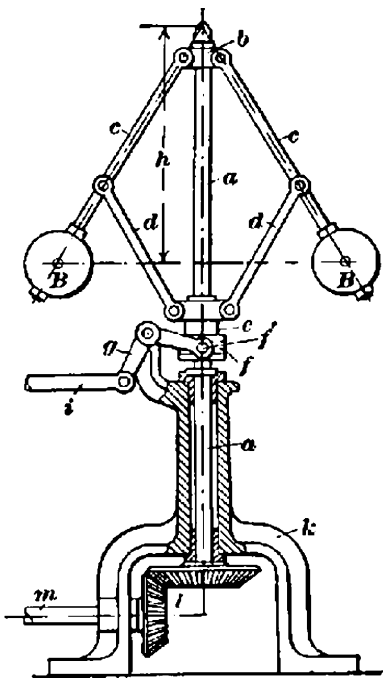
9 Dynamics and Control



In this chapter we consider the dynamics and control of a serial-link manipulator. Each link is supported by a reaction force and torque from the preceding link, and is subject to its own weight as well as the reaction forces and torques from the links that it supports.

Section 9.1 introduces the equations of motion, a set of coupled dynamic equations, that describe the joint torques necessary to achieve a particular manipulator state. The equations contains terms for inertia, gravity and gyroscopic coupling. The equations of motion provide insight into important issues such as how the motion of one joint exerts a disturbance force on other joints, how inertia and gravity load varies with configuration, and the effect of payload mass. Section 9.2 introduces real-world drive train issues such as gearing and friction. Section 9.3 introduces the forward dynamics which describe how the manipulator moves, that is, how its configuration evolves with time in response to forces and torques applied at the joints by the actuators, and by external forces such as gravity. Section 9.4 introduces control systems that compute the joint forces so that the robot end-effector follows a desired trajectory despite varying dynamic characteristics or joint flexibility.

9.1 Equations of Motion

Consider the motor which actuates the j^{th} revolute joint of a serial-link manipulator. From Fig. 7.2 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* is a function of the configuration of the outward links.

The situation at the individual link is quite complex but for the *series* of links the result can be written elegantly and concisely as a set of coupled differential equations in matrix form

$$\boldsymbol{Q} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{g} \tag{9.1}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are respectively the vector of generalized joint coordinates, velocities and accelerations, $\boldsymbol{M}$ is the joint-space inertia matrix, $\boldsymbol{C}$ is the Coriolis and centripetal coupling matrix, $\boldsymbol{F}$ is the friction force, $\boldsymbol{G}$ is the gravity loading, and $\boldsymbol{Q}$ is the vector of generalized actuator forces associated with the generalized coordinates $\boldsymbol{q}$. The last term gives the joint forces due to a wrench $\boldsymbol{g}$ applied at the end effector and $\boldsymbol{J}$ is the manipulator Jacobian. This equation describes the manipulator rigid-body dynamics and is known as the inverse dynamics – given the pose, velocity and acceleration it computes the required joint forces or torques.

These equations 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. 9.1 is the recursive 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 thus the joint torques. The recursive Newton-Euler algorithm has $O(N)$ complexity and can be written in functional form as

$$\boldsymbol{Q} = \mathcal{D}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \tag{9.2}$$

The recursive form of the inverse dynamics does not explicitly calculate the matrices M, C and G of Eq. 9.1. However we can use the recursive Newton-Euler algorithm to calculate these matrices and the Toolbox functions `inertia` and `coriolis` use Walker and Orin's (1982) *'Method 1'*. Whilst the recursive forms are computationally efficient for the inverse dynamics, to compute the coefficients of the individual dynamic terms (M, C and G) in Eq. 9.1 is quite costly – $O(N^3)$ for an N-axis manipulator.

In the Toolbox it is implemented by the `rne` method of the `SerialLink` object. Consider the Puma 560 robot

```
>> mdl_puma560
```

at the nominal pose, and with zero joint velocity and acceleration. The generalized joint forces, or joint torques in this case, are

```
>> Q = p560.rne(qn, qz, qz)
Q =
   -0.0000   31.6399    6.0351    0.0000    0.0283         0
```

Since the robot is not moving (we specified $\dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}} = 0$) these torques must be those required to *hold the robot up* against gravity. We can confirm this by computing the torques in the absence of gravity

```
>> Q = p560.rne(qn, qz, qz, [0 0 0]')
ans =
     0     0     0     0     0     0
```

where the last argument overrides the object's default gravity vector.

Like most Toolbox methods `rne` can operate on a trajectory

```
>> q = jtraj(qz, qr, 10)
>> Q = p560.rne(q, 0*q, 0*q)
```

which has returned

```
>> about(Q)
Q [double] : 10x6 (480 bytes)
```

a 10×6 matrix with each row representing the generalized force for the corresponding row of `q`. The joint torques corresponding to the fifth time step is

```
>> Q(5,:)
ans =
    0.0000   29.8883    0.2489         0         0         0
```

Consider now a case where the robot is moving. It is *instantaneously* at the nominal pose but joint 1 is moving at 1 rad s^{-1} and the acceleration of all joints is zero. Then in the absence of gravity, the joint torques

```
>> p560.rne(qn, [1 0 0 0 0 0], qz, [0 0 0]')
  -24.8240    0.6280   -0.3607   -0.0003   -0.0000         0
```

Dynamics in 3D. The dynamics of an object moving in 3 dimensions is described by two important equations. The first equation, Newton's second law, describes the translational motion in 3D

$$\boldsymbol{f} = m\dot{\boldsymbol{v}}$$

where m is the mass, $\boldsymbol{f}$ the applied force and $\boldsymbol{v}$ the velocity. The second equation, Euler's equation of motion, describes the rotational motion

$$\tau = \boldsymbol{J}\dot{\omega} + \omega \times \boldsymbol{J}\omega$$

where τ is the torque, ω is the angular velocity, and $\boldsymbol{J}$ is the rotational inertia matrix (see page 81).

Gyroscopic motion. A spinning disc has an angular momentum $\boldsymbol{h} = \boldsymbol{J}\omega$. If a torque τ is applied to the gyroscope it rotates about an axis perpendicular to both τ and $\boldsymbol{h}$ with an angular velocity $\boldsymbol{w_P}$ known as precession. These quantities are related by

$$\tau = \omega_{\boldsymbol{P}} \times \boldsymbol{h}$$

If you've ever tried to rotate the axis of a spinning bicycle wheel you will have observed this effect – "torquing" it about one axis causes it to turn in your hands in an orthogonal direction.

A strapdown gyroscopic sensor contains a high-speed flywheel which has a large $\boldsymbol{h}$. When the gyroscope is rotated a gyroscopic force is generated proportional to $\boldsymbol{w_P}$ which is measured by a force sensor.

are non zero. The torque on joint 1 is due to friction and opposes the direction of motion. More interesting is that torques have been exerted on joints 2, 3 and 4. These are gyroscopic effects (centripetal and Coriolis forces) and are referred to as velocity coupling torques since the rotational velocity of one joint has induced a torque on several other joints.

The elements of the matrices $\boldsymbol{M}$, $\boldsymbol{C}$, $\boldsymbol{F}$ and $\boldsymbol{G}$ are complex functions of the link's kinematic parameters (θ_j, d_j, a_j, α_j) and inertial parameters. Each link has ten independent inertial parameters: the link mass m_j; the centre of mass (COM) $\boldsymbol{r}_j$ with respect to the link coordinate frame; and six second moments which represent the inertia of the link about the COM but with respect to axes aligned with the link frame $\{j\}$, see page 81. We can view the dynamic parameters of a robot's link by

```
>> p560.links(1).dyn
l =
 theta=q, d=0, a=0, alpha=1.571 (R,stdDH)
  m     = 0.000000
  r     = 0.000000 0.000000 0.000000
  I     = | 0.000000 0.000000 0.000000 |
          | 0.000000 0.350000 0.000000 |
          | 0.000000 0.000000 0.000000 |
  Jm    = 0.000200
  Bm    = 0.001480
  Tc    = 0.395000(+) -0.435000(-)
  G     = -62.611100
```

which in order are: the kinematic parameters, link mass, COM position, inertia matrix, motor inertia, motor friction, Coulomb friction and gear ratio.

The remainder of this section examines the various matrix components of Eq. 9.1.

9.1.1 Gravity Term

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

We start our detailed discussion with the gravity term because it is generally the dominant term in Eq. 9.1 and is present even when the robot is stationary or moving slowly. Some robots use counterbalance weights◀ or even springs to reduce the *gravity* torque that needs to be provided by the motors – this allows the motors to be smaller and thus lower in cost.

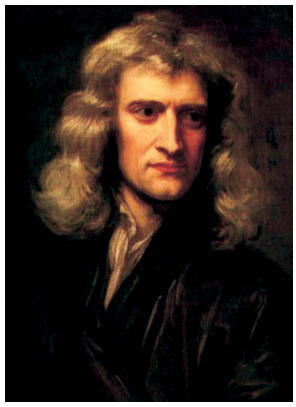
Counterbalancing will however increase the inertia associated with a joint since it adds additional mass at the end of a lever arm, and increase the overall mass of the robot.

In the previous section we used the `rne` method to compute the gravity load by setting the joint velocity and acceleration to zero. A more convenient approach is to use the `gravload` method

```
>> gravload = p560.gravload(qn)
gravload =
   -0.0000   31.6399    6.0351    0.0000    0.0283         0
```

Sir Isaac Newton (1642–1727) was an English mathematician and alchemist. He was Lucasian professor of mathematics at Cambridge, Master of the Royal Mint, and the thirteenth president of the Royal Society. His achievements include the three laws of motion, the mathematics of gravitational attraction, the motion of celestial objects and the theory of light and color (see page 224), and building the first reflecting telescope.

Many of these results were published in 1687 in his great 3-volume work "The Philosophiae Naturalis Principia Mathematica" (Mathematical principles of natural philosophy). In 1704 he published "Opticks" which was a study of the nature of light and colour and the phenomena of diffraction. The SI unit of force is named in his honour. He is buried in Westminster Abbey, London.



The `SerialLink` object contains a default gravitational acceleration vector which is inititialized to the nominal value for Earth

```
>> p560.gravity'
ans =
          0          0     9.8100
```

The `'gravity'` option for the `SerialLink` constructor can change this.

We could change gravity to the lunar value

```
>> p560.gravity = p560.gravity/6;
```

resulting in reduced joint torques

```
>> p560.gravload(qn)
ans =
    0.0000    5.2733    1.0059    0.0000    0.0047         0
```

or we could turn our lunar robot upside down

```
>> p560.base = trotx(pi);
>> p560.gravload(qn)
ans =
    0.0000   -5.2733   -1.0059   -0.0000   -0.0047         0
```

and see that the torques have changed sign. Before proceeding we bring our robot back to Earth and right-side up

```
>> mdl_puma560
```

The torque exerted on a joint due to gravity acting on the robot depends very strongly on the robot's pose. Intuitively the torque on the shoulder joint is much greater when the arm is stretched out horizontally

```
>> Q = p560.gravload(qs)
Q =
   -0.0000   46.0069    8.7722    0.0000    0.0283         0
```

than when the arm is pointing straight up

```
>> Q = p560.gravload(qr)
Q =
         0   -0.7752    0.2489         0         0         0
```

The gravity torque on the elbow is also very high in the first pose since it has to support the lower arm and the wrist. We can investigate how the gravity load on joints 2 and 3 varies with joint configuration by

```
1  [Q2,Q3] = meshgrid(-pi:0.1:pi, -pi:0.1:pi);
2  for i=1:numcols(Q2),
3      for j=1:numcols(Q3);
4      g = p560.gravload([0 Q2(i,j) Q3(i,j) 0 0 0]);
5      g2(i,j) = g(2);
6      g3(i,j) = g(3);
7      end
8  end
9  surfl(Q2, Q3, g2); surfl(Q2, Q3, g3);
```

and the results are shown in Fig. 9.1. The gravity torque on joint 2 varies between ± 40 N m and for joint 3 varies between ± 10 N m. This type of analysis is very important in robot design to determine the required torque capacity for the motors.

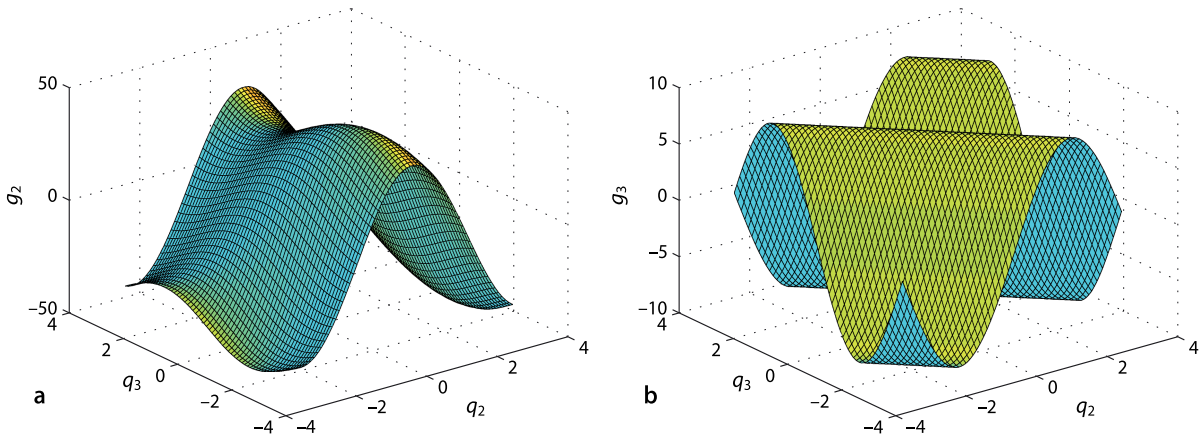


Fig. 9.1. Gravity load variation with manipulator pose. **a** Shoulder gravity load, $g_2(q_2, q_3)$; **b** elbow gravity load $g_3(q_2, q_3)$

9.1.2 Inertia Matrix

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

This inertia matrix includes the motor inertias, which are added to the diagonal elements. Motor and rigid-body inertia are discussed further in Sect. 9.4.2.

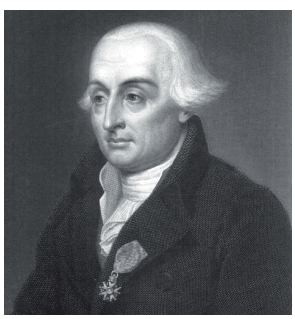
The inertia matrix◂ is a function of the manipulator pose

```
>> 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
```

which we observe 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.

We can investigate some of the elements of the inertia matrix and how they vary with robot configuration using the simple (but slow) commands

```
[Q2,Q3] = meshgrid(-pi:0.1:pi, -pi:0.1:pi);
for i=1:numcols(Q2),
    for j=1:numcols(Q3);
        M = p560.inertia([0 Q2(i,j) Q3(i,j) 0 0 0]);
        M11(i,j) = M(1,1);
        M12(i,j) = M(1,2);
    end
end
surfl(Q2, Q3, M11); surfl(Q2, Q3, M12);
```



Joseph-Louis Lagrange (1736–1813) was an Italian-born French mathematician and astronomer. He made significant contributions to the fields of analysis, number theory, classical and celestial mechanics. In 1766 he succeeded Euler as the director of mathematics at the Prussian Academy of Sciences in Berlin, where he stayed for over twenty years, producing a large body of work and winning several prizes of the French Academy of Sciences. His treatise on analytical mechanics "Mécanique Analytique" first published in 1788, offered the most comprehensive treatment of classical mechanics since Newton and formed a basis for the development of mathematical physics in the nineteenth century. In 1787 he became a member of the French Academy, became the first professor of analysis at the École Polytechnique, a member of the Legion of Honour and a Count of the Empire in 1808. He is buried in the Panthéon in Paris.

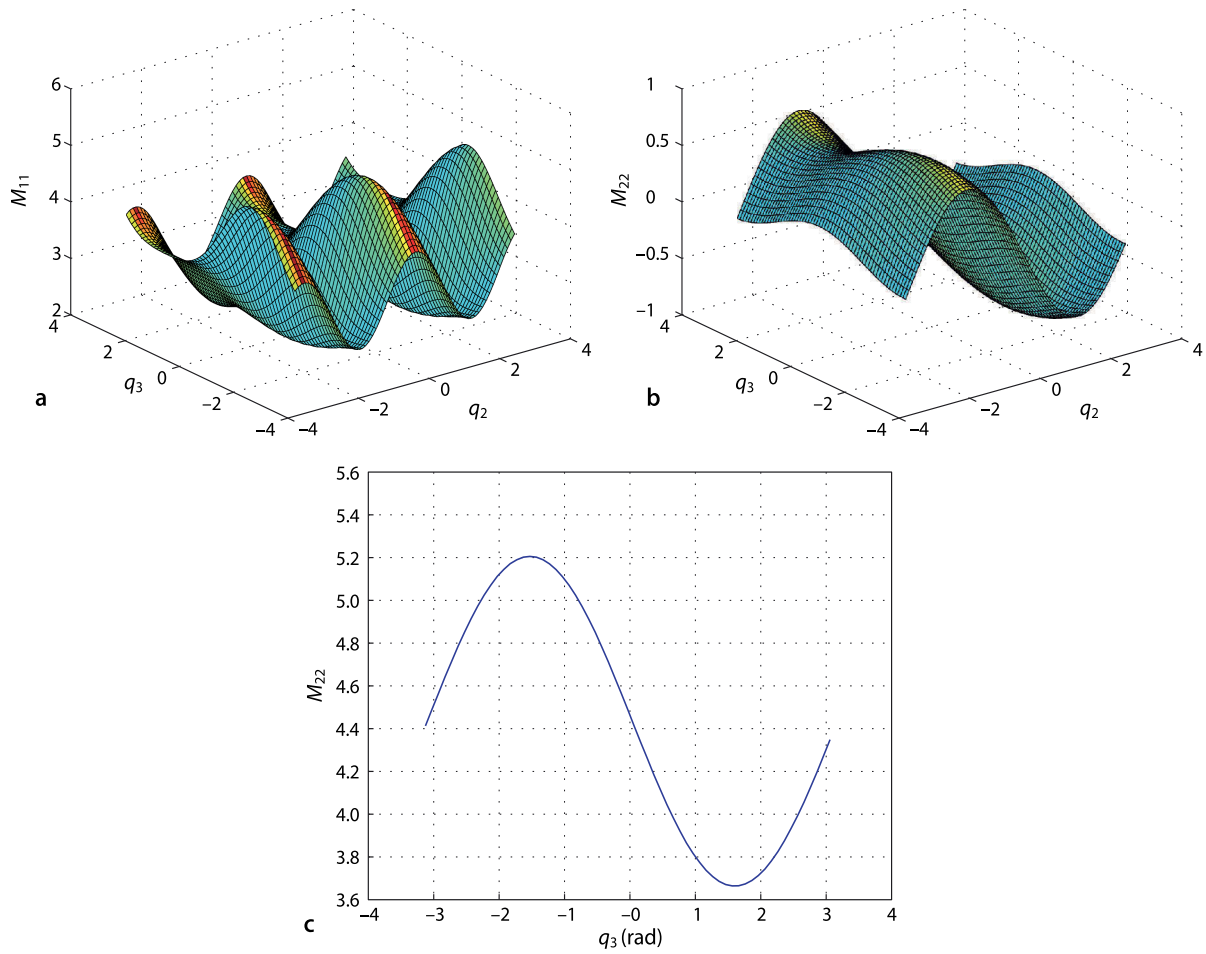


Fig. 9.2. Variation of inertia matrix elements as a function of manipulator pose. **a** Joint 1 inertia as a function of joint 2 and 3 angles $M_{11}(q_2, q_3)$; **b** product of inertia $M_{12}(q_2, q_3)$; **c** joint 2 inertia as a function of joint 3 angle $M_{22}(q_3)$. Inertia has the units of kg m^2

The results are shown in Fig. 9.2 and we see significant variation in the value of M_{11} which changes by a factor of

```
>> max(M11(:)) / min(M11(:))
ans =
    2.1558
```

This is important for robot design since, for a fixed maximum motor torque, inertia sets the upper bound on acceleration which in turn effects path following accuracy.

The off-diagonal term M_{12} represents coupling between the angular acceleration of joint 2 and the torque on joint 1. That is, if joint 2 accelerates then a torque will be exerted on joint 1 and vice versa.

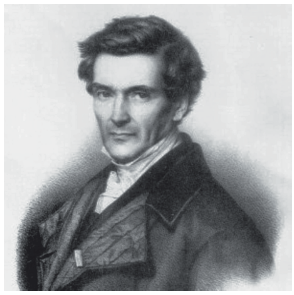
9.1.3 Coriolis Matrix

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

The Coriolis matrix $\boldsymbol{C}$ 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, at the nominal pose with all joints moving at 0.5 rad s^{-1}

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

the Coriolis matrix is



Gaspard-Gustave de Coriolis (1792–1843) was a French mathematician, mechanical engineer and scientist. Born in Paris, in 1816 he became a tutor at the École Polytechnique where he carried out experiments on friction and hydraulics and later became a professor at the École des Ponts and Chaussées (School of Bridges and Roads). He extended ideas about kinetic energy and work to rotating systems and in 1835 wrote the famous paper *Sur les équations du mouvement relatif des systëmes de corps* (On the equations of relative motion of a system of bodies) which dealt with the transfer of energy in rotating systems such as waterwheels. In the late 19th century his ideas were picked up by the meteorological community to incorporate effects due to the Earth's rotation. He is buried in Paris's Montparnasse Cemetery.

```
>> C = p560.coriolis(qn, qd)
C =
    0.0000   -0.9115    0.2173    0.0013   -0.0026    0.0001
    0.3140   -0.0000    0.5786   -0.0011   -0.0001   -0.0000
   -0.1804   -0.1929   -0.0000   -0.0005   -0.0023   -0.0000
   -0.0002    0.0006   -0.0000   -0.0000    0.0003   -0.0000
   -0.0000    0.0000    0.0014   -0.0002   -0.0000   -0.0000
         0    0.0000    0.0000    0.0000    0.0000         0
```

The off-diagonal terms $\boldsymbol{C}_{i,j}$ represent coupling of joint j velocity to the generalized force acting on joint i. $\boldsymbol{C}_{1,2} = -0.9115$ is very significant and represents coupling from joint 2 velocity to torque on joint 1 – 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 in this example are

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

9.1.4 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◂

This assumes that the last link itself has no mass which is a reasonable approximation.

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

The inertia at the nominal pose is now

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

and the *ratio* with respect to the unloaded case, computed earlier, is

```
>> 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
```

We see that the diagonal elements have increased significantly, for instance the elbow joint inertia has increased by 66% which reduces the maximum acceleration by nearly two thirds. Reduced acceleration impairs the robot's ability to accurately follow a high speed path. The inertia of joint 6 is unaffected since this added mass lies on the axis of this joint's rotation. 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 that are *felt* by all the other robot joints.

The gravity load has also increased by some significant factors

```
>> p560.gravload(qn) ./ gravload
ans =
    0.3737    1.5222    2.5416   18.7826   86.8056       NaN
```

particularly at the elbow and wrist.

Elements 1, 4 and 6 should be ignored and are the results of numerical error in what should be 0 / 0, for the case of gravity load with and without payload $g_1 = g_6 = 0$.

9.1.5 Base Force

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. This wrench is returned as an optional output argument of the `rne` method

```
>> [Q,g] = p560.rne(qn, qz, qz);
```

In this case `g` is the wrench

```
>> g'
ans =
         0   -0.0000  230.0445  -48.4024  -31.6399   -0.0000
```

that needs to be applied to the base to keep it in equilibrium. The vertical force of 230 N is the total weight of the robot which has a mass of

```
>> sum([p560.links.m])
ans =
   23.4500
```

There is also a moment about the x- and y-axes since the centre of mass of the robot is not over origin of the base coordinate frame.

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.

9.1.6 Dynamic Manipulability

In Sect. 8.1.4 we discussed a kinematic measure of manipulability, that is, how well configured the robot is to achieve velocity in particular directions. An extension of that measure is to consider how well the manipulator is able to accelerate in different Cartesian directions. Following a similar approach, we consider the set of generalized joint forces with unit norm

$$\boldsymbol{Q}^T\boldsymbol{Q} = 1$$

From Eq. 9.1 and ignoring gravity and assuming $\dot{\boldsymbol{q}} = 0$ we write

$$\boldsymbol{Q} = \boldsymbol{M}\ddot{\boldsymbol{q}}$$

Differentiating Eq. 8.2 and still assuming $\dot{\boldsymbol{q}} = 0$ we write

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

Combining these we write

$$\dot{\boldsymbol{\nu}}^T(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{M}^{-T}\boldsymbol{J}^T)^{-1}\dot{\boldsymbol{\nu}} = 1$$

or more compactly

$$\dot{\boldsymbol{\nu}}^T\boldsymbol{M}_x^{-1}\dot{\boldsymbol{\nu}} = 1$$

which is the equation of a hyper-ellipsoid in Cartesian acceleration space. For example, at the nominal pose

```
>> J = p560.jacob0(qn);
>> M = p560.inertia(qn);
>> Mx = (J * inv(M) * inv(M)' * J');
```

If we consider just the translational acceleration, that is the top left 3×3 submatrix of $\boldsymbol{M}_x$

```
>> Mx = Mx(1:3, 1:3);
```

this is a 3-dimensional ellipsoid

```
>> plot_ellipse( Mx )
```

which is plotted in Fig. 9.3. The major axis of this ellipsoid is the direction in which the manipulator has maximum acceleration at this configuration. The radii of the ellipse are the square root of the eigenvalues

```
>> sqrt(eig(Mx))
ans =
    0.4412
    0.1039
    0.1677
```

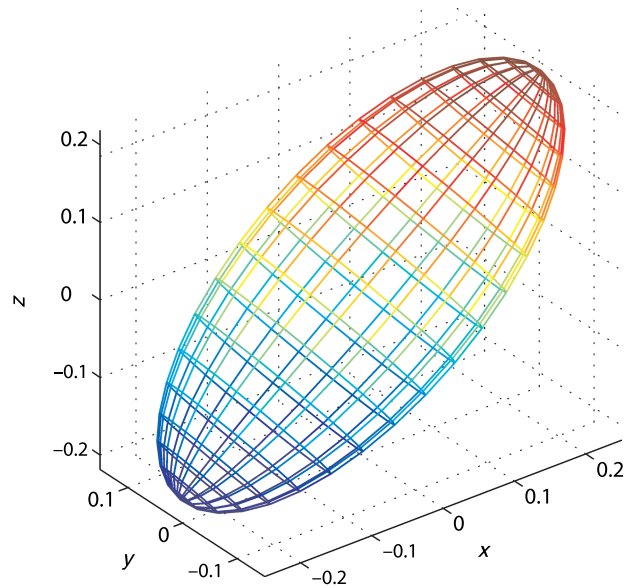


Fig. 9.3. Spatial acceleration ellipsoid for Puma 560 robot in nominal pose

and the direction of maximum acceleration is given by the first eigenvector. The ratio of the minimum to maximum radius

```
>> min(ans)/max(ans)
ans =
    0.2355
```

is a measure of the non-uniformity of end-effector acceleration.▶ It would be unity for isotropic acceleration capability. In this case acceleration capability is good in the x- and z-directions, but poor in the y-direction.

The 6-dimensional ellipsoid has dimensions with different units: m s^{-2} and rad s^{-2}. This makes comparison of all 6 radii problematic.

The manipulability measure proposed by Asada is similar but considers the ratios of the eigenvalues of

$$\ddot{\boldsymbol{x}}^T \boldsymbol{J}^{-T} \boldsymbol{M} \boldsymbol{J}^{-1} \ddot{\boldsymbol{x}} = 1$$

and returns a uniformity measxure $m \in [0, 1]$ where 1 indicates uniformity of acceleration in all directions. For this example

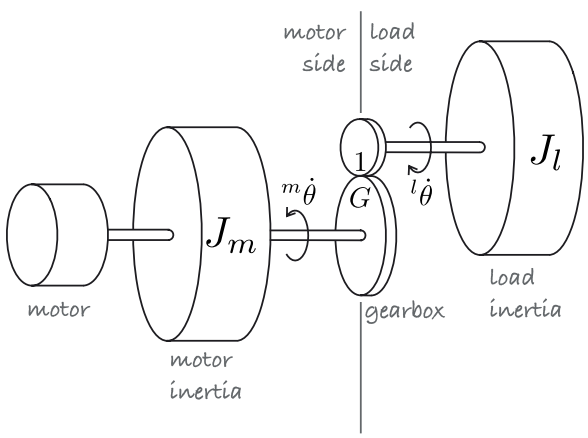
```
>> p560.maniplty(qn, 'asada')
ans =
    0.2094
```

9.2 Drive Train

The vast majority of robots today are driven by electric motors. Typically brushless servo motors are used for large industrial robots while small laboratory or hobby robots use brushed DC motors or stepper motors. Robots with very large payloads, hundreds of kilograms, would generally use electro-hydraulic actuators.

Electric motors are compact and efficient but do not produce very high torque. However they can rotate at very high speed so it is common to use a reduction gearbox to tradeoff speed for increased torque. The disadvantage of a gearbox is increased cost, weight, friction and mechanical noise. Many very high-performance robots, such as used in high-speed electronic assembly, use expensive high-torque motors with a direct drive or a very low gear ratio achieved using cables or thin metal bands rather than gears.

Figure 9.4 shows in schematic form the drive train of a typical robot joint. For a G:1 reduction drive the torque at the link is G times the torque at the motor. For rotary joints the quantities measured at the link, reference frame l, are related to the motor referenced quantities, reference frame m, as shown in the table in Fig. 9.4. For example if you turned the motor shaft by hand you would *feel* the inertia of the load through the gearbox but it would be reduced by G^2 as would the frictional force. However if you turned the load side shaft by hand you would *feel* the inertia of the motor through the gearbox but it would be increased by G^2, as would the frictional force.



$$\begin{aligned}
{}^lJ &= G^2\,{}^mJ \\
{}^lB &= G^2\,{}^mB \\
{}^l\tau_C &= G\,{}^m\tau_C \\
{}^l\tau &= G\,{}^m\tau \\
{}^l\dot{\theta} &= {}^m\dot{\theta}/G \\
{}^l\ddot{\theta} &= {}^m\ddot{\theta}/G
\end{aligned}$$

Fig. 9.4.
Schematic of a typical robot drivetrain showing motor and load inertia, gearbox and disturbance torque. The frame of reference, motor or load, is indicated by the leading superscript. The table at the right shows the relationship between load and motor referenced quantities for gear ratio G

There are two components of inertia *seen* by the motor. The first is due to the rotating part of the motor itself, its rotor. It is denoted J_m and is an intrinsic characteristic of the motor and the value is provided in the motor manufacturer's data sheet. The second component is the load inertia J_l which is the inertia of the driven link and all the other links that are attached to it. For joint j this is element M_{jj} of the manipulator inertia matrix discussed previously and is a function of the robot's configuration.

The total inertia *seen* by the motor for joint j is therefore

$$^{m}J_j = J_{m_j} + \frac{1}{G_j^2} M_{jj} \tag{9.3}$$

The gearing reduces the significance of the second term which is configuration dependent. Reducing the variability in total inertia is beneficial when it comes to designing a control system for the joint.

A negative aspect of gearing is an increase in viscous friction and non-linear effects such as backlash and Coulomb friction which will be discussed in the next section. Flexible couplings and long drive shafts between the motor and the joint can act as torsional springs and introduce complex dynamics into the system and this is discussed in Sect. 9.4.4.

9.2.1 Friction

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

For most electric drive robots friction is the next most dominant joint force after gravity.◂ For any rotating machinery, motor or gearbox, the friction torque versus speed characteristic has a form similar to that shown in Fig. 9.5. 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*.

For the Puma robot joint friction varied from 10 to 47% of the maximum motor torque for the first three joints (Corke 1996b).

Once the machine is moving the stiction force rapidly decreases and viscous friction dominates. Viscous friction, shown by the dashed line in Fig. 9.5, is commonly modeled by

$$Q_f = B\dot{q} + Q_C \tag{9.4}$$

where the slope B is the viscous friction coefficient and the offset is Coulomb friction. The latter is frequently modeled by the non-linear function

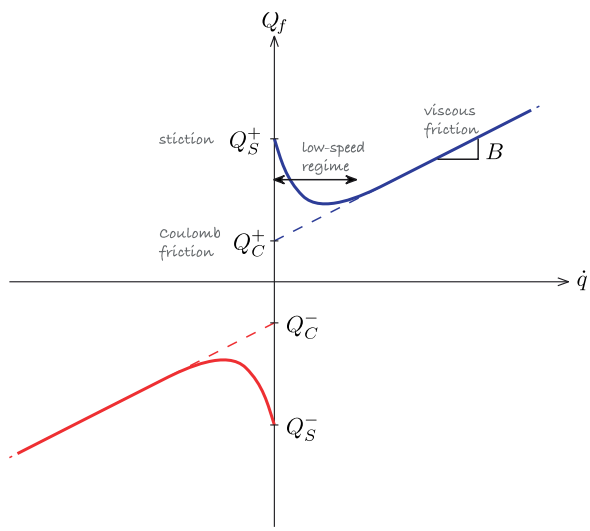


Fig. 9.5. Typical friction versus speed characteristic. The dashed lines depict a simple piecewise-linear friction model characterized by slope (viscous friction) and intercept (Coulomb friction)

Charles-Augustin de Coulomb (1736–1806) was a French physicist. He was born in Angoulême to a wealthy family and studied mathematics at the Collége des Quatre-Nations under Pierre Charles Monnier, and later at the military school in Méziéres. He spent eight years in Martinique involved in the construction of Fort Bourbon and there he contracted tropical fever.

Later he worked at the shipyards in Rochefort which he used as laboratories for his experiments in static and dynamic friction of sliding surfaces. His paper *Théorie des machines simples* won the Grand Prix from the Académie des Sciences in 1781. His later research was on electromagnetism and electrostatics and he is best known for the formula on electrostatic forces, named in his honor, as is the SI unit of charge. After the revolution he was involved in determining the new system of weights and measures.

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

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

There are several components of the friction *seen* by the motor. The first component is due to the motor itself: its bearings and, for a brushed motor, the brushes rubbing on the commutator. The viscous friction coefficient for a motor is constant and often provided in the manufacturer's data sheet. Information about Coulomb friction is not generally provided. Other components of friction are due to the gearbox and the bearings that support the link.

The Toolbox models friction within the `Link` object. The friction values are lumped and motor referenced, that is, they apply to the motor side of the gearbox. Viscous friction is a scalar that applies for positive and negative velocity. Coulomb friction is a 2-vector comprising (Q_C^+, Q_C^-). For example, the dynamic parameters of the Puma robot's second link are

In practice some mechanisms have a velocity dependent friction characteristic.

```
>> p560.links(2).dyn
l =
 theta=q, d=0, a=0.4318, alpha=0 (R,stdDH)
  m     = 17.400000
  r     = -0.363800 0.006000 0.227500
  I     = | 0.130000 0.000000 0.000000 |
          | 0.000000 0.524000 0.000000 |
          | 0.000000 0.000000 0.539000 |
  Jm    = 0.000200
  Bm    = 0.000817
  Tc    = 0.126000(+) -0.071000(-)
  G     = 107.815000
```

The last three lines show the viscous friction coefficient, Coulomb friction coefficients and the gear ratio. The online documentation for the `Link` class describes how to set these parameters.

9.3 Forward Dynamics

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 the equations of motion Eq. 9.1 we obtain the joint acceleration

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\big(\boldsymbol{Q} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{F}(\dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q})\big) \tag{9.6}$$

This is computed by the `accel` method of the `SerialLink` class

```
>> qdd = p560.accel(q, qd, Q)
```

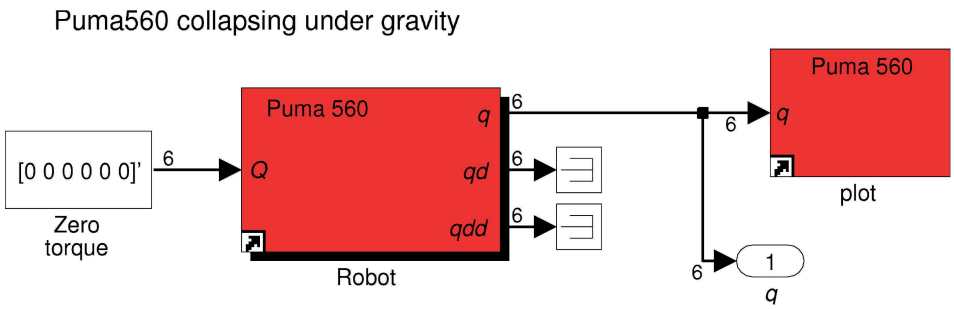


Fig. 9.6.
Simulink® model `sl_ztorque` for the Puma 560 manipulator with zero joint torques

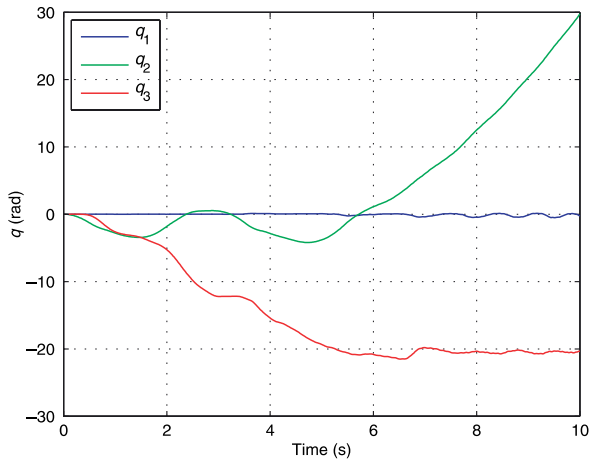


Fig. 9.7.
Joint angle trajectory for Puma 560 robot collapsing under gravity and starting at `qz`

given the joint coordinates, joint velocity and applied joint torques. This functionality is also encapsulated in the Simulink® block `Robot` and an example of its use is

```
>> sl_ztorque
```

which is shown in Fig. 9.6. The torque applied to the robot is zero and the initial joint angles is set as a parameter of the `Robot` block, in this case to the *zero-angle pose*. The simulation is run

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

and the joint angles as a function of time are returned in the object `r`

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

We can show the robot's motion in animation

```
>> p560.plot(q)
```

and see it collapsing under gravity since there are no torques to counter gravity and hold in upright. The shoulder falls and swings back and forth as does the elbow,◂ while the waist joint rotates because of Coriolis coupling. The motion will slowly decay as the energy is dissipated by viscous friction.

These motions are not mechanically possible on the real robot.

Alternatively we can plot the joint angles as a function of time

```
>> plot(t, q(:,1:3))
```

and this is shown in Fig. 9.7. The method `fdyn` can be used as a non-graphical alternative to Simulink® and is described in the online documentation.

This example is rather unrealistic and in reality the joint torques would be computed as some function of the actual and desired robot joint angles. This is the topic of the next section.

Coulomb friction is a strong non-linearity and can cause difficulty when using numerical integration routines to solve the forward dynamics. This is usually manifested by very long integration times. Fixed-step solvers tend to be more tolerant, and these can be selected through the Simulink® `Simulation+Configuration Parameters` menu item.

The default Puma 560 model, defined using `mdl_puma560`, has non-zero viscous and Coulomb friction parameters for each joint. Sometimes it is useful to zero all the friction parameters for a robot and this can be achieved by

```
>> p560_nf = p560.nofriction();
```

which returns a copy of the robot object that is similar in all respects except that the Coulomb friction is zero. Alternatively we can set Coulomb *and* viscous friction coefficients to zero

```
>> p560_nf = p560.nofriction('all');
```

9.4 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. In this section we discuss the two main approaches to robot joint control: independent control and model-based control.

9.4.1 Actuators

Most laboratory and industrial robots are electrically actuated. Electrical motors can be either current or voltage controlled and we consider here the current control case.▸ We assume a motor driver or amplifier which provides motor current

$$i = K_a u$$

that is linearly related to the applied control voltage u and where K_a is the transconductance of the amplifier with units of $\mathrm{A\,V^{-1}}$. The torque generated by the motor is proportional to current

$$\tau = K_m i$$

where K_m is the motor torque constant with units of $\mathrm{N\,m\,A^{-1}}$. The dynamics of the motor are described by

$$J_m \dot{\omega} + B\omega + \tau_c(\omega) = K_m K_a u \tag{9.7}$$

where J_m is the total inertia seen by the motor from Eq. 3.9, B is the viscous friction coefficient and τ_c is the Coulomb friction torque.

▸ Current control is implemented by an electronic constant current source, or a variable voltage source with feedback of actual motor current. In the latter case the electrical dynamics of the motor due to its resistance and inductance must be taken into account. A variable voltage source is most commonly implemented by a pulse-width modulated (PWM) switching circuit.

9.4.2 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 as we shall see, this is complicated by various *disturbance* torques such as gravity, velocity and acceleration coupling and friction that act on the joint.

Table 9.1. Motor and drive parameters for Puma 560 shoulder joint (Corke 1996b)

Parameter	Symbol	Value	Unit
Motor torque constant	K_m	0.228	N m A^{-1}
Motor inertia	J_m	200×10^{-6}	kg m^2
Drive viscous friction	B_m	817×10^{-6}	N m s rad^{-1}
Drive Coulomb friction	τ_C^+ τ_C^-	0.126 −0.709	N m N m
Gear ratio	G	107.815	
Maximum torque	τ_{max}	0.900	N m
Maximum speed	$\dot{q}_{max}$	165	rad s^{-1}

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.

Velocity loop. We will study the inner velocity loop first and we will use as an example the shoulder joint of the Puma 560 robot since its parameters are well known, see Table 9.1.

Ignoring Coulomb friction we write the Laplace transform of Eq. 9.7 as

$$sJ\Omega(s) + B\Omega(s) = K_m K_a U(s)$$

where $\Omega(s)$ and $U(s)$ are the Laplace transform of the time domain signals $\omega(t)$ and $u(t)$ respectively. Rearranging as a linear transfer function we write

$$\frac{\Omega(s)}{U(s)} = \frac{K_m K_a}{Js + B}$$

The effective inertia from Eq. 9.3 is

$$J = J_m + \frac{1}{G^2} M_{22} \tag{9.8}$$

Figure 9.2 shows that M_{22} varies significantly with manipulator pose so for now we will take the mean value which is 2 kg m^2 which yields a total inertia of

$$J = 200 \times 10^{-6} + \frac{2}{(107.815)^2} = 200 \times 10^{-6} + 172 \times 10^{-6} = 372 \times 10^{-6} \text{ kg m}^2$$

Note that the refered link inertia is comparable to the inertia of the motor itself.

The Simulink® model is shown in Fig. 9.8. A delay of 1 ms is included to model the computational time of the velocity loop control algorithm and a saturator models the finite maximum torque that the motor that can deliver. We use a proportional controller based on the error between demanded and actual velocity to compute the demand to the motor driver

$$u^* = K_v(\dot{q}^* - \dot{q}) \tag{9.9}$$

The motor velocity is typically computed by taking the difference in motor position at each sample time, and the position is measured by a shaft encoder.

To test this velocity controller we create a test harness

```
>> vloop_test
```

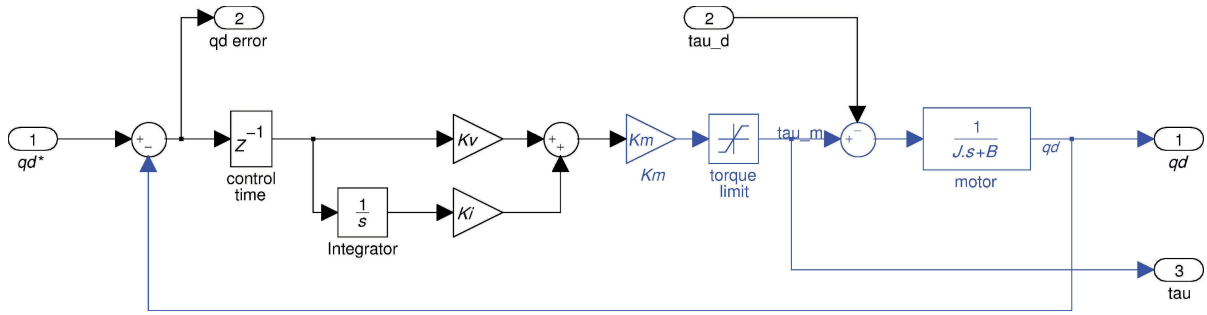


Fig. 9.8. Velocity control loop, Simulink® model `vloop`

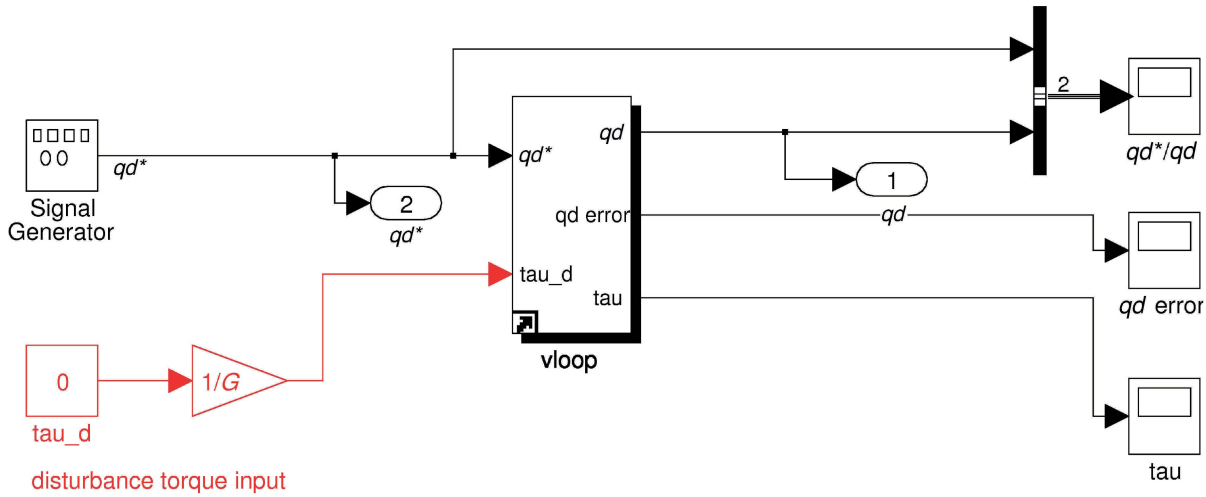


Fig. 9.9. Test harness for the velocity control loop, Simulink® model `vloop_test`. The input `tau_d` is used to simulate a disturbance torque acting on the joint

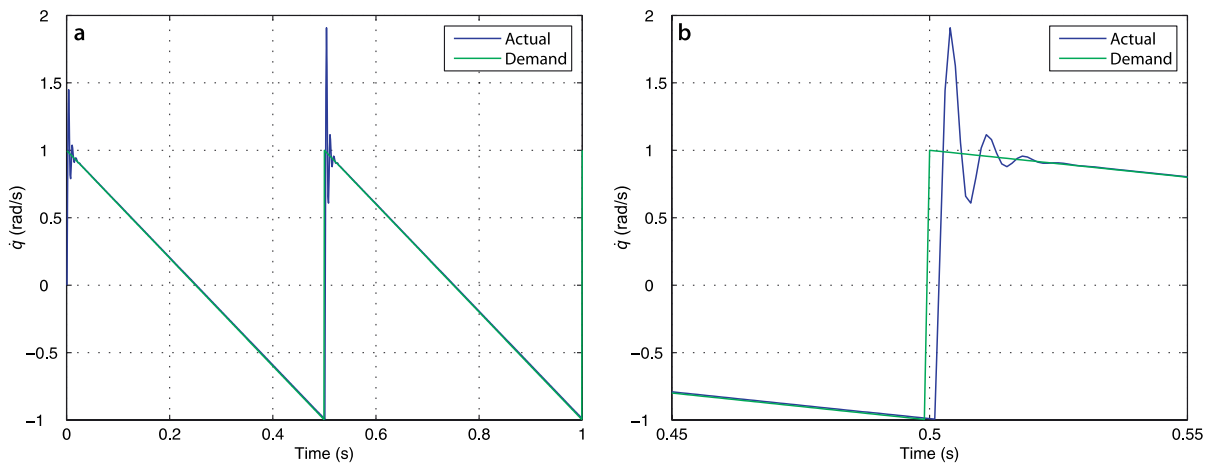


Fig. 9.10. Velocity loop with a sawtooth demand. **a** Response; **b** closeup of response

with a sawtooth-shaped velocity demand which is shown in Fig. 9.9. Running the simulator

```
>> sim('vloop_test');
```

While a step response is a common and useful measure of control perforamance, in practice a velocity loop would never receive a step demand.

and with a little experimentation we find that a gain of $K_v = 1$ gives satisfactory performance as shown in Fig. 9.10. There is some overshoot at the step◀ but less gain leads to increased velocity error and more gain leads to oscillation – as always in control engineering it is a matter of tradeoffs.

So far we have ignored one very important dynamic effect on robot arms – gravity. Figure 9.1b shows that the gravity torque on this joint varies from approximately −40 to 40 N m. We now add a disturbance torque equal to just half that amount, 20 N m at the load. We edit the test harness and rerun the simulation. The results shown in Fig. 9.11 indicate that the control performance has been badly degraded – the tracking error has increased to more than 1 rad s^{-1}.

There are three common strategies to counter this error. The simplest is to increase the gain. This will reduce the tracking error but push the system into instability.

The second strategy, commonly used in industrial motor drives, is to add integral action. We change Eq. 9.9 to a proportional-integral controller

$$u^* = \left(K_v + \frac{K_i}{s}\right)(\dot{q}^* - \dot{q}), \quad K_i > 0$$

In the Simulink® model of Fig. 9.8 this is achieved by setting `Ki` to a non-zero value. With some experimentation we find the gains $K_v = 1$ and $K_i = 10$ work well and the performance is shown in Fig. 9.12. The integrator state evolves over time to cancel out the disturbance term and we can see the error decaying to zero. In practice the disturbance varies over time and the integrator's ability to track it depends on the value of the integral gain K_i. In reality other disturbances affect the joint, for instance Coulomb friction and torques due to velocity and acceleration coupling. The controller needs to be well tuned so that these have minimal effect on the tracking performance.

From a classical control system perspective the original velocity loop contains no integrator block which means it is classified as a Type 0 system. A characteristic of such systems is they exhibit a finite error for a constant input or constant disturbance input, just as we observed in Fig. 9.11. Adding the integral controller changed the system to Type 1 which has zero error for a constant input or constant disturbance. As always in engineering there are some tradeoffs. The integral term can lead to increased overshoot so increasing K_i usually requires some compensating reduction of K_v. If the joint actuator is pushed to its performance limit, for instance the torque limit is reached, then the tracking error will grow with time since the motor acceleration will be lower than required. The integral of this increasing error will grow leading to a condition known as integral windup. When the joint finally reaches its destination the large accumulated integral keeps driving the motor forward until the integral decays – leading to large overshoot. Various strategies are employed to combat this, such as limiting the maximum value of the integrator, or only allowing integral action when the motor is close to its setpoint.

Motor limits. Electric motors are limited in both torque and speed. The maximum torque is defined by the maximum current the drive electronics can provide. A motor also has a maximum rated current beyond which the motor can be damaged by overheating or demagnetization of its permanent magnets which irreversibly reduces its torque constant. As speed increases so does friction and the maximum speed is $\omega_{max} = \tau_{max}/B$.

The product of motor torque and speed is the mechanical output power and also has an upper bound. Motors can tolerate some overloading, peak power and peak torque, for short periods of time but the sustained rating is significantly lower than the peak.

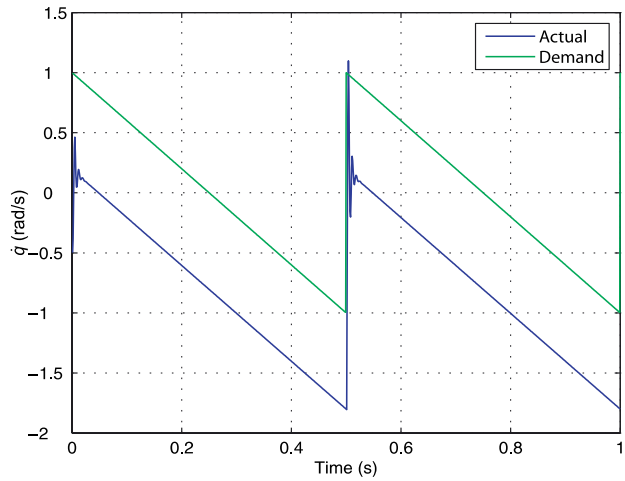


Fig. 9.11.
Velocity loop response to a saw-tooth demand with a gravity disturbance of 20 N m

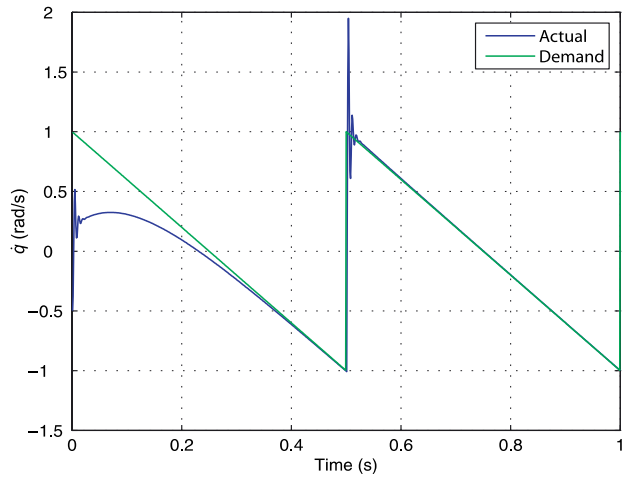


Fig. 9.12.
Velocity loop response to a saw-tooth demand with a gravity disturbance of 20 N m and proportional-integral control

Strategies one and two are collectively referred to as disturbance rejection and are concerned with reducing the effect of an unknown disturbance. However if we think about the problem in its robotics context the gravity disturbance is not unknown. In Sect. 9.1.1 we showed how to compute the torque due to gravity that acts on each joint. If we know this torque, and the motor torque constant, we can *add* it to the output of the PI controller. The third strategy to reduce the effect of disturbance is therefore to predict it and cancel it out – a strategy known as torque feedforward control. The block diagram of Fig. 9.8 is augmented with a feedforward term which is shown by the red wiring in Fig. 9.13.

Even if the gravity load is known imprecisely feedforward can reduce the magnitude of the disturbance.

The final consideration in control design is how inertia variation affects the closed-loop response. Using Eq. 9.8 and the data from Fig. 9.2c we find that the minimum and maximum joint inertia are 320×10^{-6} and 450×10^{-6} kg m^2 respectively. Figure 9.14 shows the velocity tracking error using the control gains chosen above for the case of minimum and maximum inertia. We can see that the tracking error decays more slowly for larger inertia, and is showing signs of instability for the minimum inertia case. In practice the gain would be chosen to optimize the closed-loop performance at both extremes.

Motor speed control is important for all types of robots, not just arms. For example it is used to control the speed of the wheels for car-like vehicles and and the rotors of a quad-rotor as discussed in Chap. 4.

Back EMF. A spinning motor acts like a generator and produces a voltage V_b called the back EMF which opposes the current flowing into the motor. Back EMF is proportional to motor speed $V_b = K_m\omega$ where K_m is the motor torque constant again whose units can also be interpreted as V s rad^{-1}. When this voltage equals the maximum possible voltage the drive electronics can provide then no more current can flow into the motor and torque falls to zero. This provides a practical upper bound on motor speed.

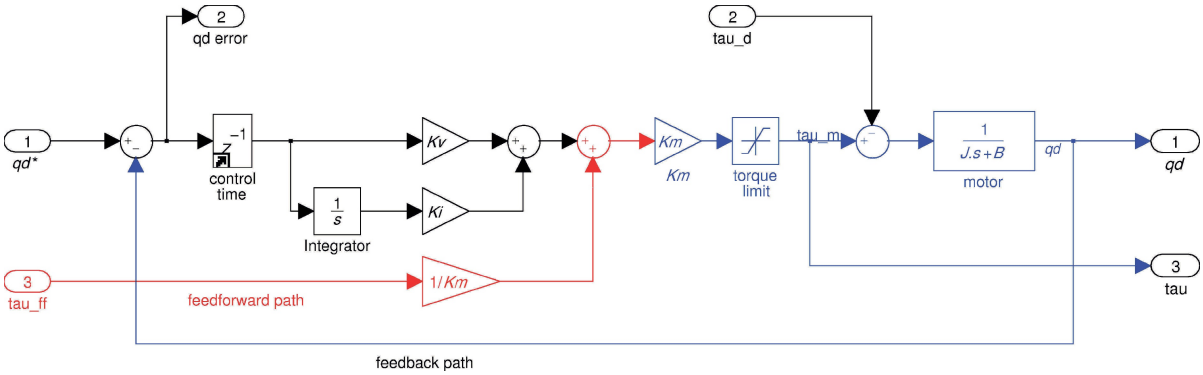


Fig. 9.13. Velocity control loop with feedforward (shown in red), from Simulink® model `vloop_test2`

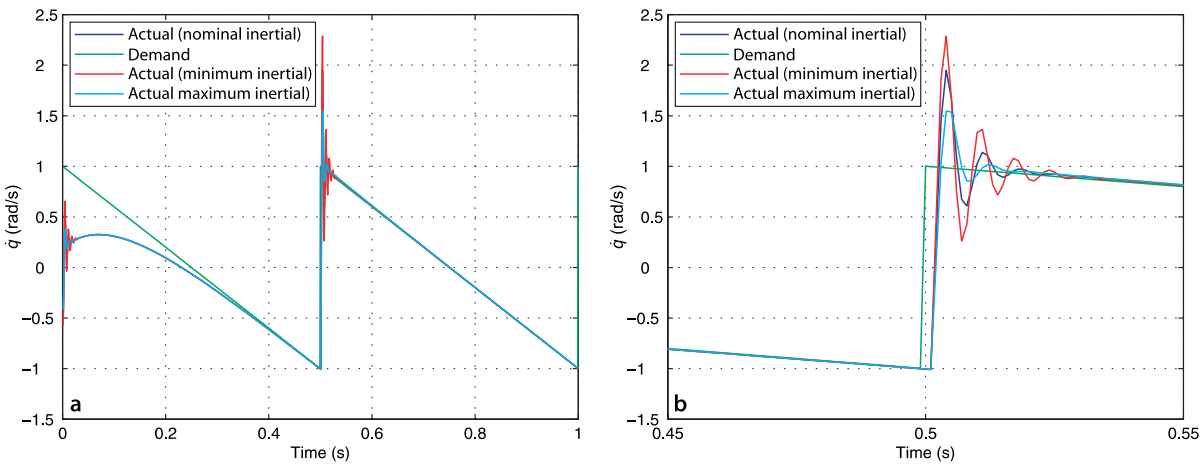


Fig. 9.14. Velocity loop with a sawtooth demand with integral action and 20 N m torque disturbance, but varying inertia M_{22}. **a** Response; **b** closeup of response

Position loop. The outer loop is responsible for maintaining position and its Simulink® model is shown in Fig. 9.15. The error in position provides the velocity demand for the inner loop.

To test the position control loop we create another test harness

```
>> ploop_test
```

The position demand comes from an LSPB trajectory generator that moves from 0 to 0.5 rad in 1 s with a sample rate of 1 000 Hz. The test harness shown in Fig. 9.16 can also inject a disturbance torque into the velocity loop.

We use a proportional controller based on the error between actual and demanded position to compute the desired speed of the motor

$$\dot{q}^* = K_p(q^* - q) \tag{9.10}$$

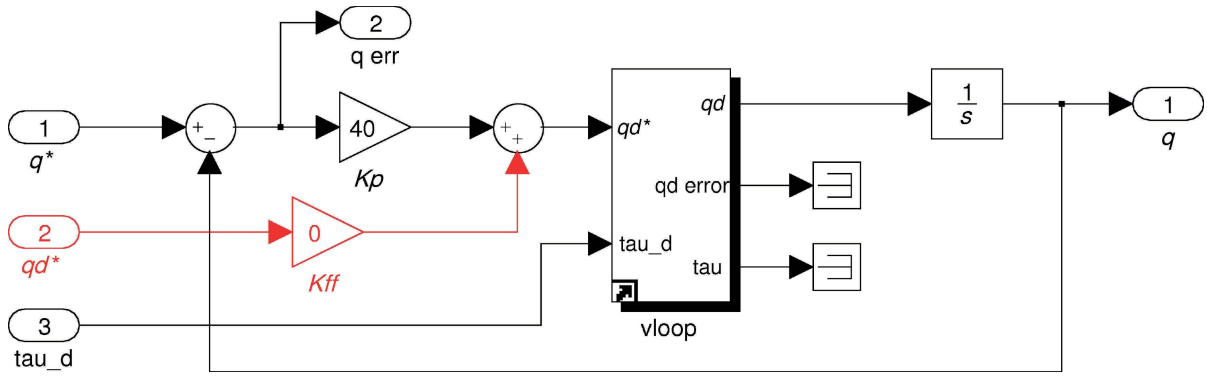


Fig. 9.15. Position control loop, Simulink® model `ploop`

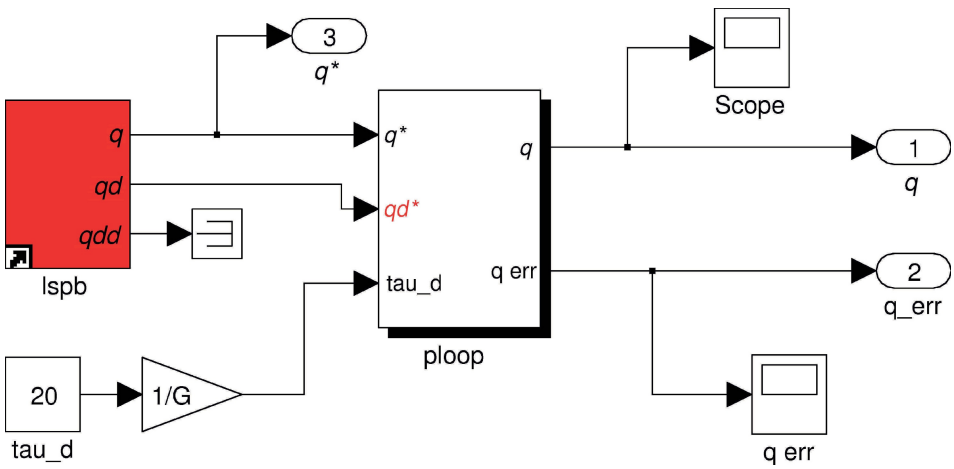


Fig. 9.16. Test harness for position control loop, Simulink® model `ploop_test`

The joint controller is tuned by adjusting the three gains: K_p, K_v, K_i in order to achieve good tracking performance at both low and high speed. For $K_p = 40$ the tracking and error responses are shown in Fig. 9.17a. The error between the demand and actual curves is due to the integral of the error in the velocity loop which has units of angle.

The position loop is based on feedback of error which is of course a classical approach. An often overlooked characteristic of proportional control is that zero error means zero demand to the controlled system. In this case zero position error means zero demanded velocity to the inner loop – to achieve non-zero joint velocity demand from Eq. 9.10 *requires* non-zero error which is of course not desirable for accurate tracking. Usefully the LSPB trajectory function computes velocity as a function of time as well as position. If we know the velocity we can add it to the output of the proportional control loop, the input to the velocity loop – a strategy known as velocity feedforward control. The time response with feedforward is shown in Fig. 9.17b and we see that tracking error is greatly reduced.

Let us recap what we have learnt about independent joint control. A common structure is the nested control loop. The inner loop uses a proportional or proportional-integral control law to generate a torque so that the actual velocity closely follows the velocity demand. The outer loop uses a proportional control law to generate the velocity demand so that the actual position closely follows the position demand. 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. Gearing reduces the magnitude of disturbance torques by $1 / G$ and the variation in inertia and friction by $1 / G^2$ but at the expense of cost, weight, increased friction and mechanical noise.

The velocity loop performance can be improved by adding an integral control term, or by feedforward of the disturbance torque which is largely predictable. The position loop performance can also be improved by feedforward of the desired joint velocity. In

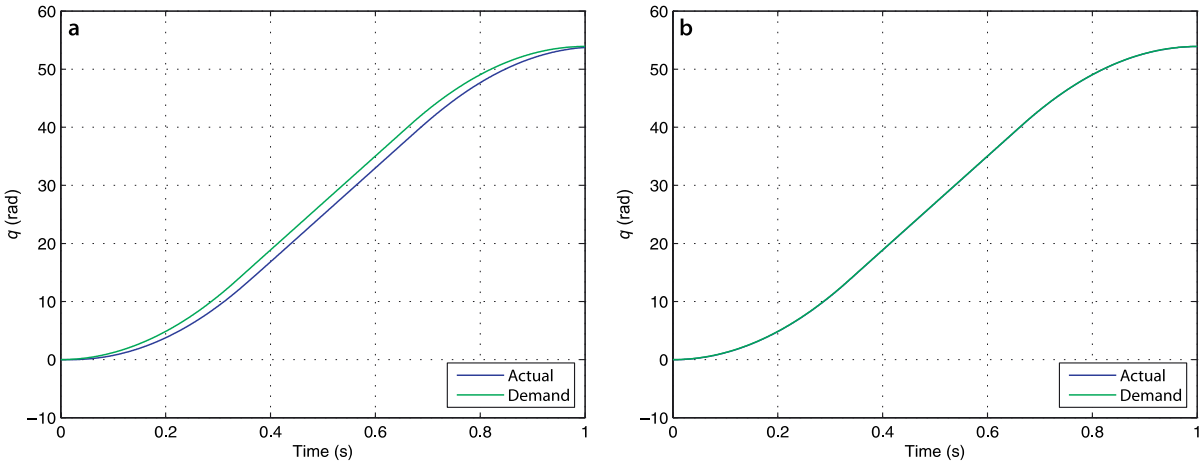


Fig. 9.17. Position loop following an LSPB trajectory. **a** Proportional control only **b** proportional control plus velocity demand feedforward

practice control systems use both feedforward and feedback control. Feedforward is used to inject signals that we can compute, in this case the joint velocity, and in the earlier case the gravity torque. Feedback control compensates for all remaining sources of error including variation in inertia due to manipulator pose and payload, changes in friction with time and temperature, and all the disturbance torques due to velocity and acceleration coupling. In general the use of feedforward allows the feedback gain to be reduced since a large part of the demand signal now comes from the feedforward.

9.4.3 Rigid-Body Dynamics Compensation

The previous section showed the limitations of independent joint control and introduced the concept of feedforward to compensate for the gravity disturbance torque. Inertia variation and other dynamic coupling forces were not explicitly dealt with and were left for the feedback controller to handle. However inertia and coupling torques can be computed according to Eq. 9.1 given knowledge of joint angles, joint velocities and accelerations, and the inertial parameters of the links. We can incorporate these torques into the control law using one of two *model-based* approaches: feedforward control, and computed torque control. The structural differences are contrasted in Fig. 9.18.

9.4.3.1 Feedforward Control

The torque feedforward controller shown in Fig. 9.18a is given by

$$\begin{aligned} \boldsymbol{Q}^* &= \underbrace{\boldsymbol{M}(\boldsymbol{q}^*)\ddot{\boldsymbol{q}}^* + \boldsymbol{C}(\boldsymbol{q}^*, \dot{\boldsymbol{q}}^*)\dot{\boldsymbol{q}}^* + \boldsymbol{F}(\dot{\boldsymbol{q}}^*) + \boldsymbol{G}(\boldsymbol{q}^*)}_{\text{feedforward}} + \underbrace{\left\{\boldsymbol{K}_v(\dot{\boldsymbol{q}}^* - \dot{\boldsymbol{q}}) + \boldsymbol{K}_p(\boldsymbol{q}^* - \boldsymbol{q})\right\}}_{\text{feedback}} \\ &= \mathcal{D}(\boldsymbol{q}^*, \dot{\boldsymbol{q}}^*, \ddot{\boldsymbol{q}}^*) + \left\{\boldsymbol{K}_v(\dot{\boldsymbol{q}}^* - \dot{\boldsymbol{q}}) + \boldsymbol{K}_p(\boldsymbol{q}^* - \boldsymbol{q})\right\} \end{aligned}$$

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

We can also consider that the feedforward term linearizes the non-linear dynamics about the operating point $(\boldsymbol{q}^*, \dot{\boldsymbol{q}}^*, \ddot{\boldsymbol{q}}^*)$. If the linearization is ideal then the dynamics of the error $\boldsymbol{e} = \boldsymbol{q}^* - \boldsymbol{q}$ are given by

$$\boldsymbol{M}(\boldsymbol{q}^*)\ddot{\boldsymbol{e}} + \boldsymbol{K}_v\dot{\boldsymbol{e}} + \boldsymbol{K}_p\boldsymbol{e} = 0 \tag{9.11}$$

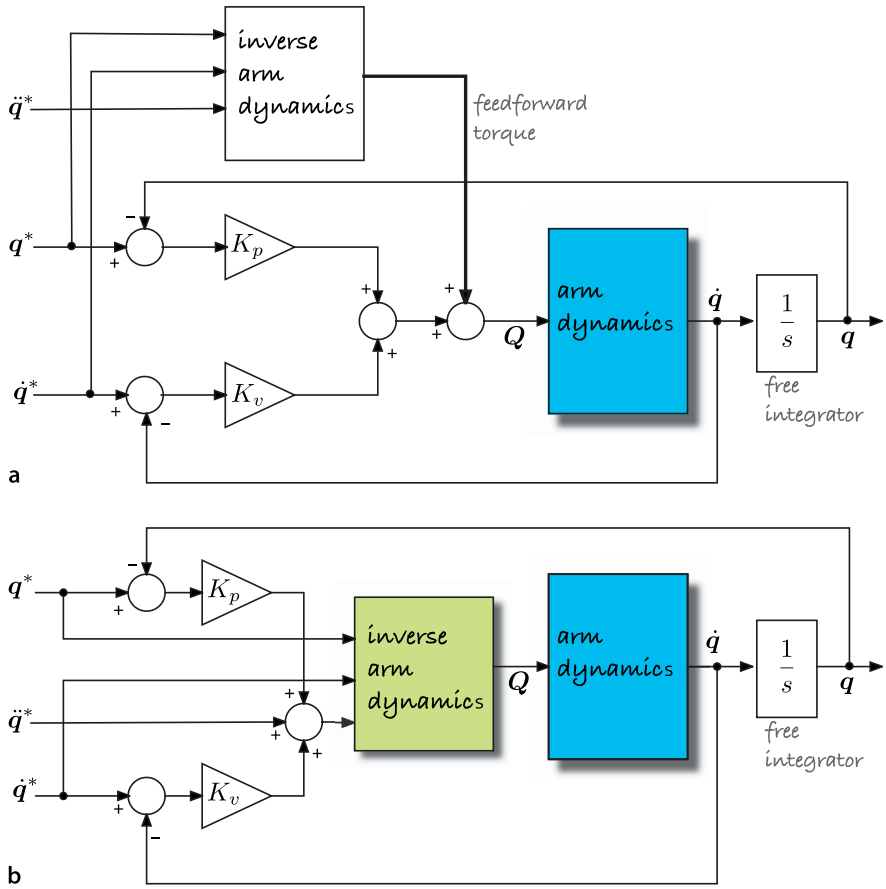


Fig. 9.18.
Manipulator control structures. **a** Feedforward control, **b** computed torque control

For well chosen K_p and K_v the error will decay to zero but the joint errors are coupled and their dynamics are dependent on the manipulator configuration. ▸ Due to the non-diagonal matrix M.

To test this controller using Simulink® we first create a `SerialLink` object

```
>> mdl_puma560
```

and then load the torque feedforward controller model

```
>> sl_fforward
```

which is shown in Fig. 9.19. The feedforward torque is computed using the `RNE` block and added to the feedback torque computed from position and velocity error. The desired joint angles and velocity are generated using a `jtraj` block.

Since the robot configuration changes relatively slowly the feedforward torque can be evaluated at a lower rate, T_{ff}, than the error feedback loops, T_{fb}. This is demonstrated in Fig. 9.19 by the zero-order hold block sampling at the relatively low sample rate of 20 Hz.

9.4.3.2 Computed Torque Control

The computed torque controller is shown in Fig. 9.18b. It belongs to a class of controllers known as inverse dynamic control. The principle is that the non-linear system is cascaded with its inverse so that the overall system has a constant unity gain. In practice the inverse is not perfect so a feedback loop is required to deal with errors.

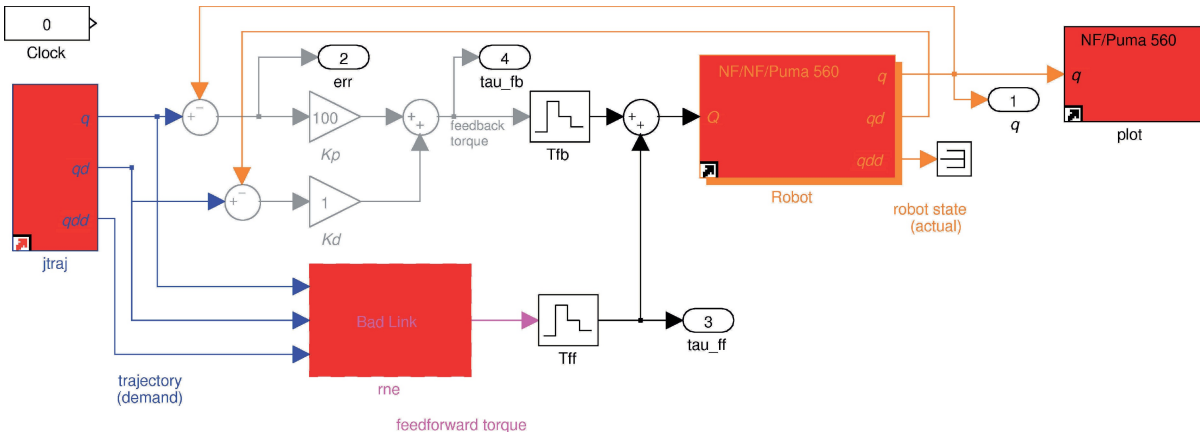


Fig. 9.19. The Simulink® model `sl_fforward` for Puma 560 with torque feedforward control. The blocks with the staircase icons are zero-order holds

The computed torque control is given by

$$\begin{aligned}\boldsymbol{Q} &= \boldsymbol{M}(\boldsymbol{q})\left\{\ddot{\boldsymbol{q}}^* + \boldsymbol{K}_v(\dot{\boldsymbol{q}}^* - \dot{\boldsymbol{q}}) + \boldsymbol{K}_p(\boldsymbol{q}^* - \boldsymbol{q})\right\} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) \\ &= \mathcal{D}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \left(\ddot{\boldsymbol{q}}^* + \boldsymbol{K}_v(\dot{\boldsymbol{q}}^* - \dot{\boldsymbol{q}}) + \boldsymbol{K}_p(\boldsymbol{q}^* - \boldsymbol{q})\right)\right)\end{aligned}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_v$ are the position and velocity gain (or damping) matrices respectively, and $\mathcal{D}(\cdot)$ is the inverse dynamics function.

In this case the inverse dynamics must be evaluated at each servo interval, although the coefficient matrices $\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{G}$ could be evaluated at a lower rate since the robot configuration changes relatively slowly. Assuming ideal modelling and parameterization the error dynamics of the system are

$$\ddot{\boldsymbol{e}} + \boldsymbol{K}_v\dot{\boldsymbol{e}} + \boldsymbol{K}_p\boldsymbol{e} = 0 \tag{9.12}$$

where $\boldsymbol{e} = \boldsymbol{q}^* - \boldsymbol{q}$. Unlike the torque feedforward controller the joint errors are uncoupled and their dynamics are therefore independent of manipulator configuration. In the case of model error there will be some coupling between axes, and the right-hand side of Eq. 9.12 will be a non-zero forcing function.

Using Simulink® we first create a `SerialLink` object, remove Coulomb friction and then load the computed torque controller

```
>> mdl_puma560
>> p560 = p560.nofriction();
>> sl_ctorque
```

which is shown in Fig. 9.20. The torque is computed using the Toolbox `RNE` block and added to the feedback torque computed from position and velocity error. The desired joint angles and velocity are generated using a `jtraj` block whose parameters are the initial and final joint angles. We run the simulation

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

The tracking error is shown in Fig. 9.21.

9.4.4 Flexible Transmission

In many high-performance robots the flexibility of the transmission between motor and link can be a significant dynamic effect. The flexibility might be caused by torsional flexure of an elastic coupling between the motor and the shaft, the drive shaft itself,

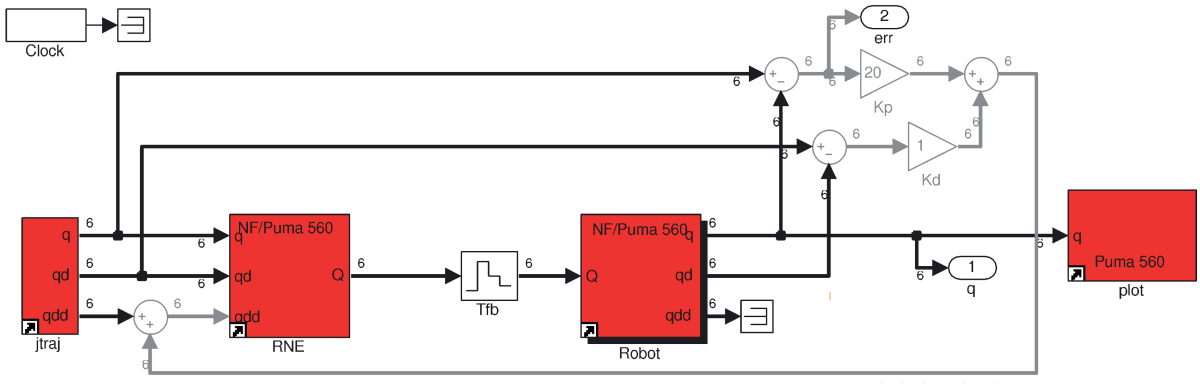


Fig. 9.20. Robotics Toolbox example `sl_ctorque`, computed torque control

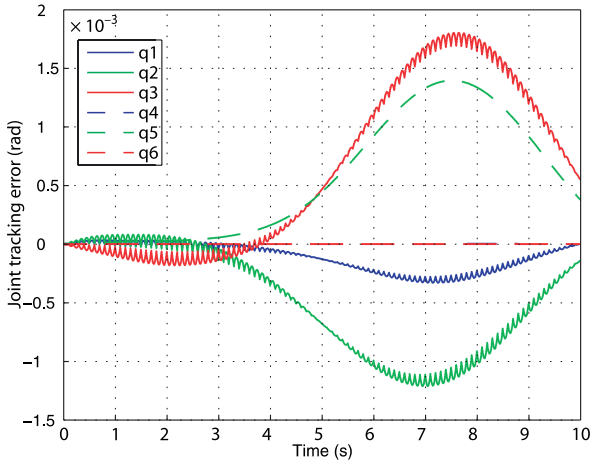


Fig. 9.21. Computed torque control: joint tracking error for first 3 joints

Series-elastic actuator (SEA). So far we have considered the flexibility between motor and load as a nuisance, but in many situations it has a real benefit. A class of actuators known as series-elastic actuators deliberately introduce a soft spring between the motor and the load. This is useful for robots that interact closely with people since it makes the robot less dangerous in case of collision. For robots that must exert a force as part of their task the extension of the elastic element provides an estimate of the applied force which can be used for control.

or in the case of a cable driven robot, the longitudinal stiffness of the cable. In dynamic terms the result is to add a poorly damped spring between the motor and the load.

We will demonstrate this for the 2-link robot introduced in Sect. 7.2.1. The Simulink® model

```
>> mdl_twolink
>> sl_flex
```

is shown in Fig. 9.22. The compliant drive is modeled by the motor angle q_m and the link angle q_l. The torque acting on the link is due to the spring constant and damping

$$\tau_l = K(q_m - q_l) + B(\dot{q}_m - \dot{q}_l)$$

For simplicity we assume that the motor and its control loop is infinitely stiff, that is, the reaction torque from the spring does not influence the angle of the motor.

We run the simulation

```
>> r = sim('sl_flex')
```

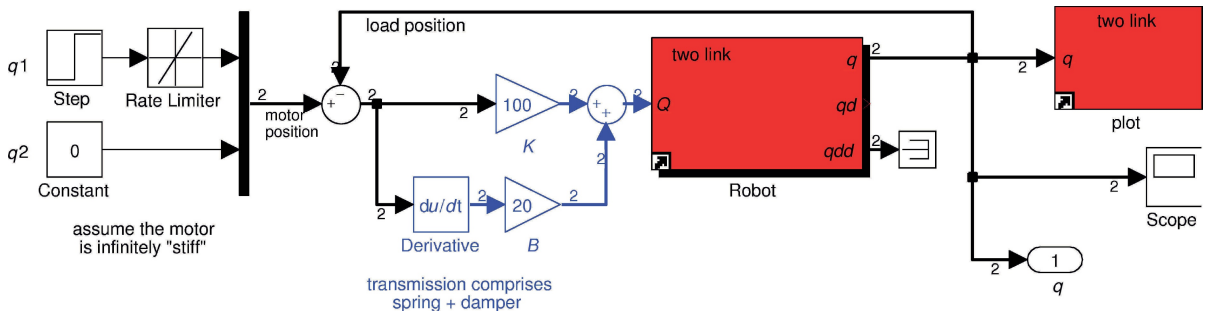


Fig. 9.22. Robotics Toolbox example `sl_flex`, simple flexible 2-link manipulator

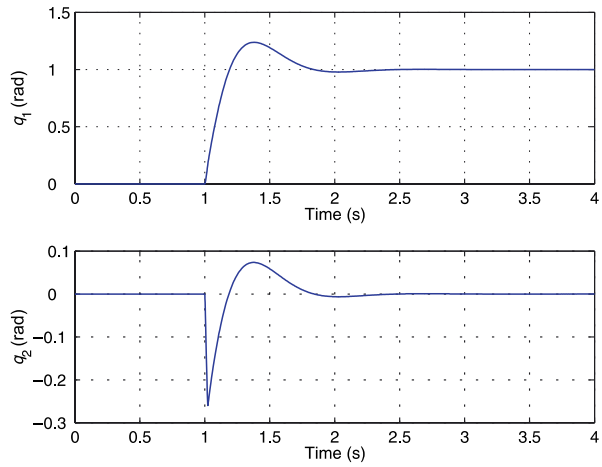


Fig. 9.23. Flexible link control: joint angle response to step input

and the results are shown in Fig. 9.23. The first joint receives a step position demand change at time 1 s and we see some significant overshoot in the response of both joints. Note that joint 2 has an initial negative error due to the inertial coupling from the acceleration of joint 1.

An effective controller for this system would need to measure the motor and the link angle, or the motor angle and the length of the spring for each joint. A more complex phenomena in high-performance robots is link flexibility, but this cannot be modelled using the rigid-link assumptions of the Toolbox.

9.5 Wrapping Up

In this Chapter we learnt how to model the forces and torques acting on the individual links of a serial-link manipulator. The equations of motion or inverse dynamics compute the joint forces required to achieve particular link angle, velocity and acceleration. The equations have terms corresponding to inertia, gravity, velocity coupling, friction and externally applied forces. We looked at the significance of these terms and how they vary with manipulator configuration. The equations of motion provide insight into important issues such as how the velocity or acceleration of one joint exerts a disturbance force on other joints which is important for control design. We then discussed the forward dynamics which describe how the configuration evolves with time in response to forces and torques applied at the joints by the actuators and by external forces such as gravity.

We then discussed approaches to control, starting with the simplest case of independent joint control, and explored the effect of disturbance torques and variation in inertia. We showed how feedforward of disturbances such as gravity could provide significant improvement in performance. We extended the feedforward notion to full model-based control using torque feedforward and computed torque controllers. Finally we discussed the effect of compliance between the robot motor and the link.

Further Reading

The dynamics of serial-link manipulators is well covered by all the standard robotics textbooks such as Paul (1981), Spong et al. (2006), Siciliano et al. (2008) and the Robotics Handbook (Siciliano and Khatib 2008, § 2). The efficient recursive Newton-Euler method we use today is the culmination of much research in the early 1980s and described in Hollerbach (1982). The equations of motion can be derived via a number of techniques, including Lagrangian (energy based), Newton-Euler, d'Alembert (Fu et al. 1987; Lee et al. 1983) or Kane's (Kane and Levinson 1983) method. However the computational cost of Lagrangian methods (Uicker 1965; Kahn 1969) is enormous, $O(N^4)$, which made it infeasible for real-time use on computers of that era and many simplifications and approximation had to be made. Orin et al. (1979) proposed an alternative approach based on the Newton-Euler (NE) equations of rigid-body motion applied to each link. Armstrong (1979) then showed how recursion could be applied resulting in $O(N)$ complexity. Luh et al. (1980) provided a recursive formulation of the Newton-Euler equations with linear and angular velocities referred to link coordinate frames which resulted in a 1 000-fold improvement in execution time making it practical to implement in real-time. Hollerbach (1980) showed how recursion could be applied to the Lagrangian form, and reduced the computation to within a factor of 3 of the recursive NE form, and Silver (1982) showed the equivalence of the recursive Lagrangian and Newton-Euler forms, and that the difference in efficiency was due to the representation of angular velocity.

The forward dynamics, Sect. 9.3, is computationally more expensive. An $O(N^3)$ method was proposed by Walker and Orin (1982). Featherstone's (1987) articulated-body method has $O(N)$ complexity but for $N < 9$ is more expensive than Walker's method.

Critical to any consideration of robot dynamics is knowledge of the inertial parameters, ten per link, as well as the motor's parameters. Corke and Armstrong-Hélouvry (1994, 1995) published a meta-study of Puma parameters and provide a consensus estimate of inertial and motor parameters for the Puma 560 robot. Some of this data is obtained by painstaking disassembly of the robot and determining the mass and dimensions of the components. Inertia of components can be estimated from mass and dimensions by assuming mass distribution, or it can be measured using a bifilar pendulum.

Alternatively the parameters can be estimated by measuring the joint torques or the base reaction force and moment as the robot moves. A number of early works in this area include Mayeda et al. (1990), Izaguirre and Paul (1985), Khalil and Dombre (2002) and a more recent summary is Siciliano and Khatib (2008, § 14). Key to successful identification is that the robot moves in a way that is sufficiently exciting (Gautier and Khalil 1992; Armstrong 1989). Friction is an important dynamic characteristic and is well described in Armstrong's (1988) thesis. The survey by Armstrong-Hélouvry et al. (1994) is a very readable and thorough treatment of friction modelling and control. A technique for measuring the electrical parameters of motors, such as torque constant and armature resistance, without having to remove the motor from the robot is described by Corke (1996a).

The discussion on control has been quite brief and has strongly emphasized the advantages of feedforward control. Robot joint control techniques are well covered by Spong et al. (2006), Craig (2004) and Siciliano et al. (2008) and summarized in Siciliano and Khatib (2008, § 6). Siciliano et al. have a good discussion of actuators and sensors as does the, now quite old, book by Klafter et al. (1989). The control of flexible joint robots is discussed in Spong et al. (2006). Adaptive control can be used to accomodate the time varying inertial parameters and there is a large literature on this topic but some good early references include the book by Craig (1987) and key papers include Craig et al. (1987), Spong (1989), Middleton and Goodwin (1988) and Ortega and Spong (1989).

Dynamic manipulability is discussed in Spong et al. (2006) and Siciliano et al. (2008). The Asada measure used in the Toolbox is described in Asada (1983).

Newton's Principia was written in Latin but an English translation is available on line at http://www.archive.org/details/newtonspmathema00newtrich. His writing on other subjects, including transcripts of his notebooks, can be found online at http://www.newtonproject.sussex.ac.uk.

Exercises

1. Run the code on page 194 to compute gravity loading on joints 2 and 3 as a function of configuration. Add a payload and repeat.
2. Run the code on page 195 to show how the inertia of joints 1 and 2 vary with payload?
3. Generate the curve of Fig. 9.2c. Add a payload and compare the results.
4. By what factor does this inertia vary over the joint angle range?
5. Why is the manipulator inertia matrix symmetric?
6. The robot exerts a wrench on the base as it moves (page 198). Consider that the robot is sitting on a frictionless horizontal table (say on a large air puck). Create a simulation model that includes the robot arm dynamics and the sliding dynamics on the table. Show that moving the arm causes the robot to translate and spin. Can you devise an arm motion that moves the robot base from one position to another and stops?
7. Overlay the manipulability ellipsoid on the display of the robot. Compare this with the velocity ellipsoid from Sect. 8.1.4.
8. Independent joint control (page 204)
   a) Investigate different values of `Kv` and `Ki` as well as demand signal shape and amplitude.
   b) Perform a root-locus analysis of `vloop` to determine the maximum permissible gain for the proportional case. Repeat this for the PI case.
   c) Consider that the motor is controlled by a voltage source instead of a current source, and that the motor's impedance is 1 mH and 1.6 Ω. Modify `vloop` accordingly. Extend the model to include the effect of back EMF.
   d) Increase the required speed of motion so that the motor torque becomes saturated. With integral action you will observe a phenomena known as integral windup – examine what happens to the state of the integrator during the motion. Various strategies are employed to combat this, such as limiting the maximum value of the integrator, or only allowing integral action when the motor is close to its setpoint. Experiment with some of these.
   e) Create a Simulink® model of the Puma robot with each joint controlled by `vloop` and `ploop`. Parameters for the different motors in the Puma are described in Corke and Armstrong-Hélouvry (1995).
9. The motor torque constant has units of N m A^{-1} and is equal to the back EMF constant which has units of V s rad^{-1}. Show that these units are equivalent.
10. Model-based control (page 211)
    a) Compute and display the joint tracking error for the torque feedforward and computed torque cases. Experiment with different motions, control parameters and samplerate T_{fb}.
    b) Reduce the rate at which the feedforward torque is computed and observe its effect on tracking error.
    c) In practice the dynamic model of the robot is not exactly known, we can only invert our best estimate of the rigid-body dynamics. In simulation we can model this by using the `perturb` method, see the online documentation, which returns a robot object with inertial parameters varied by plus and minus the specified percentage. Modify the Simulink® models so that the `RNE` block is using a

robot model with parameters perturbed by 10%. This means that the inverse dynamics are computed for a slightly different dynamic model to the robot under control and shows the effect of model error on control performance. Investigate the effects on error for both the torque feedforward and computed torque cases.

11. Flexible drive robot (page 213)
    a) Experiment with different values of joint stiffness and control parameters.
    b) Modify the simulation to eliminate the assumption that the motor is infinitely stiff. That is, replace the motor position with a nested control loop that is subject to a disturbance torque from the elastic element. Tune the controller for good performance.
    c) Introduce a rigid object into the environment and modify the simulation so that the robot arm moves to touch the object and applies no more than 5 N force to it.