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

\[
Q = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T \mathbf{g}
\]  

(9.1)

where \( q, \dot{q} \) and \( \ddot{q} \) are respectively the vector of generalized joint coordinates, velocities and accelerations, \( M \) is the joint-space inertia matrix, \( C \) is the Coriolis and centripetal coupling matrix, \( F \) is the friction force, \( G \) is the gravity loading, and \( Q \) is the vector of generalized actuator forces associated with the generalized coordinates \( q \). The last term gives the joint forces due to a wrench \( \mathbf{g} \) applied at the end effector and \( J \) 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

$$\mathbf{Q} = D(q, \dot{q}, \ddot{q})$$

(9.2)

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

```matlab
>> mdl_puma560
```

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

```matlab
>> 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 $\dot{q} = \ddot{q} = 0$) 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

```matlab
>> 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

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

which has returned

```matlab
>> 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

```matlab
>> Q(5,:)
an =
   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

```matlab
>> p560.rne(qn, [1 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

\[
f = m\ddot{v}
\]

where $m$ is the mass, $f$ the applied force and $v$ the velocity. The second equation, Euler’s equation of motion, describes the rotational motion

\[
\tau = J\ddot{\omega} + \omega \times J\omega
\]

where $\tau$ is the torque, $\omega$ is the angular velocity, and $J$ is the rotational inertia matrix (see page 81).