Aalto University
School of Electrical
Engineering

Dynamics and control of serial-link manipulator

ELEC-C1320 Robotics

Pekka Forsman

Topics

- Dynamics of serial-link manipulator
  - Equations of Motion (inverse dynamics)
  - Forward Dynamics (dynamic simulation)

- Manipulator Joint Control
  - Independent Joint Control
  - Rigid-Body Dynamics Compensation

- Example problems

Dynamics of serial-link manipulator

Dynamics of a serial-link manipulator deals with the forces and torques affecting the links and the joints of a mechanism when it is in a given configuration motionless or moving with a certain velocity and acceleration.

When the manipulator is **not moving** the forces and torques are due to gravity and, in case the mechanism is in contact with another object (with its' tool) due to the counterforces and -torques of the contact force/moment.

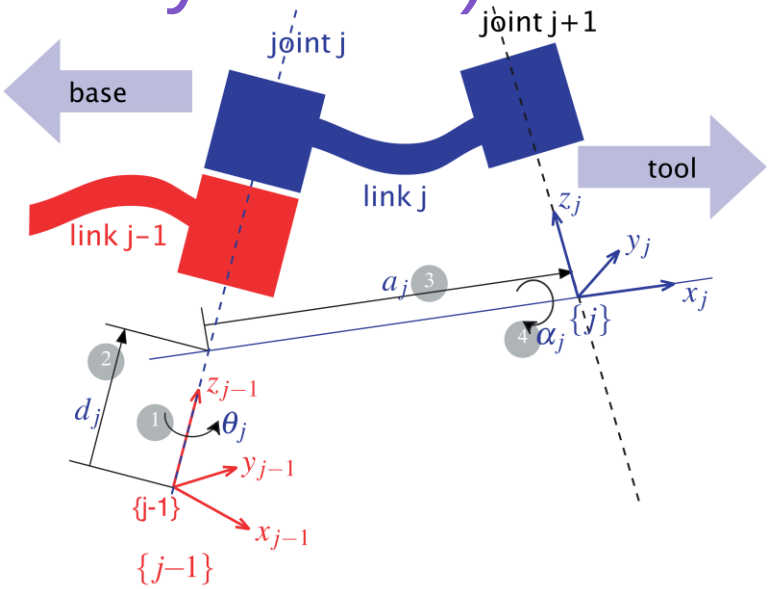
When the mechanism is **moving** the forces and torques affecting the links and the joints of the mechanism can be described (in addition to gravity and contact force related terms) with terms due to acceleration of inertia and gyroscopic coupling between links.

The dynamic forces and torgues affecting the mechanism are a function of the motion state as well as the joint configuration of the mechanism at the current instant of time.

Equations of Motion (inverse dynamics)

Consider the motor which actuates the j^{th} revolute joint of a serial-link manipulator. We recall that joint j connects link $j-1$ to link j. The motor exerts a **torque** that causes the outward link, j, to rotationally accelerate but it also exerts a **reaction torque** on the inward link $j-1$.

Gravity acting on the outward links j to N exert a weight force, and rotating links also exert **gyroscopic forces** on each other. The **inertia** that the motor experiences, when it accelerates, is a function of the configuration of the outward links.



These forces and moments (torques) acting on the manipulator links and joints can be expressed in the form of a **set of coupled differential equations** and presented in matrix form as:

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W \qquad (9.8)$$

This equation describes the manipulator rigid-body dynamics and is known as the **inverse dynamics** – given the pose, velocity and acceleration of the mechanism (and contact forces/torques) it computes the required joint forces or torques.

The **equations of motion** can be **derived** using any classical dynamics method such as Newton's second law and Euler's equation of motion or a Lagrangian energy-based approach

A very efficient way for computing Eq. 9.8 is the **iterative Newton-Euler algorithm** which starts at the base and working outward adds the velocity and acceleration of each joint in order to determine the velocity and acceleration of each link. Then working from the tool back to the base, it computes the forces and moments acting on each link and from them calculates the joint torques. See eg. *Craig's text book, pp. 173-176.*

In this course we will not go into more details of the derivation of the equations of motion. Instead we try to analyse, in more detail, the different phenomena causing the forces and torques on a moving kinematic mechanism.

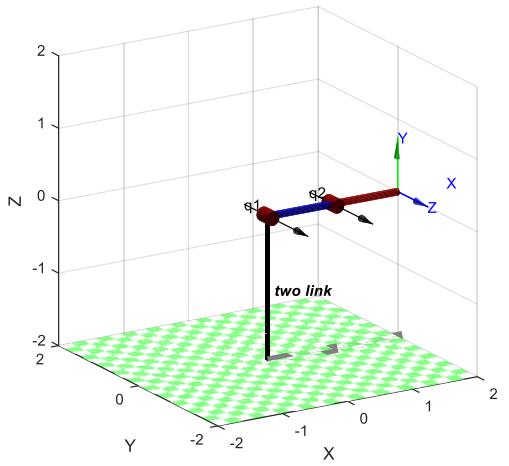
Gravity term

Gravity term is generally the dominant term in Eq. 9.8 and is present even when the robot is stationary or moving slowly.

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

An example two-link manipulator model in zero configuration:

```
>> mdl_twolink
>> twolink.plot(qz, 'jvec')
```



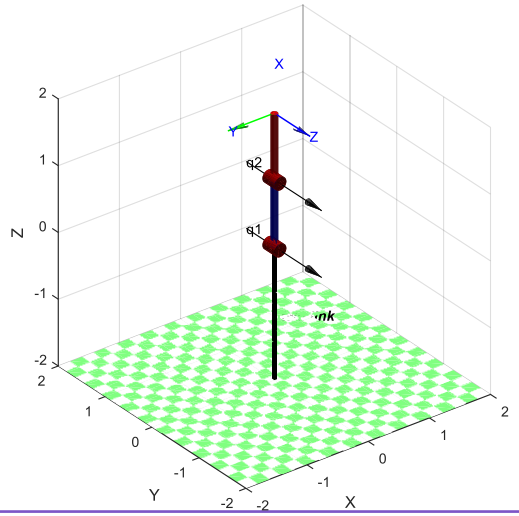
% Use the Recursive Newton-Euler dynamics equations
% to calculate the joint torques for the manipulator

```
>> twolink.rne([0 0], [0 0], [0 0])
ans =
   19.6200    4.9050
```

(*note that while robot is not moving the joint torques are due to gravity*)

Now make the angle of joint 1 equal to pi/2 radians.

```
>> twolink.plot([pi/2 0],'jvec')
```



And calculate the joint torques due to gravity

```
>> twolink.gravload([pi/2 0])
ans =
 1.0e-14 *
 0.1201      0.0300
```

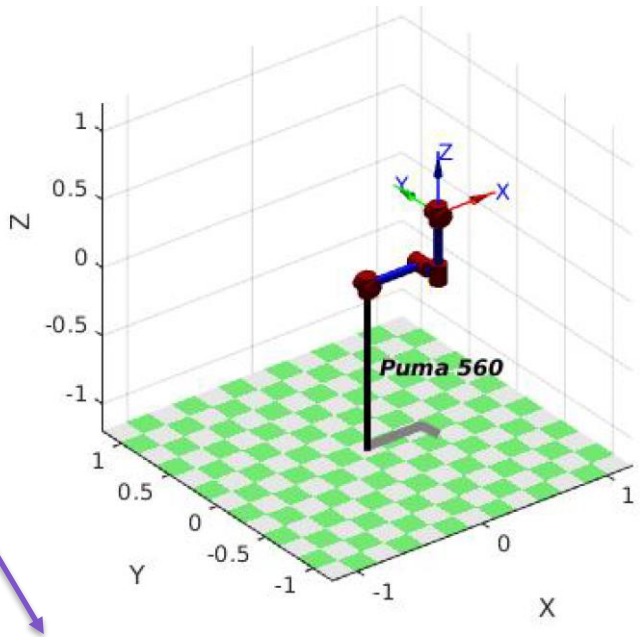
The joint torques are very small which makes sense since the gravity force vectors of the link masses intersect the joint axes making the lever arms for creating joint torques to be zero

An example Puma 560 robot model in zero configuration:

Load the Puma-560 robot model, calculate the gravity load with zero joint angles and plot the configuration.

```
>> mdl_puma560
>> p560.plot([0 0 0 0 0 0])

>> p560.gravload([0 0 0 0 0 0])
ans =
 0 37.4837 0.2489 0 0 0
```



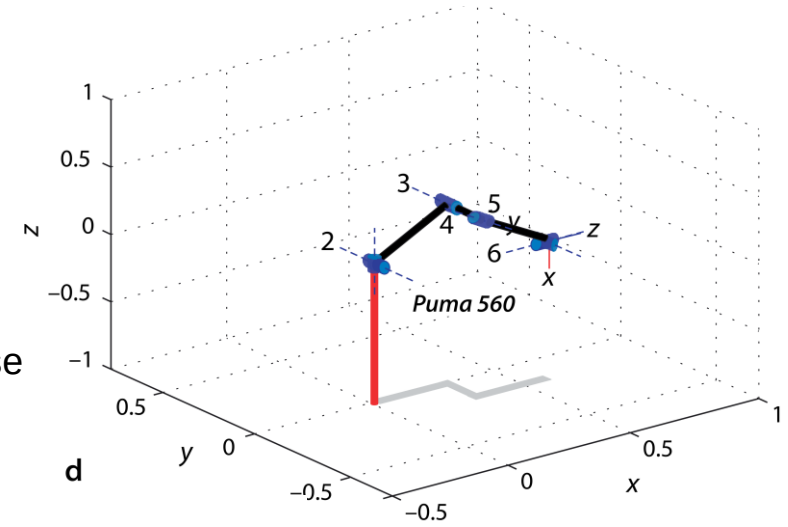
<u>Change the gravity property of the model</u> to approximately that of gravity on the moon. Now recalculate the gravity load at the zero joint angle configuration.

```
>> p560.gravity = [0 0 9.81/6];
>> p560.gravload([0 0 0 0 0 0])
ans =
 -0.0000 6.2473 0.0415 0 0 0
```

When the gravity is decreased to 1/6 of the oririginal one the joint torques are decreased in the same proportion

Inertia Matrix

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$



The inertia matrix is a function of the manipulator pose for example Puma 560 in the *nominal* configuration:

```
>> M = p560.inertia(qn)
M =
    3.6594   -0.4044    0.1006   -0.0025    0.0000   -0.0000
   -0.4044    4.4137    0.3509    0.0000    0.0024    0.0000
    0.1006    0.3509    0.9378    0.0000    0.0015    0.0000
   -0.0025    0.0000    0.0000    0.1925    0.0000    0.0000
    0.0000    0.0024    0.0015    0.0000    0.1713    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

The matrix is symmetric. The diagonal elements M_{jj} describe the inertia *seen* by joint j, that is, $Q_j = M_{jj}\ddot{q}_j$ Note that the first two diagonal elements, corresponding to the robot's waist and shoulder joints, are large since motion of these joints involves rotation of the heavy upper- and lower-arm links. The off-diagonal terms $M_{ij} = M_{ji}$, $i \neq j$ represent coupling of acceleration from joint j to the generalized force on joint i *(i.e joint torque for a rotational joint or force for a prismatic joint)*.

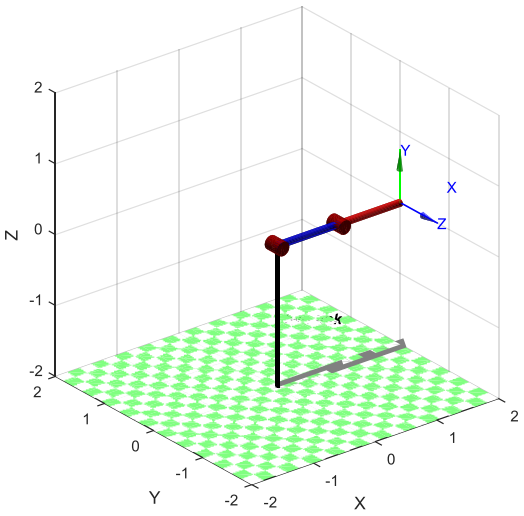
An example: inertia matrix of a two-link manipulator

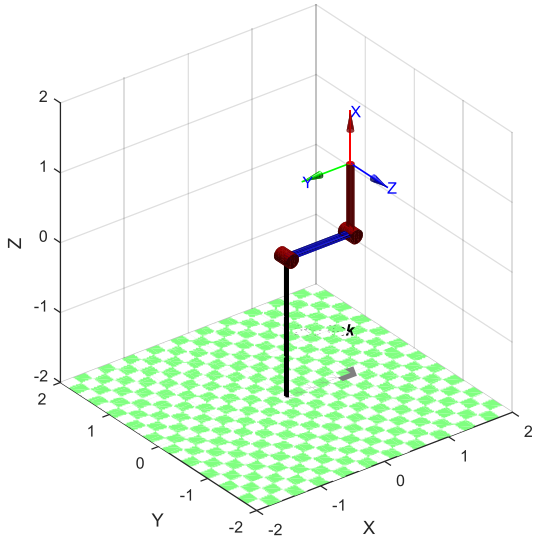
Load the two-link robot model and compute the inertia matrix for three different joint angle configurations:

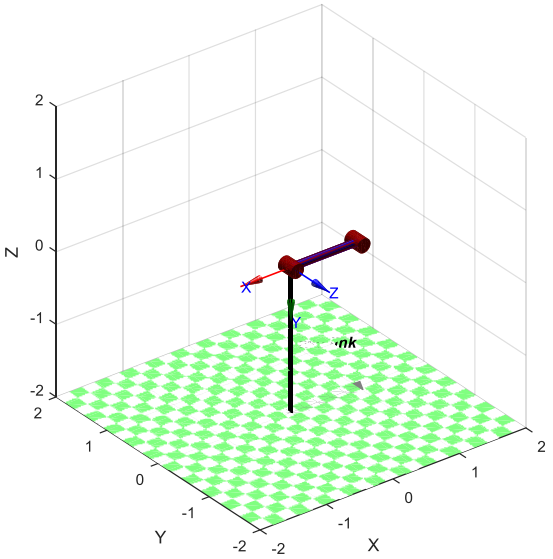
```
>> mdl_twolink
>> twolink.plot([0 0])
>> twolink.inertia([0 0])
ans =
 2.5000 0.7500
 0.7500 0.2500
```

```
>> twolink.plot([0 pi/2])
>> twolink.inertia([0 pi/2])
ans =
 1.5000 0.2500
 0.2500 0.2500
```

```
>> twolink.plot([0 pi])
>> twolink.inertia([0 pi])
ans =
 0.5000 -0.2500
 -0.2500 0.2500
```







Coriolis Matrix

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

The Coriolis matrix ***C*** is a function of joint coordinates and joint velocity. The **centripetal torques** are proportional to $\dot{q}_i^{\,2}$, while the **Coriolis torques** are proportional to $\dot{q}_i\dot{q}_j$. For example Puma 560, at the nominal pose with all joints moving at 0.5 rad s^{-1} results into following Coriolis matrix

```
>> qd = 0.5*[1 1 1 1 1 1];
```

C =

-0.1335	-0.6453	0.0848	-0.0002	-0.0014	0.0000
0.3137	0.1929	0.3857	-0.0008	-0.0001	-0.0000
-0.1804	-0.1933	-0.0005	-0.0005	-0.0014	-0.0000
0.0002	0.0003	-0.0000	0.0001	0.0001	-0.0000
-0.0001	0.0005	0.0009	-0.0001	-0.0000	-0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0

The off-diagonal terms $C_{i,j}$ represent coupling of joint j velocity to the generalized force acting on joint i. $C_{1,2}$ = -0.6453 is very significant and represents coupling from joint 2 velocity to torque on joint 1, i.e. rotation of the shoulder exerts a torque on the waist.

Since the elements of this matrix represents a coupling from velocity to joint force they have the same dimensions as viscous friction or damping, however the sign can be positive or negative.

The joint torques due the Coriolis term in this example are: -->

```
>> C*qd'
ans =
   -0.3478
    0.4457
   -0.1880
    0.0003
    0.0006
    0.0000
```

Effect of Payload

Any real robot has a specified maximum payload which is dictated by two dynamic effects. The first is that a **mass at the end of the robot will increase the inertia** seen by the joints which reduces acceleration and dynamic performance. The second is that **mass generates a weight force which the joints needs to support**. In the worst case the increased gravity torque component might exceed the rating of one or more motors. However even if the rating is not exceeded there is less torque available for acceleration which again **reduces dynamic performance**.

As an example we will add a 2.5 kg point mass to the Puma 560 which is its rated maximum payload. The centre of mass of the payload cannot be at the centre of the wrist coordinate frame, that is inside the wrist, so we will offset it 100 mm in the z-direction of the wrist frame. We achieve this by modifying the inertial parameters of the robot's last link

```
>> p560.payload(2.5, [0, 0, 0.1]);
```

```
>> M_loaded = p560.inertia(qn);
```

We see that the diagonal elements have increased significantly, for instance the elbow joint inertia has increased by 66%.

```
>> M_loaded ./ M;
ans =
    1.3363    0.9872    2.1490   49.3960   80.1821    1.0000
    0.9872    1.2667    2.9191    5.9299   74.0092    1.0000
    2.1490    2.9191    1.6601   -2.1092   66.4071    1.0000
   49.3960    5.9299   -2.1092    1.0647   18.0253    1.0000
   83.4369   74.0092   66.4071   18.0253    1.1454    1.0000
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

The off-diagonal terms have increased significantly, particularly in rows and columns four and five. This indicates that motion of joints 4 and 5, the wrist joints, which are swinging the offset mass give rise to large reaction forces (**compared to the no-payload situation**) that are *felt* by all the other robot joints.

Tool Contact Force / Base Force

If the robot **tool** is in contact with the environment the effects of the **contact forces/torques** are distributed on the joint torques (and forces in case of a prismatic joint) by the last term of Eq. 9.8. (***g*** is the **wrench vector** compricing forces and torques affecting the robot tool contact point)

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

By multiplying the vector of Cartesian forces and moments, acting on the origin of the tool-frame, with the **transpose** of the **manipulator Jacobian** we get the equivalent joint torques.

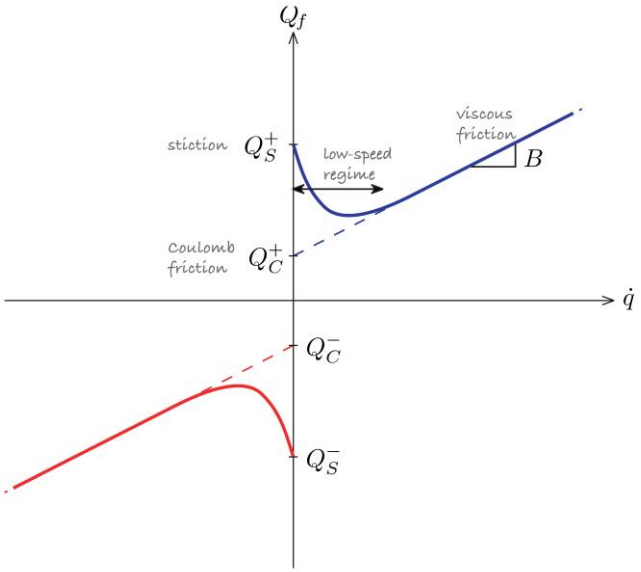
A moving robot exerts a wrench on its base, a vertical force to hold it up and other forces and torques as the arm moves around. The **base forces** are important in situations where the robot does not have a rigid base such as on a satellite in space, on a boat, an underwater vehicle or even on a vehicle with soft suspension

Friction

For most electric drive robots friction is the next most dominant joint force after gravity

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W$$

For any rotating machinery, motor or gearbox, the friction torque versus speed characteristic has a form similar to that shown in the figure. At zero speed we observe an effect known as stiction which the applied torque must exceed before rotation can
occur – a process known as breaking stiction. Once the machine is moving the stiction force rapidly decreases and viscous friction dominates. Viscous friction is shown in the figure by the dashed line.



$$Q_f = B\dot{q} + Q_C \qquad (9.4)$$

In general the friction value depends on the direction of rotation but this asymmetry is more pronounced for Coulomb than for viscous friction.

$$Q_C = \begin{cases} 0 & \dot{q} = 0 \\ Q_C^+ & \dot{q} > 0 \\ Q_C^- & \dot{q} < 0 \end{cases} \qquad (9.5)$$

Forward Dynamics (dynamic simulation)

To determine the motion of the manipulator in response to the forces and torques applied to its joints we require the forward dynamics or integral dynamics. Rearranging Eq. 9.8 we can obtain the joint accelerations:

$$\ddot{q} = M^{-1}(q)\left(Q - C(q,\dot{q})\dot{q} - F(\dot{q}) - G(q) - J(q)^T W\right) \qquad (9.10)$$

Forward dynamics is also called **dynamic simulation** because it gives us the motion of the manipulator as a function of the forces affecting on it.

To get the velocities and positions of the robot joints we have to integrate Eq. 9.10 by applying a numerical integration technique.

Manipulator Joint Control

In order for the robot end-effector to follow a desired Cartesian trajectory each of its joints must follow a specific joint-space trajectory. *Chapter 9.4 of Corke's text book (starting at p.204)* discuss the two main approaches to robot joint control: **independent control** and **model-based control**.

Independent Joint Control

A common approach to robot joint control is to **consider each joint as an independent control system** that attempts to accurately follow the joint angle trajectory. However, this is complicated by various *disturbance* torques such as gravity, velocity and acceleration coupling and friction that act on the joint.

A very common control structure is the nested control loop. The outer loop is responsible for maintaining position and determines the velocity of the joint that will minimize position error. The inner loop is responsible for maintaining the velocity of the joint as demanded by the outer loop.

Disturbance torques due to gravity and other dynamic coupling effects impact the performance of the velocity loop as do variation in the parameters of the plant being controlled, and this in turn lead to errors in position tracking.

Rigid-Body Dynamics Compensation

The velocity loop performance can be improved by adding an integral control term, or by **feedforward of the disturbance torque** which is largely predictable. *In practice control systems use both feedforward and feedback control*.

The disturbance torques can be computed according to Eq. 9.8 given knowledge of joint angles, joint velocities and accelerations, and the inertial parameters of the links.

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W \qquad (9.8)$$

We can incorporate these torques into the control law using one of two ***model-based* approaches**: feedforward control, and computed torque control.

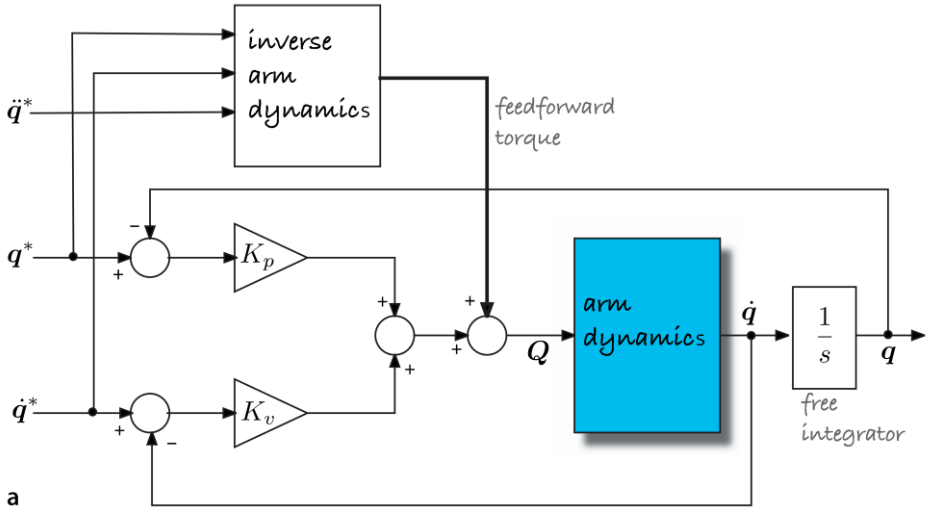
Feedforward Control

A generic equation for a torque feedforward controller is given by

$$Q^* = \underbrace{M(q^*)\ddot{q}^* + C(q^*, \dot{q}^*)\dot{q}^* + F(\dot{q}^*) + G(q^*)}_{\text{feedforward}} + \underbrace{\left\{K_v(\dot{q}^* - \dot{q}) + K_p(q^* - q)\right\}}_{\text{feedback}}$$

$$= \mathcal{D}(q^*, \dot{q}^*, \ddot{q}^*) + \left\{K_v(\dot{q}^* - \dot{q}) + K_p(q^* - q)\right\}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_v$ are the position and velocity gain (or damping) matrices respectively, and $D(\cdot)$ is the inverse dynamics function. The gain matrices are typically diagonal. The **feedforward term** provides the joint torques/forces required for the desired manipulator state $(q^*, \dot{q}^*, \ddot{q}^*)$ and the **feedback term** compensates for any errors due to uncertainty in the inertial parameters, unmodeled forces or external disturbances.

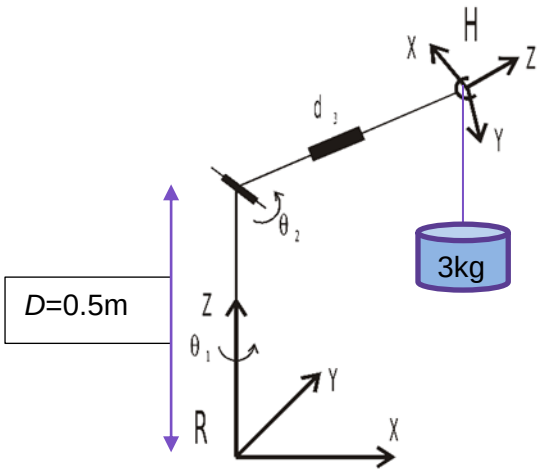


Example problems

The task is to calculate the additional torques required for the robot joints to counter the gravitational force due to a weight of 3 kg attached to the tip of the robot arm (see the figure). To solve the problem here **you must utilize the Jacobian matrix of the robot arm.**

The forward kinematics solution of the robot in the form of manipulator arm matrix is:

$$ {}^{R}T_{H} = \begin{bmatrix} -c\theta_1 s\theta_2 & s\theta_1 & c\theta_1 c\theta_2 & c\theta_1 c\theta_2 d_3 \\ -s\theta_1 s\theta_2 & -c\theta_1 & s\theta_1 c\theta_2 & s\theta_1 c\theta_2 d_3 \\ c\theta_2 & 0 & s\theta_2 & s\theta_2 d_3 + D \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



To solve the task, we need a Jacobian matrix that relates the robot joint rates to the linear velocity of the H-frame

expressed with respect to the R-frame: $$ \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \boldsymbol{J} \begin{bmatrix} \dot{\boldsymbol{\theta}}_1 \\ \dot{\boldsymbol{\theta}}_2 \\ \dot{\boldsymbol{d}}_3 \end{bmatrix} $$

So, we are now only interested in the x-, y- and z-coordinates of the origin of the Tool-frame, i.e. the tree top elements of the fourth column of the arm matrix.

x= $c\theta_1 c\theta_2 d_3$

y= $s\theta_1 c\theta_2 d_3$

z= $s\theta_2 d_3 + D$

The Jacobian matrix can now be calculated by taking partial derivatives of the coordinates of the origin of H-frame w.r.t. the joint variables:

$$\begin{bmatrix} \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} & \frac{dx}{dd_3} \\ \frac{dy}{d\theta_1} & \frac{dy}{d\theta_2} & \frac{dy}{dd_3} \\ \frac{dz}{d\theta_1} & \frac{dz}{d\theta_2} & \frac{dz}{dd_3} \end{bmatrix} = \begin{bmatrix} -s\theta_1 c\theta_2 d_3 & -c\theta_1 s\theta_2 d_3 & c\theta_1 c\theta_2 \\ c\theta_1 c\theta_2 d_3 & -s\theta_1 s\theta_2 d_3 & s\theta_1 c\theta_2 \\ 0 & c\theta_2 d_3 & s\theta_2 \end{bmatrix}$$

So, the task is to calculate torques and forces affecting joints 1, 2 and 3 (due to gravity) in two different configurations of the manipulator arm. The joint configurations to be considered are:

a) θ_1=0.0°, θ_2=0.0°, d_3=0.5m (*the total length of the upper link is described by d_3*)

a) θ_1=0.0°, θ_2=90.0°,d_3=0.5m (*the total length of the upper link is described by d_3*)

The links itself are assumed to be weightless.
The gravitational acceleration vector is pointing in the direction of negative Z_R-axis and its value is 9.81 m/s^2.

For the solution we remember that Jacobian transpose transforms a wrench applied at the end-effector, ${}^0\boldsymbol{W}$ to torques and forces experienced at the joints $\boldsymbol{Q}$:

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

The Jacobian matrix is: $\boldsymbol{J} = \begin{bmatrix} -s\theta_1 c\theta_2 d_3 & -c\theta_1 s\theta_2 d_3 & c\theta_1 c\theta_2 \\ c\theta_1 c\theta_2 d_3 & -s\theta_1 s\theta_2 d_3 & s\theta_1 c\theta_2 \\ 0 & c\theta_2 d_3 & s\theta_2 \end{bmatrix}$

and now we can form its transpose "by turning all the rows of the matrix into columns": $\boldsymbol{J^T} = \begin{bmatrix} -s\theta_1 c\theta_2 d_3 & c\theta_1 c\theta_2 d_3 & 0 \\ -c\theta_1 s\theta_2 d_3 & -s\theta_1 s\theta_2 d_3 & c\theta_2 d_3 \\ c\theta_1 c\theta_2 & s\theta_1 c\theta_2 & s\theta_2 \end{bmatrix}$

In part **a)** the manipulator was in the configuration: θ_1=0.0°, θ_2=0.0°, d_3=0.5m which corresponds the following transpose of the Jacobian matrix:

$$\boldsymbol{J^T} = \begin{bmatrix} -s\theta_1 c\theta_2 d_3 & c\theta_1 c\theta_2 d_3 & 0 \\ -c\theta_1 s\theta_2 d_3 & -s\theta_1 s\theta_2 d_3 & c\theta_2 d_3 \\ c\theta_1 c\theta_2 & s\theta_1 c\theta_2 & s\theta_2 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 1 & 0 & 0 \end{bmatrix}$$

The force/wrench vector due to the 3 kg weight (gravity) force pulling the tip of the arm in the direction of negative Z_R-axis is: $\boldsymbol{f} = \begin{bmatrix} 0 \\ 0 \\ -9.81 * 3.0N \end{bmatrix}$

And now we can calculate the additional joint load torques and forces, Q_1, Q_2 and F_3 :

$$\boldsymbol{\tau} = \boldsymbol{J}^T \boldsymbol{f} = \begin{bmatrix} Q_1 \\ Q_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 & 0.5m & 0 \\ 0 & 0 & 0.5m \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -9.81 * 3.0N \end{bmatrix} = \begin{bmatrix} 0 \\ -14{,}715Nm \\ 0 \end{bmatrix}$$

The joint torque required to counter the additional load would be the opposite, ie. $+14{,}715Nm$

In part **b)** the manipulator was in the configuration: θ_1=0.0°, θ_2=90.0°, d_3=0.5m which corresponds the following transpose of the Jacobian matrix:

$$\boldsymbol{J^T} = \begin{bmatrix} -s\theta_1 c\theta_2 d_3 & c\theta_1 c\theta_2 d_3 & 0 \\ -c\theta_1 s\theta_2 d_3 & -s\theta_1 s\theta_2 d_3 & c\theta_2 d_3 \\ c\theta_1 c\theta_2 & s\theta_1 c\theta_2 & s\theta_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -0.5m & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

And now we can calculate the additional joint load torques and forces, Q_1, Q_2 and F_3:

$$\boldsymbol{\tau} = \boldsymbol{J}^T \boldsymbol{f} = \begin{bmatrix} Q_1 \\ Q_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -0.5m & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -9.81 * 3.0N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -29{,}43N \end{bmatrix}$$

The joint force required to counter the additional load would be the opposite, ie. $+29{,}43N$

Recommended reading:

Peter Corke, Robotics, Vision and Control, Fundamental Algorithms in MATLAB, Second Edition, Springer, 2017, pages 251, 263-269, 271-274.

Craig, J.J, Introduction to Robotics: Mechanics and Control, Third Edition, Prentice Hall, 2005, pages173-176.