol{q})) = 0$, is described as singular or degenerate. Singularities occur when one or more axes become aligned resulting in the loss of degrees of freedom – the gimbal lock problem again. However, singularities may occur also in some other situations, such as at the boundaries of the robot motion space, e.g. when the arm is stretched.

Note that singularity is a problem related to the calculation of the inverse of the Jacobian matrix, where we have a division with the determinant of the matrix which obviously leads to numerical problems if the determinant approaches zero.

When the robot is **in** a singular configuration it becomes impossible to find velocities for the robot joints that would result in the desired Cartesian tool frame velocity in some **arbitrary** direction of the manipulator task space

On the other hand, when the robot is **close** to a singular configuration, some of the required joint velocities may become excessively large causing problems for the manipulator motion control and joint actuation system

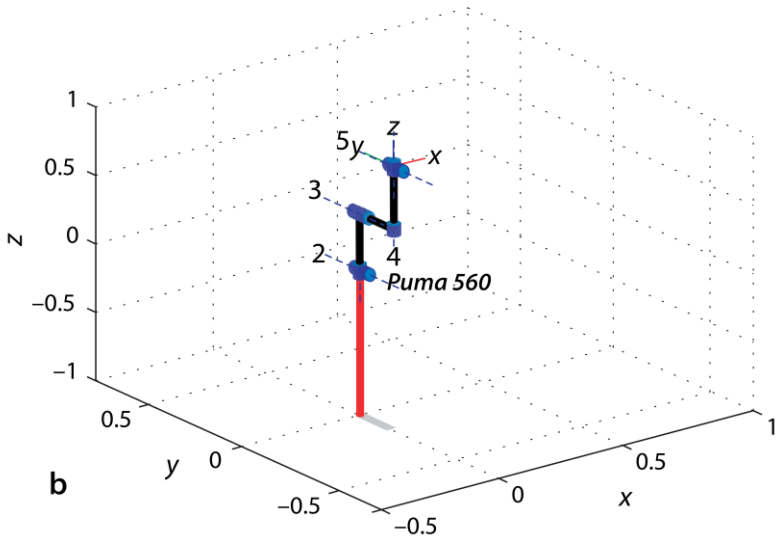
For example at the Puma's *ready* pose (see the figure) two of the wrist joints (joints 4 and 6) are aligned resulting in the loss of one degree of freedom. The Jacobian in this case is

```
>> 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
```

which is rank deficient. The rank is only five compared to a maximum of six for the 6 × 6 Jacobian:

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

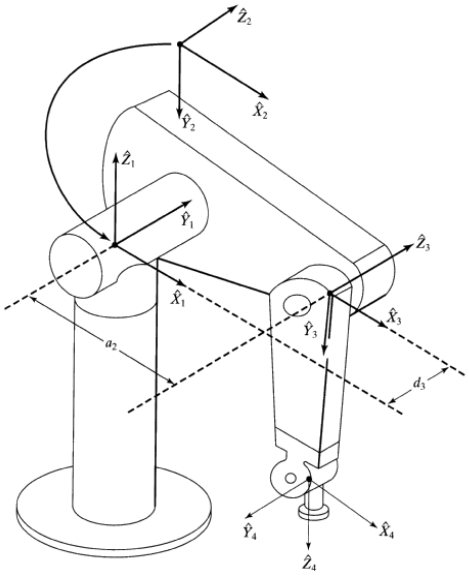
The rank deficiency and the resulting singularity can be noted by considering the columns 4 and 6 of the Jacobian matrix of Puma 560 robot at the *ready* pose. Both columns are identical indicating that the corresponding joints, 4 and 6, will cause identical Cartesian velicity (here rotation of the tool frame about the base frame z-axis)

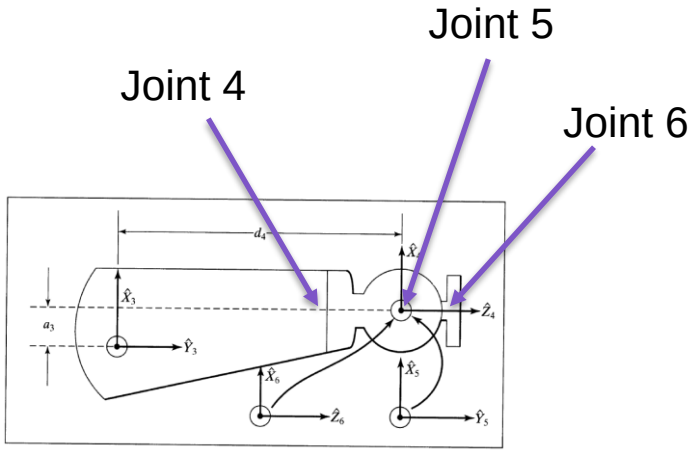


It is interesting to note that "for the Puma 560 robot arm joints 4 and 6 are the only ones that can become **aligned** and lead to singularity. The offset distances, d_j and a_j between links prevents other axes becoming aligned."
Corke's text book, p.234

However, we can also have so called workspace boundary singularities, eg. In Puma 560 when shoulder and elbow links are aligned

Puma 560 robot:





If the robot is close to, but not actually at, a singularity we encounter problems where some Cartesian end-effector velocities require very high joint rates – at the singularity those rates will go to infinity.

For example, if we choose a pose slightly away from *ready* pose which we just showed was singular. To do this, we set q_5 to a small but non-zero value of 5 deg

```
>> q = qr
>> q(5) = 5 * pi/180
q =
         0    1.5708   -1.5708         0    0.0873         0
```

and calculate the corresponding Jacobian:

```
>> J=p560.jacob0(q);
```

Now, we can calculate joint velocities required to move the robot tool frame origin at 10 cm/s in the positive z-axis direction w.r.t. robot base frame

```
>> qd = inv(J)*[0 0 0.1 0 0 0]' ;
>> qd'
ans =   -0.0000   -4.9261    9.8522    0.0000   -4.9261         0
```

This, relative slow Cartesian space motion would require very high-speed motion of the shoulder and elbow – the elbow would have to move at 9.85 rad/s or nearly 600 deg/s. (*Note that in practice we would not be able move very far in the positive z-direction, maybe only a few millimeters, because the arm is almost stretched*)

The reason is that although the robot is no longer at a singularity, the determinant of the Jacobian is still very small

```
>> det(J)
ans =
   -1.5509e-05
```

Alternatively we can say that its condition number is very high and the Jacobian is *poorly conditioned*.

```
>> cond(J)
ans =
   235.2498
```

This leads to the concept of manipulability. Consider the set of 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

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

which is the equation of points on the surface of a 6-dimensional ellipsoid in the 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.

To illustrate this we return the robot to the *nominal* configuration and compute the ellipsoid corresponding to *translational* velocity in the world frame

Commands to create fig a:

Linear velocity

```
>> J = p560.jacob0(qn);
>> J = J(1:3, :);
>> plot_ellipse(J*J')
```

Commands to create fig b:

Rotational velocity

```
>> J = p560.jacob0(qr);
>> J = J(4:6, :);
>> plot_ellipse(J*J')
```

Fig **a** indicates that at the nominal configuration origin of the tool-frame can achieve higher velocity in the z- than in the x- and y-directions. Fig **b** indicates that the tool-frame can be rotated around z- and y-axis but not around x-axis (w.r.t the Puma 560 base frame)

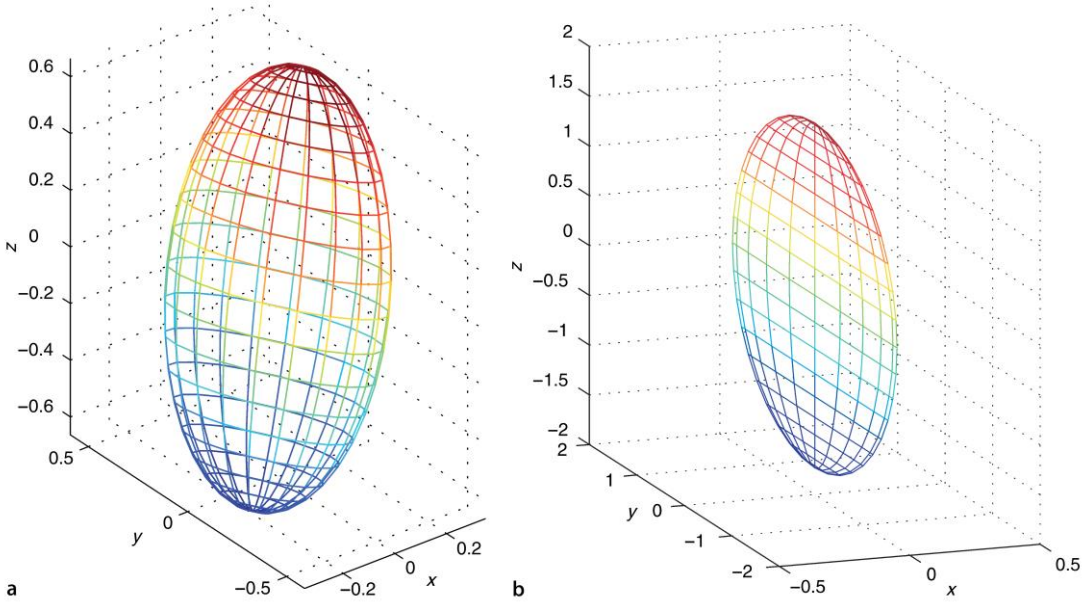


Fig. 8.2 (p. 179, Corke's text book). End-effector velocity ellipsoids. **a** Translational velocity ellipsoid for the *nominal* pose, **qn**; **b** rotational velocity ellipsoid for a (near) singular *ready* pose **qr**, the ellipsoid is an elliptical plate

This is quite obvious result if you look at the robot in the **qr**-configuration, slide 5, all robot joint axes are oriented either along y- or z-axis of the base frame

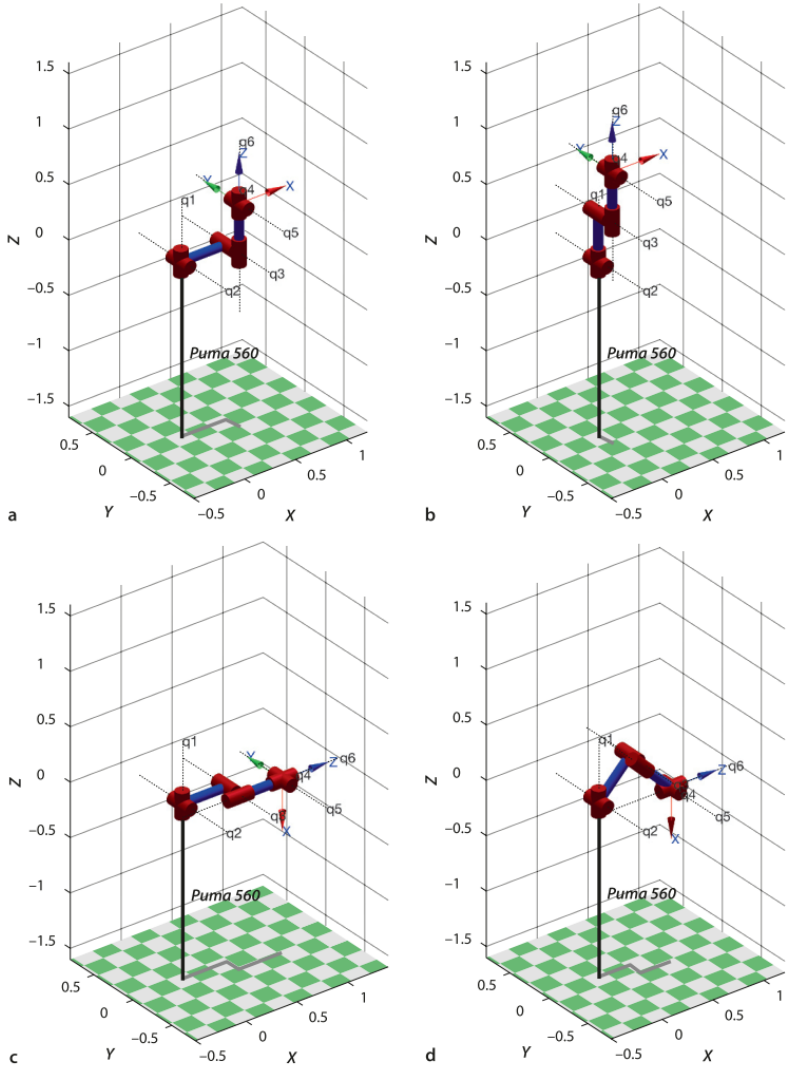


Fig. 7.6.
The Puma robot in 4 different poses. **a** *Zero angle*; **b** *ready* pose; **c** *stretch*; **d** *nominal*

Note that here we have not applied any tool transformation to move the TCP (tool center point) away from the middle of the robot "wrist" were the origins of the last three link frames (4,5,6) locate.

Moving the tool frame away from the wrist center **could** change the velocity ellipsoids presented on the previous slide

Jacobian singularity

For the case of a square Jacobian where det(J(q)) = 0 we cannot solve Eq. 8.3 directly. One simple strategy to deal with singularity is to replace the inverse with the damped inverse Jacobian:

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

where p is a small constant added to the diagonal which places a *floor* under the deter- minant. However, this will introduces some error in $\dot{\boldsymbol{q}}$ which integrated over time could lead to a significant error in the tool pose w.r.t. the desired pose.

An alternative is to use the pseudo-inverse of the Jacobian $\boldsymbol{J}^{+}$ which has the property

$$\boldsymbol{J}^{+}\boldsymbol{J} = \boldsymbol{I}$$

just as the inverse does. And it is defined as

$$\boldsymbol{J}^{+} = \left(\boldsymbol{J}^{T}\boldsymbol{J}\right)^{-1}\boldsymbol{J}^{T}$$

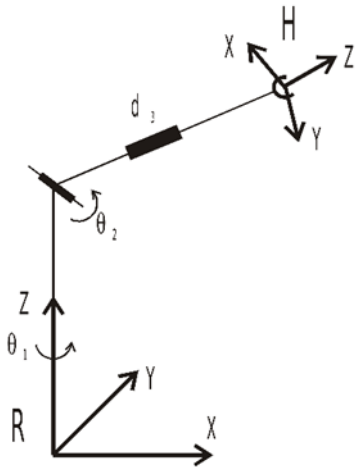
The solution for the robot joint velocities $\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})^{+}\boldsymbol{\nu}$ provides a least-squares solution for which $\|\boldsymbol{J}\dot{\boldsymbol{q}} - \boldsymbol{\nu}\|$ is smallest.

Yet another approach is to delete from the Jacobian all those columns that are linearly dependent on other columns. This is effectively locking the joints corresponding to the deleted columns. This will change the system to be underactuated (*if the original system has 6 DOF*). However, if we lock one joint of a **7 DOF arm** such as the **KUKA LWR 4+** we change an over-actuated arm to become a fully actuated one.

An example, singularities of an RRP-arm

The Jacobian matrix for the RRP-arm, shown in the figure, is:

$$\boldsymbol{J} = \begin{bmatrix} -s\theta_1 c\theta_2 d_3 & -c\theta_1 s\theta_2 d_3 & c\theta_1 c\theta_2 \\ c\theta_1 c\theta_2 d_3 & -s\theta_1 s\theta_2 d_3 & s\theta_1 c\theta_2 \\ 0 & c\theta_2 d_3 & s\theta_2 \end{bmatrix}$$



The determinant of which is: $\det J(q) = d_3^2 c\theta_2$

Singularities, caused by the determinant approaching zero occur when:

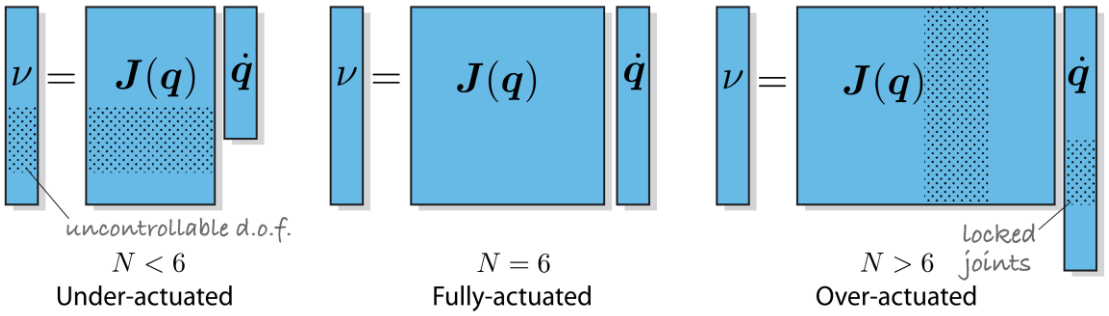
- $\Theta_2 = 90^\circ$ (*the upper arm points directly upwards*)
- $d_3=0$ (*third link is fully retracted – in practise not very likely situation*)

When d_3=0 the upper arm would have zero length and therefore the origin of H-frame would be located in the middle of shoulder joint and consequently could not be moved with a non-zero linear velocity by rotating joints 1 and 2.

Jacobian for under-actuated robot

The case of number of columns of the Jacobian matrix $N < 6$ is referred to as under-actuated robot, and $N > 6$ is over-actuated or redundant. (*Note that, in case of a full 3D task space, the number of rows is 6, i.e. corresponding to* v_x, v_y, v_z, ω_x, ω_y, ω_z)

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 ***v*** and ***J*** – accepting that some Cartesian degrees of freedom are not controllable given the low number of joints.



An under-actuated robot has $N < 6$, and a Jacobian that is taller than it is wide. For example the two-link manipulator at a nominal pose

```
>> mdl_planar2
>> qn = [1 1];
```

and the Jacobian is

```
>> J = p2.jacob0(qn)
J =
   -1.7508   -0.9093
    0.1242   -0.4161
         0         0
         0         0
         0         0
    1.0000    1.0000
```

We cannot solve the inverse problem Eq. 8.3 using the pseudo-inverse since it will attempt to satisfy motion constraints that the manipulator cannot meet.

We have to confront the reality that we have *only* two degrees of freedom which we will use to control just v_x and v_y. We rewrite Eq. 8.2 in partitioned form as

$$\boldsymbol{\nu} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \quad (8.2)$$

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

and taking the top partition, i.e. the first two rows, we get

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \boldsymbol{J}_{xy} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix}$$

where $\boldsymbol{J}_{xy}$ is a 2 × 2 matrix, which we can invert (if $\det(\boldsymbol{J}_{xy}) \neq 0$) to get

$$\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} = \boldsymbol{J}_{xy}^{-1} \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

Jacobian for over-actuated robot

For the over-actuated case the system of equations is over-constrained and we could find a least squares solution. Alternatively we can *square up* the Jacobian to make it invertible by deleting some columns – effectively *locking* the corresponding axes.

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

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

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

Transforming Wrenches to Joint Space

In the earlier discussion of motion we introduced the concept of a spatial velocity $v = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$.

For forces there is a spatial equivalent called a **wrench** $\boldsymbol{W} = (f_x, f_y, f_z, m_x, m_y, m_z) \in R^6$ which is **a vector of forces and moments**.

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

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

Interestingly this mapping from external quantities (the wrench) to joint quantities (the generalized forces) can never be singular as it can be for mapping Cartesian velocities to manipulator joint velocities.

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

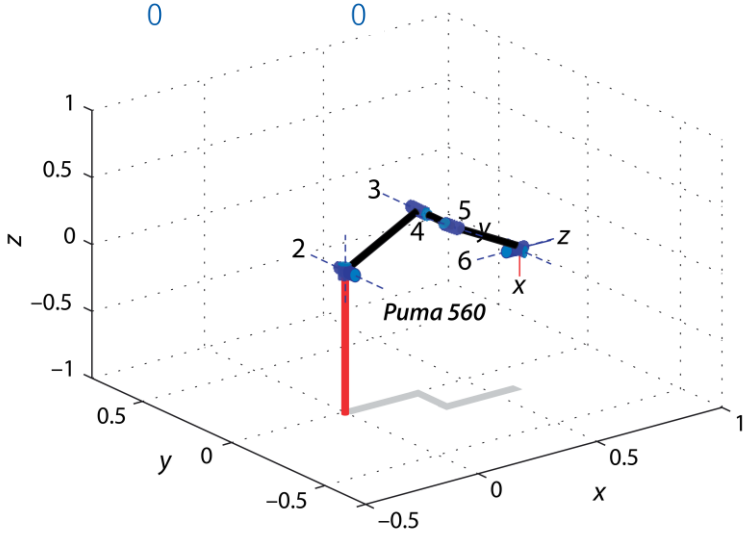
If the wrench is defined in the end-effector coordinate frame then we use instead

$$Q = {}^{E}J(q)^{T}\,{}^{E}W \qquad (8.10)$$

For the Puma 560 robot in its nominal pose, see fig., 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 feel the pressure in response – experiencing a torque of 11.93 Nm due to a lever arm effect.



Recommended reading:

Peter Corke, Robotics, Vision and Control, Fundamental Algorithms in MATLAB, Second Edition, Springer, 2017, pages 234-237, 240-244.

Craig, J.J, Introduction to Robotics: Mechanics and Control, Third Edition, Prentice Hall, 2005, pages 51-52, 156-159.