Chapter

3 Time and Motion

The only reason for time is
so that everything doesn't happen at once
Albert Einstein

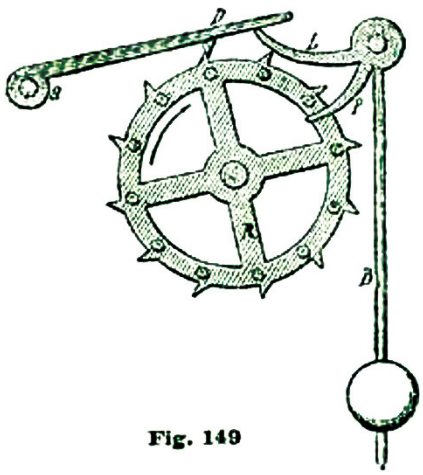


Fig. 149

In the previous chapter we learned how to describe the pose of objects in 2- or 3-dimensional space. This chapter extends those concepts to poses that change as a function of time. Section 3.1 introduces the derivative of time-varying position, orientation and pose and relates that to concepts from mechanics such as velocity and angular velocity. Discrete-time approximations to the derivatives are covered which are useful for computer implementation of algorithms such as inertial navigation. Section 3.2 is a brief introduction to the dynamics of objects moving under the influence of forces and torques and discusses the important difference between inertial and noninertial reference frames.

Section 3.3 discusses how to generate a temporal sequence of poses, a trajectory, that smoothly changes from an initial pose to a final pose. For robots this could be the path of a robot gripper moving to grasp an object or the flight path of a flying robot. Section 3.4 brings many of these topics together for the important application of inertial navigation. We introduce three common types of inertial sensor and learn how to how to use their measurements to update the estimate of pose for a moving object such as a robot.

3.1 Time-Varying Pose

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

3.1.1 Derivative of Pose

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

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

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

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

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

which we write succinctly as

$$ {}^{A}\dot{\boldsymbol{R}}_{B} = \left[{}^{A}\boldsymbol{\omega}\right]_{\times} {}^{A}\boldsymbol{R}_{B} \in \mathbb{R}^{3\times 3} \tag{3.1} $$

where ${}^{A}\boldsymbol{\omega} = {}^{A}\hat{\boldsymbol{\omega}}\dot{\theta}$ is the angular velocity in frame $\{A\}$. This is a vector quantity ${}^{A}\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ that defines the *instantaneous* axis and rate of rotation. The direction of ${}^{A}\boldsymbol{\omega}$ is parallel to the axis about which the coordinate frame is rotating at a particular instant of time, and the magnitude $\|{}^{A}\boldsymbol{\omega}\|$ is the rate of rotation about that axis. Note that the derivative of a rotation matrix is not a rotation matrix, it is a general 3×3 matrix.

For a tumbling object the axis of rotation changes with time.

Consider now that angular velocity is expressed in frame $\{B\}$ and we know that

$$ {}^{A}\boldsymbol{\omega} = {}^{A}\boldsymbol{R}_{B}\, {}^{B}\boldsymbol{\omega} $$

and using the identity $[\boldsymbol{A}\boldsymbol{v}]_{\times} = \boldsymbol{A}[\boldsymbol{v}]_{\times}\boldsymbol{A}^{T}$ it follows that

$$ {}^{A}\dot{\boldsymbol{R}}_{B} = {}^{A}\boldsymbol{R}_{B}\left[{}^{B}\boldsymbol{\omega}\right]_{\times} \in \mathbb{R}^{3\times 3} \tag{3.2} $$

The derivative of a unit quaternion, the quaternion equivalent of Eq. 3.1, is defined as

$$ {}^{A}\dot{\boldsymbol{q}}_{B} = \tfrac{1}{2}\, {}^{A}\mathring{\omega} \circ {}^{A}\mathring{q}_{B} = \tfrac{1}{2}\, {}^{A}\mathring{q}_{B} \circ {}^{B}\mathring{\omega} \in \mathbb{H} \tag{3.3} $$

where $\mathring{\omega}$ is a pure quaternion formed from the angular velocity vector. These are implemented by the Toolbox methods `dot` and `dotb` respectively. The derivative of a unit-quaternion is not a unit-quaternion, it is a regular quaternion which can also be considered as a 4-vector.

The derivative of pose can be determined by expressing pose as a homogeneous transformation matrix

$$ \xi \sim {}^{A}\boldsymbol{T}_{B} = \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & {}^{A}\boldsymbol{t}_{B} \\ 0_{1\times 3} & 1 \end{pmatrix} $$

and taking the derivative with respect to time and substituting Eq. 3.1 gives

$$ \dot{\xi} \sim {}^{A}\dot{\boldsymbol{T}}_{B} = \begin{pmatrix} {}^{A}\dot{\boldsymbol{R}}_{B} & {}^{A}\dot{\boldsymbol{t}}_{B} \\ 0_{1\times 3} & 0 \end{pmatrix} = \begin{pmatrix} \left[{}^{A}\boldsymbol{\omega}\right]_{\times} {}^{A}\boldsymbol{R}_{B} & {}^{A}\dot{\boldsymbol{t}}_{B} \\ 0_{1\times 3} & 0 \end{pmatrix} $$

The rate of change can be described in terms of the current orientation ${}^{A}\boldsymbol{R}_{B}$ and *two* velocities. The linear or translational velocity $\boldsymbol{v} = {}^{A}\dot{\boldsymbol{t}}_{B}$ is the velocity of the origin of $\{B\}$ with respect to $\{A\}$. The angular velocity ${}^{A}\boldsymbol{\omega}_{B}$ we have already introduced. We can combine these two velocity vectors to create the spatial velocity vector

$$ {}^{A}\nu_{B} = \left({}^{A}\boldsymbol{v}_{B}, {}^{A}\boldsymbol{\omega}_{B}\right) \in \mathbb{R}^{6} $$

which is the instantaneous velocity of frame $\{B\}$ with respect to $\{A\}$.

Every point in the body has the same angular velocity. Knowing that, plus the translational velocity vector of any point is enough to fully describe the instantaneous motion of a rigid body. It is common to place $\{B\}$ at the body's center of mass.

3.1.2 Transforming Spatial Velocities

The velocity of a moving body can be expressed with respect to a world reference frame $\{A\}$ or the moving body frame $\{B\}$ as shown in Fig. 3.1. The spatial velocities are linearly related by

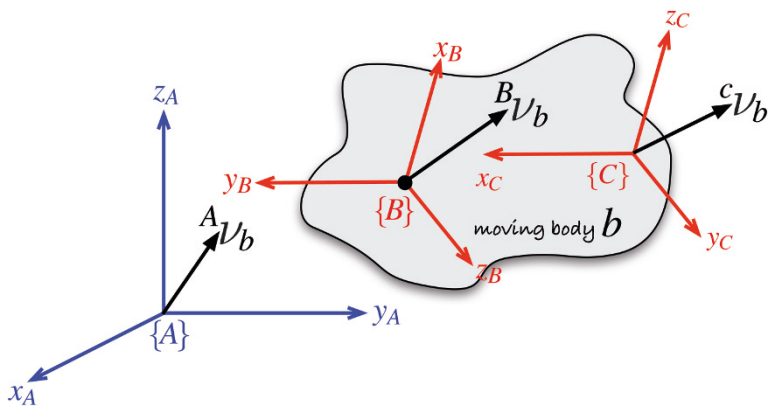


Fig. 3.1. Representing the spatial velocity of a moving body b with respect to various coordinate frames. Note that ν is a 6-dimensional vector

$$ {}^A\nu = \begin{pmatrix} {}^A\boldsymbol{R}_B & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & {}^A\boldsymbol{R}_B \end{pmatrix} {}^B\nu = {}^A\boldsymbol{J}_B\left({}^A\xi_B\right){}^B\nu \tag{3.4} $$

where ${}^A\xi_B \sim ({}^A\boldsymbol{R}_B, {}^A\boldsymbol{t}_B)$ and ${}^A\boldsymbol{J}_B(\cdot)$ is a Jacobian or interaction matrix. For example, we can define a body-fixed frame and a spatial velocity in that frame

```
>> TB = SE3(1, 2, 0) * SE3.Rz(pi/2);
>> vb = [0.2 0.3 0 0 0 0.5]';
```

and the spatial velocity in the world frame is

```
>> va = TB.velxform * vb;
>> va'
ans =
    0.2000    0.0000    0.3000         0   -0.5000    0.0000
```

For the case where frame $\{C\}$ is also on the moving body the transformation becomes

$$ {}^C\nu = \begin{pmatrix} {}^C\boldsymbol{R}_B & \left[{}^C\boldsymbol{t}_B\right]_\times {}^C\boldsymbol{R}_B \\ \boldsymbol{0}_{3\times 3} & {}^C\boldsymbol{R}_B \end{pmatrix} {}^B\nu = \mathrm{Ad}\left({}^C\xi_B\right){}^B\nu $$

and involves the adjoint matrix of the relative pose which is discussed in Appendix D. Continuing the example above we will define an additional frame $\{C\}$ relative to frame $\{B\}$

```
>> TBC = SE3(0, 0.4, 0);
```

To determine velocity at the origin of this frame we first compute ${}^C\xi_B$

```
>> TCB = inv(TBC);
```

and the velocity in frame $\{C\}$ is

```
>> vc = TBC.Ad * vb;
>> vc'
ans =
         0    0.3000         0         0         0    0.5000
```

which has zero velocity in the x_C-direction since the rotational and translational velocity components cancel out.

Lynch and Park (2017) use the term velocity twist while Murray et al. 1994 call this a spatial velocity.

The scalar product of a velocity twist and a wrench represents power.

Some texts introduce a *velocity twist* $\boldsymbol{V}$ which is different to the spatial velocity introduced above. The velocity twist of a body-fixed frame $\{B\}$ is ${}^B\boldsymbol{V} = ({}^B\boldsymbol{v}, {}^B\boldsymbol{\omega})$ which has a translational and rotational velocity component but ${}^B\boldsymbol{v}$ is the body-frame velocity of an imaginary point rigidly attached to the body and located at the world frame origin. The body- and world-frame velocity twists are related by the adjoint matrix rather than Eq. 3.4. The velocity twist is the dual of the wrench described in Sect. 3.2.2.

3.1.3 Incremental Rotation

The physical meaning of $\dot{\boldsymbol{R}}$ is not intuitively obvious – it is simply the way that the elements of $\boldsymbol{R}$ change with time. To gain some insight we consider a first-order approximation to the derivative

The only valid operator for the group SO(n) is composition $\oplus$, so the result of subtraction cannot belong to the group. The result is a 3×3 matrix of element-wise differences. Groups are introduced in Appendix D.

$$\dot{\boldsymbol{R}} \approx \frac{\boldsymbol{R}\langle t+\delta_t\rangle - \boldsymbol{R}\langle t\rangle}{\delta_t} \in \mathbb{R}^{3\times 3} \tag{3.5}$$

Consider an object whose body frames $\{B\}$ at two consecutive timesteps are related by a small rotation ${}^{B}\boldsymbol{R}_{\Delta}$ expressed in the body frame

$$\boldsymbol{R}_B\langle t+\delta_t\rangle = \boldsymbol{R}_B\langle t\rangle\, {}^{B}\boldsymbol{R}_{\Delta}$$

We substitute Eq. 3.2 into 3.5 and rearrange to obtain

$${}^{B}R_{\Delta} \approx \delta_t \left[{}^{B}\boldsymbol{\omega}\right]_{\times} + \boldsymbol{I}_{3\times 3} \tag{3.6}$$

which says that an infinitesimally small rotation can be approximated by the sum of a skew-symmetric matrix and an identity matrix. For example

This is the first two terms of the Rodrigues' rotation formula on, Eq. 2.18, when $\theta = \delta_t \boldsymbol{\omega}$.

```
>> rotx(0.001)
ans =
    1.0000         0         0
         0    1.0000   -0.0010
         0    0.0010    1.0000
```

Equation 3.6 directly relates rotation between timesteps to the angular velocity. Rearranging it allows us to compute the approximate angular velocity vector

$$\boldsymbol{\omega} \approx \tfrac{1}{\delta_t} \vee_{\times} \left(\boldsymbol{R}_B\langle t\rangle^{T} \boldsymbol{R}_B\langle t+\delta_t\rangle - \boldsymbol{I}_{3\times 3} \right)$$

from two consecutive rotation matrices where $\vee_{\times}(\cdot)$ is the inverse skew-symmetric matrix operator such that if $\mathbf{S} = [\boldsymbol{v}]_{\times}$ then $\boldsymbol{v} = \vee_{\times}(\mathbf{S})$. Alternatively, if the angular velocity in the body frame is known we can approximately update the rotation matrix

$$\boldsymbol{R}_B\langle t+\delta_t\rangle \approx \boldsymbol{R}_B\langle t\rangle + \delta_t \boldsymbol{R}_B\langle t\rangle \left[\boldsymbol{\omega}\right]_{\times} \tag{3.7}$$

The only valid operator for the group SO(n) is composition $\oplus$, so the result of addition cannot be within the group. The result is a general 3×3 matrix.

which is cheap to compute, involves no trigonometric operations, and is key to inertial navigation systems which we discuss in Sect. 3.4.

Adding any nonzero matrix to a rotation matrix results in a matrix that is *not* a rotation matrix. However if the increment is sufficiently small, that is the angular velocity and/or sample time is small, the result will be close to orthonormal and we can *straighten it up*. The resulting matrix should be normalized, as discussed in Sect. 2.3.1, to make it a proper rotation matrix. This is a common approach when implementing inertial navigation systems on low-end computing hardware.

Which is why inertial navigation systems operate at a high sample rate and δ_t is small.

We can also approximate the quaternion derivative by a first-order difference

$$\dot{q} \approx \frac{\mathring{q}\langle k+1\rangle - \mathring{q}\langle k\rangle}{\delta_t} \in \mathbb{H}$$

which combined with Eq. 3.3 gives us the approximation

Similar to the case for SO(n), addition and subtraction are not operators for the unit-quaternion group $\mathbb{S}^3$ so the result will be a quaternion $q \in \mathbb{H}$ for which addition and substraction are permitted. The Toolbox supports this with overloaded operators + and − and appropriate object class conversions.

$$\mathring{q}\langle k+1\rangle \approx \mathring{q}\langle k\rangle + \frac{\delta_t}{2}\mathring{\omega} \circ \mathring{q}\langle k\rangle \tag{3.8}$$

which is even cheaper to compute than the rotation matrix approach. Adding a nonzero vector to a unit-quaternion results in a nonunit quaternion but if the angular velocity and/or sample time is small then the approximation is reasonable. Normalizing the result to create a unit-quaternion is computationally cheaper than normalizing a rotation matrix, as discussed in Sect. 2.3.1.

3.1.4 Incremental Rigid-Body Motion

Consider two poses ξ_1 and ξ_2 which differ infinitesimally and are related by

$$\xi_2 = \xi_1 \oplus \xi_\Delta$$

where $\xi_\Delta = \ominus\xi_1 \oplus \xi_2$. In homogeneous transformation matrix form

$$\xi_\Delta \sim \boldsymbol{T}_\Delta = \begin{pmatrix} \boldsymbol{R}_\Delta & \boldsymbol{t}_\Delta \\ 0_{1\times 3} & 1 \end{pmatrix}$$

where $\boldsymbol{t}_\Delta$ is an incremental displacement and $\boldsymbol{R}_\Delta$ is an incremental rotation matrix which will be skew symmetric with only three unique elements $\vee_x(\boldsymbol{R}_\Delta - \boldsymbol{I}_{3\times 3})$ plus an identity matrix. The incremental rigid-body motion can therefore be described by just six parameters

$$\Delta(\xi_1, \xi_2) \mapsto \boldsymbol{\Delta}_\xi \in \mathbb{R}^6$$

This is useful in optimization procedures that seek to minimize the error between two poses: we can choose the cost function $e = \|\Delta(\xi_1, \xi_2)\|$ which is equal to zero when $\xi_1 \equiv \xi_2$. This is very approximate when the poses are significantly different, but becomes ever more accurate as $\xi_1 \to \xi_2$.

where $\boldsymbol{\Delta}_\xi = (\boldsymbol{\Delta}_t, \boldsymbol{\Delta}_R)$ can be considered as a spatial displacement. A body with constant spatial velocity $\boldsymbol{\nu}$ for δ_t seconds undergoes a spatial displacement of $\boldsymbol{\Delta}_\xi = \delta_t \boldsymbol{\nu}$.

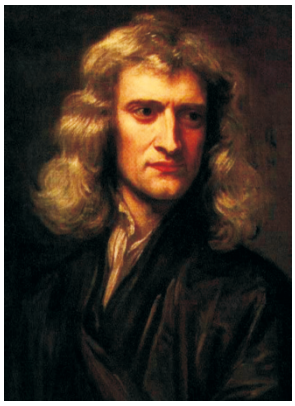
The inverse operator

$$\Delta^{-1}\left(\boldsymbol{\Delta}_\xi\right) \mapsto \xi_\Delta \in \mathbf{SE}(3)$$

is given by

$$\xi_\Delta \sim \boldsymbol{T}_\Delta = \begin{pmatrix} \left[\boldsymbol{\Delta}_R\right]_\times + \boldsymbol{I}_{3\times 3} & \boldsymbol{\Delta}_t \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}$$

The spatial displacement operator and its inverse are implemented by the Toolbox functions `tr2delta` and `delta2tr` respectively. These functions assume that the displacements are infinitesimal and become increasingly approximate with displacement magnitude.



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 287), 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 color and the phenomena of diffraction. The SI unit of force is named in his honor. He is buried in Westminster Abbey, London.

3.2 Accelerating Bodies and Reference Frames

So far we have considered only the first derivative, the velocity of coordinate frames. However all motion is ultimately caused by a force or a torque which leads to acceleration and the consideration of dynamics.

3.2.1 Dynamics of Moving Bodies

For translational motion Newton's second law describes, in the inertial frame, the acceleration of a particle with position x and mass m

$$m\,{}^0\ddot{\boldsymbol{x}} = {}^0\boldsymbol{f} \tag{3.9}$$

due to the applied force f.

Rotational motion in $\mathbf{SO}(3)$ is described by Euler's equations of motion which relates the angular acceleration of the body in the body frame

$${}^B\boldsymbol{J}\,{}^B\dot{\boldsymbol{\omega}} + {}^B\boldsymbol{\omega} \times \left({}^B\boldsymbol{J}\,{}^B\boldsymbol{\omega}\right) = {}^B\boldsymbol{\tau} \tag{3.10}$$

to the applied torque or moment τ and a positive-definite rotational inertia matrix ${}^B\boldsymbol{J} \in \mathbb{R}^{3\times 3}$. Nonzero angular acceleration implies that angular velocity, the axis and/or angle of rotation, evolves over time.

Notice that inertia has an associated reference frame, it is a matrix and its elements depend on the choice of the coordinate frame.

In the absence of torque a body generally rotates with a time-varying angular velocity – this is quite different to the linear velocity case. It is angular momentum $\boldsymbol{h} = \boldsymbol{J}\boldsymbol{\omega}$ in the inertial frame that is constant.

The matrix must be positive definite, that is symmetric and all its eigenvalues are positive.

Consider the motion of a tumbling object which we can easily simulate. We define an inertia matrix

```
>> J = [2 -1 0;-1 4 0;0 0 3];
```

and initial conditions for orientation and angular velocity

```
>> attitude = UnitQuaternion();
>> w = 0.2*[1 2 2]';
```

The simulation loop computes angular acceleration with Eq. 3.10, uses rectangular integration to obtain angular velocity and attitude, and then updates a graphical coordinate frame

```
>> dt = 0.05;
>> h = attitude.plot();
>> for t=0:dt:10
        wd = -inv(J) * (cross(w, J*w));
        w = w + wd*dt; attitude = attitude .* UnitQuaternion.omega(wd*dt);
        attitude.plot('handle', h); pause(dt)
   end
```

The rotational inertia of a body that moves in $\mathbf{SE}(3)$ is represented by the 3×3 symmetric matrix

$$\boldsymbol{J} = \begin{pmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{pmatrix}$$

The diagonal elements are the positive moments of inertia, and the off-diagonal elements are products of inertia. Only six of these nine elements are unique: three moments and three products of inertia. The products of inertia are all zero if the object's mass distribution is symmetrical with respect to the coordinate frame.

3.2.2 Transforming Forces and Torques

The spatial velocity is a vector quantity that represents translational and rotational velocity. In a similar fashion we can combine translational force and rotational torque into a 6-vector that is called a wrench $\boldsymbol{W} = (f_x, f_y, f_z, m_x, m_y, m_z) \in \mathbb{R}^6$. A wrench ${}^B\boldsymbol{W}$ is defined with respect to the coordinate frame $\{B\}$ and applied at the origin of that frame.

The wrench ${}^C\boldsymbol{W}$ is equivalent if it causes the same motion of the body when applied to the origin of coordinate frame $\{C\}$ and defined with respect to $\{C\}$. The wrenches are related by

$$ {}^C\boldsymbol{W} = \begin{pmatrix} {}^B\boldsymbol{R}_C & \left[{}^B\boldsymbol{t}_C\right]_\times {}^B\boldsymbol{R}_C \\ \boldsymbol{0}_{3\times 3} & {}^B\boldsymbol{R}_C \end{pmatrix}^T {}^B\boldsymbol{W} = \mathrm{Ad}\left({}^B\xi_C\right)^T {}^B\boldsymbol{W} \tag{3.11} $$

which is similar to the spatial velocity transform of Eq. 3.4 but uses the transpose of the adjoint of the *inverse* relative pose.

Continuing the MATLAB example from page 65 we define a wrench with respect to frame $\{B\}$ with forces of 3 and 4 Nm in the x- and y-directions respectively

```
>> WB = [3 4 0 0 0 0]';
```

The equivalent wrench in frame $\{C\}$ would be

```
>> WC = TBC.Ad' * WB;
>> WC'
ans =
    3.0000    4.0000         0         0         0    1.2000
```

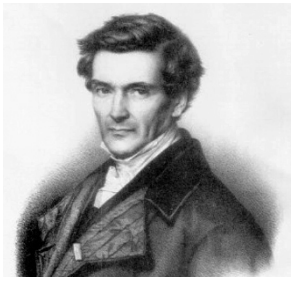
which is the same forces as applied at $\{B\}$ *plus* a torque of 1.2 Nm about the z-axis to counter the moment due to the application of the x-axis force along a different line of action.

3.2.3 Inertial Reference Frame

The term *inertial reference frame* is frequently used in robotics and it is crisply defined as "*a reference frame that is not accelerating or rotating*".

Consider a particle P at rest with respect to a stationary reference frame $\{0\}$. Frame $\{B\}$ is moving with constant velocity ${}^0\boldsymbol{v}_B$ relative to frame $\{0\}$. From the perspective of $\{B\}$ the particle would be moving at constant velocity, in fact ${}^B\boldsymbol{v}_P = -{}^0\boldsymbol{v}_B$. The particle is not accelerating and obeys Newton's first law "*that in the absence of an applied force a particle moves at a constant velocity*". Frame $\{B\}$ is therefore also an inertial reference frame.

Now imagine that frame $\{B\}$ is accelerating at a constant acceleration ${}^0\boldsymbol{a}_B$ with respect to $\{0\}$. From the perspective of $\{B\}$ the particle appear to be accelerating, in fact ${}^B\boldsymbol{a}_P = -{}^0\boldsymbol{a}_B$ and this violates Newton's first law. An observer in frame $\{B\}$ who was aware of Newton's theories might invoke some magical force to explain what they observe. We call such a force a fictitious, apparent, pseudo, inertial or d'Alembert force – they only exist in an accelerating or noninertial reference frame. This accelerating



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 systèmes 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.

frame $\{B\}$ is *not* an inertial reference frame. In Newtonian mechanics, gravity is considered a real body force $m\boldsymbol{g}$– a free object will accelerate relative to the inertial frame.

Albert Einstein's equivalence principle is that "*we assume the complete physical equivalence of a gravitational field and a corresponding acceleration of the reference system*" – we are unable to distinguish between gravity and being on a rocket accelerating at 1 *g* far from the gravitational influence of any celestial object.

An everyday example of a noninertial reference frame is an accelerating car or airplane. Inside an accelerating vehicle we observe fictitious forces pushing objects around in a way that is not explained by Newton's law in an inertial reference frame. We also experience real forces acting on our body which, in this case, are provided by the seat and the restraint.

For a rotating reference frame things are more complex still. Imagine that you and a friend are standing on a large rotating turntable, and throwing a ball back and forth. You will observe that the ball follows a curved path in space. As a Newton-aware observer in this noninertial reference frame you would have to resort to invoking some magical force that explains why flying objects follow curved paths.

Of course if we look down onto the turntable from an inertial reference frame the ball is moving in a straight line.

If the reference frame $\{B\}$ is rotating with angular velocity $\boldsymbol{\omega}$ about its origin then Newton's second law Eq. 3.9 becomes

$$m\left({}^{B}\ddot{\boldsymbol{v}} + \underbrace{\boldsymbol{\omega}\times\left(\boldsymbol{\omega}\times{}^{B}\boldsymbol{p}\right)}_{\text{centripetal}} + \underbrace{2\boldsymbol{\omega}\times{}^{B}\boldsymbol{v}}_{\text{Coriolis}} + \underbrace{\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{dt}}\times{}^{B}\boldsymbol{p}}_{\text{Euler}}\right) = {}^{0}\boldsymbol{f}$$

with three *new* acceleration terms. Centripetal acceleration always acts inward toward the origin. If the point is moving then Coriolis acceleration will be normal to its velocity. If rotational velocity is time varying then Euler acceleration will be normal to the position vector. Frequently the centripetal term is moved to the right-hand side in which case it becomes a fictitious outward centrifugal force. This complexity is symptomatic of being in a noninertial reference frame, and another definition of an inertial frame is one in which the "*physical laws hold good in their simplest form*".

Einstein, "*The foundation of the general theory of relativity*".

In robotics the term inertial frame and world coordinate frame tend to be used loosely and interchangeably to indicate a frame fixed to some point on the Earth. This is to distinguish it from the body-frame attached to the robot or vehicle. The surface of the Earth is an approximation of an inertial reference frame – the effect of the Earth's rotation is a finite acceleration less than 0.04 m s^{-2} due to centripetal acceleration. From the perspective of an Earth-bound observer a moving body will experience Coriolis acceleration. Both effects are small, dependent on latitude, and typically ignored.

Coriolis acceleration is significant for weather systems and meteorological prediction but below the sensitivity of low-cost sensors.

3.3 Creating Time-Varying Pose

In robotics we often need to generate a time-varying pose that moves smoothly in translation and rotation. A path is a spatial construct – a locus in space that leads from an initial pose to a final pose. A trajectory is a path with specified timing. For example there is a path from A to B, but there is a trajectory from A to B in 10 s or at 2 m s^{-1}.

An important characteristic of a trajectory is that it is *smooth* – position and orientation vary smoothly with time. We start by discussing how to generate smooth trajectories in one dimension. We then extend that to the multi-dimensional case and then to piecewise-linear trajectories that visit a number of intermediate points without stopping.

3.3.1 Smooth One-Dimensional Trajectories

We start our discussion with a scalar function of time. Important characteristics of this function are that its initial and final value are specified and that it is *smooth*. Smoothness in this context means that its first few temporal derivatives are continuous. Typically velocity and acceleration are required to be continuous and sometimes also the derivative of acceleration or jerk.

An obvious candidate for such a function is a polynomial function of time. Polynomials are simple to compute and can easily provide the required smoothness and boundary conditions. A quintic (fifth-order) polynomial is often used

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F \tag{3.12}$$

where time $t \in [0, T]$. The first- and second-derivatives are also smooth polynomials

$$\dot{s}(t) = 5At^4 + 4Bt^3 + 3Ct^2 + 2D + E \tag{3.13}$$

$$\ddot{s}(t) = 20At^3 + 12Bt^2 + 6Ct + 2D \tag{3.14}$$

Time	s	$\dot{s}$	$\ddot{s}$
$t=0$	s_0	$\dot{s}_0$	$\ddot{s}_0$
$t=T$	s_T	$\dot{s}_T$	$\ddot{s}_T$

The trajectory has defined boundary conditions for position, velocity and acceleration◂ and frequently the velocity and acceleration boundary conditions are all zero.

Writing Eq. 3.12 to Eq. 3.14 for the boundary conditions $t = 0$ and $t = T$ gives six equations which we can write in matrix form as

$$\begin{pmatrix} s_0 \\ s_T \\ \dot{s}_0 \\ \dot{s}_T \\ \ddot{s}_0 \\ \ddot{s}_T \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \end{pmatrix}$$

This is the reason for choice of quintic polynomial. It has six coefficients that enable it to meet the six boundary conditions on initial and final position, velocity and acceleration.

Since the matrix is square◂ we can solve for the coefficient vector (A, B, C, D, E, F) using standard linear algebra methods such as the MATLAB \-operator. For a quintic polynomial acceleration will be a smooth cubic polynomial, and jerk will be a parabola.

The Toolbox function `tpoly` generates a quintic polynomial trajectory as described by Eq. 3.12. For example

```
>> tpoly(0, 1, 50);
```

generates a polynomial trajectory and plots it, along with the corresponding velocity and acceleration, as shown in Fig. 3.2a. We can get these values into the workspace by providing output arguments

```
>> [s,sd,sdd] = tpoly(0, 1, 50);
```

where `s`, `sd` and `sdd` are respectively the trajectory, velocity and acceleration – each a 50×1 column vector. We observe that the initial and final velocity and acceleration

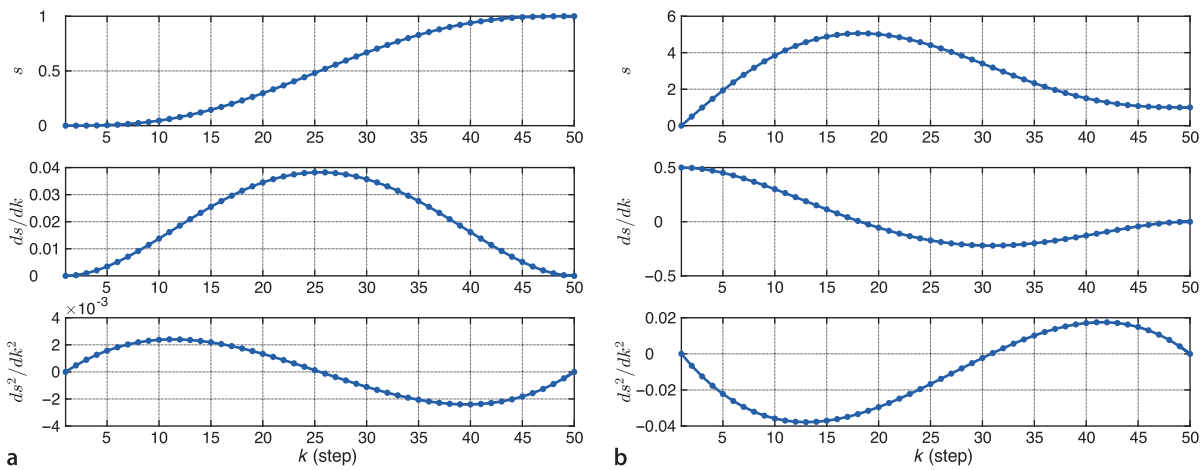


Fig. 3.2. Quintic polynomial trajectory. From top to bottom is position, velocity and acceleration versus time step. **a** With zero-velocity boundary conditions, **b** initial velocity of 0.5 and a final velocity of 0. Note that velocity and acceleration are in units of timestep not seconds

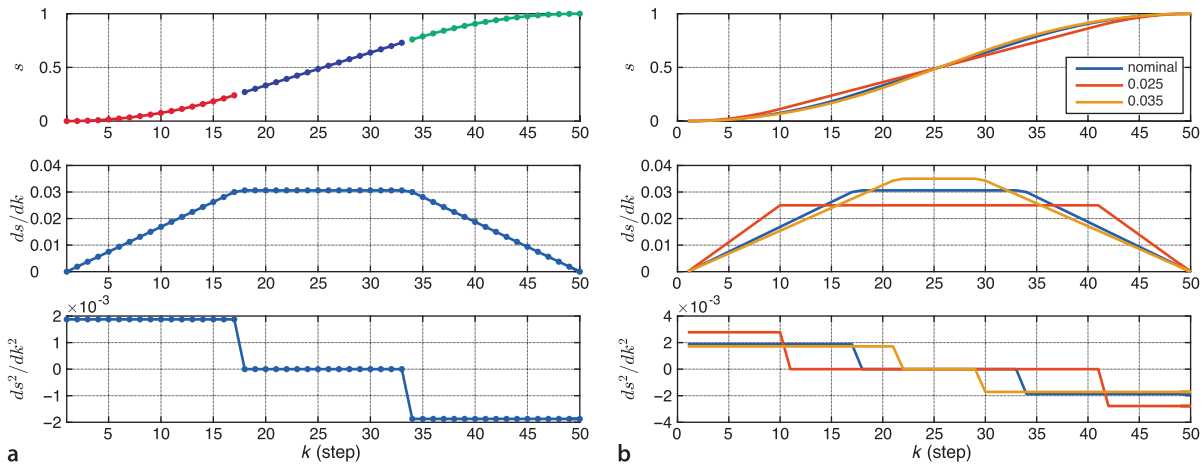


Fig. 3.3. Linear segment with parabolic blend (LSPB) trajectory: **a** default velocity for linear segment; **b** specified linear segment velocity values

are all zero – the default value. The initial and final velocities can be set to nonzero values

```
>> tpoly(0, 1, 50, 0.5, 0);
```

in this case, an initial velocity of 0.5 and a final velocity of 0. The results shown in Fig. 3.2b illustrate an important problem with polynomials. The nonzero initial velocity causes the polynomial to overshoot the terminal value – it peaks at 5 on a trajectory from 0 to 1.

Another problem with polynomials, a very practical one, can be seen in the middle graph of Fig. 3.2a. The velocity peaks at $k = 25$ which means that for most of the time the velocity is far less than the maximum. The mean velocity

```
>> mean(sd) / max(sd)
ans =
    0.5231
```

is only 52% of the peak so we are not using the motor as fully as we could. A real robot joint has a well defined maximum velocity and for minimum-time motion we want to be operating at that maximum for as much of the time as possible. We would like the velocity curve to be *flatter* on top.

A well known alternative is a hybrid trajectory which has a constant velocity segment with polynomial segments for acceleration and deceleration. Revisiting our first example the hybrid trajectory is

```
>> lspb(0, 1, 50);
```

where the arguments have the same meaning as for `tpoly` and the trajectory is shown in Fig. 3.3a. The trajectory comprises a linear segment (constant velocity) with parabolic blends, hence the name `lspb`. The term blend is commonly used to refer to a trajectory segment that smoothly joins linear segments. As with `tpoly` we can also return the trajectory and its velocity and acceleration

```
>> [s,sd,sdd] = lspb(0, 1, 50);
```

This type of trajectory is also referred to as trapezoidal due to the shape of the velocity curve versus time, and is commonly used in industrial motor drives.

The trapezoidal trajectory is smooth in velocity, but not in acceleration.

The function `lspb` has *chosen* the velocity of the linear segment to be

```
>> max(sd)
ans =
    0.0306
```

but this can be overridden by specifying it as a fourth input argument

```
>> s = lspb(0, 1, 50, 0.025);
>> s = lspb(0, 1, 50, 0.035);
```

The trajectories for these different cases are overlaid in Fig. 3.3b. We see that as the velocity of the linear segment increases its duration decreases and ultimately its duration would be zero. In fact the velocity cannot be chosen arbitrarily◀, too high or too low a value for the maximum velocity will result in an infeasible trajectory and the function returns an error.

The system has one design degree of freedom. There are six degrees of freedom (blend time, three parabolic coefficients and two linear coefficients) and five constraints (total time, initial and final position and velocity).

3.3.2 Multi-Dimensional Trajectories

Most useful robots have more than one axis of motion and it is quite straightforward to extend the smooth scalar trajectory to the vector case. In terms of configuration space (Sect. 2.3.5), these axes of motion correspond to the dimensions of the robot's configuration space – to its degrees of freedom. We represent the robot's configuration as a vector $\boldsymbol{q} \in \mathbb{R}^N$ where N is the number of degrees of freedom. The configuration of a 3-joint robot would be its joint angles $\boldsymbol{q} = (q_1, q_2, q_3)$. The configuration vector of wheeled mobile robot might be its position $\boldsymbol{q} = (x, y)$ or its position and heading angle $\boldsymbol{q} = (x, y, \theta)$. For a 3-dimensional body that had an orientation in $\mathbf{SO}(3)$ we would use a configuration vector $\boldsymbol{q} = (\theta_r, \theta_p, \theta_y)$ or for a pose in $\mathbf{SE}(3)$ we would use $\boldsymbol{q} = (x, y, z, \theta_r, \theta_p, \theta_y)$◀. In all these cases we would require smooth multi-dimensional motion from an initial configuration vector to a final configuration vector.

Or an equivalent 3-angle representation.

In the Toolbox this is achieved using the function `mtraj` and to move from configuration (0, 2) to (1, −1) in 50 steps we write

```
>> q = mtraj(@lspb, [0 2], [1 -1], 50);
```

which results in a 50 × 2 matrix `q` with one row per time step and one column per axis. The first argument is a handle to a function that generates a *scalar* trajectory, `@lspb` as in this case or `@tpoly`. The trajectory for the `@lspb` case

```
>> plot(q)
```

is shown in Fig. 3.4.

If we wished to create a trajectory for 3-dimensional pose we might consider converting a pose $\boldsymbol{T}$ to a 6-vector by a command like

```
q = [T1.t' T1.torpy]
```

though as we shall see later interpolation of 3-angle representations has some limitations.

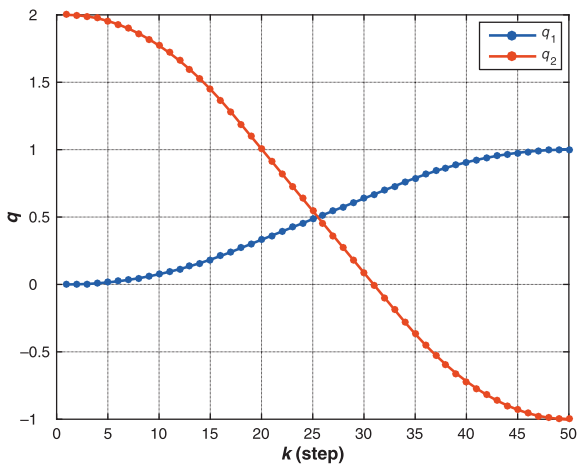


Fig. 3.4. Multi-dimensional motion. q_1 varies from $0 \rightarrow 1$ and q_2 varies from $2 \rightarrow -1$

3.3.3 Multi-Segment Trajectories

In robotics applications there is often a need to move smoothly along a path through one or more intermediate or *via* points without stopping. This might be to avoid obstacles in the workplace, or to perform a task that involves following a piecewise continuous trajectory such as welding a seam or applying a bead of sealant in a manufacturing application.

To formalize the problem consider that the trajectory is defined by M configurations $\boldsymbol{q}_k$, $k \in [1, M]$ and there are $M-1$ motion segments. As in the previous section $\boldsymbol{q}_k \in \mathbb{R}^N$ is a *vector* representation of configuration.

The robot starts from $\boldsymbol{q}_1$ at rest and finishes at $\boldsymbol{q}_M$ at rest, but moves through (or close to) the intermediate configurations without stopping. The problem is over constrained and in order to attain continuous velocity we surrender the ability to reach each intermediate configuration. This is easiest to understand for the 1-dimensional case shown in Fig. 3.5. The motion comprises linear motion segments with polynomial blends, like `lspb`, but here we choose quintic polynomials because they are able to match boundary conditions on position, velocity and acceleration at their start and end points.

The first segment of the trajectory accelerates from the initial configuration $\boldsymbol{q}_1$ and zero velocity, and joins the line heading toward the second configuration $\boldsymbol{q}_2$. The blend time is set to be a constant t_{acc} and $t_{\text{acc}}/2$ before reaching $\boldsymbol{q}_2$ the trajectory executes a polynomial blend, of duration t_{acc}, onto the line from $\boldsymbol{q}_2$ to $\boldsymbol{q}_3$, and the process repeats. The constant velocity $\dot{\boldsymbol{q}}_k$ can be specified for each segment. The average acceleration during the blend is

$$\ddot{\boldsymbol{q}} = \frac{\dot{\boldsymbol{q}}_{k+1} - \dot{\boldsymbol{q}}_k}{t_{\text{acc}}}$$

If the maximum acceleration capability of the axis is known then the minimum blend time can be computed.

The real limit of the axis will be its peak, rather than average, acceleration. The peak acceleration for the blend can be determined from Eq. 3.14 once the quintic coefficients are known.

On a particular motion segment each axis will have a different distance to travel and traveling at its maximum speed there will be a minimum time before it can reach its goal. The first step in planning a segment is to determine which axis will be the slowest to complete the segment, based on the distance that each axis needs to travel for the segment and its maximum achievable velocity. From this the duration of the segment can be computed and then the required velocity of each axis. This ensures that all axes reach the next target $\boldsymbol{q}_k$ at the *same time*.

The Toolbox function `mstraj` generates a multi-segment multi-axis trajectory based on a matrix of via points. For example 2-axis motion via the corners of a rotated square can be generated by

```
>> via = SO2(30, 'deg') * [-1 1; 1 1; 1 -1; -1 -1]';
>> q0 = mstraj(via(:,[2 3 4 1])', [2,1], [], via(:,1)', 0.2, 0);
```

The first argument is the matrix of via points, each row is the coordinates of a point. The remaining arguments are respectively: a vector of maximum speeds per axis, a vector of

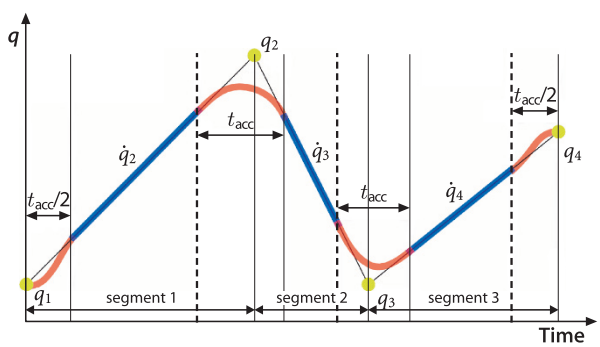


Fig. 3.5.
Notation for multi-segment trajectory showing four points and three motion segments. Blue indicates constant velocity motion, red indicates regions of acceleration

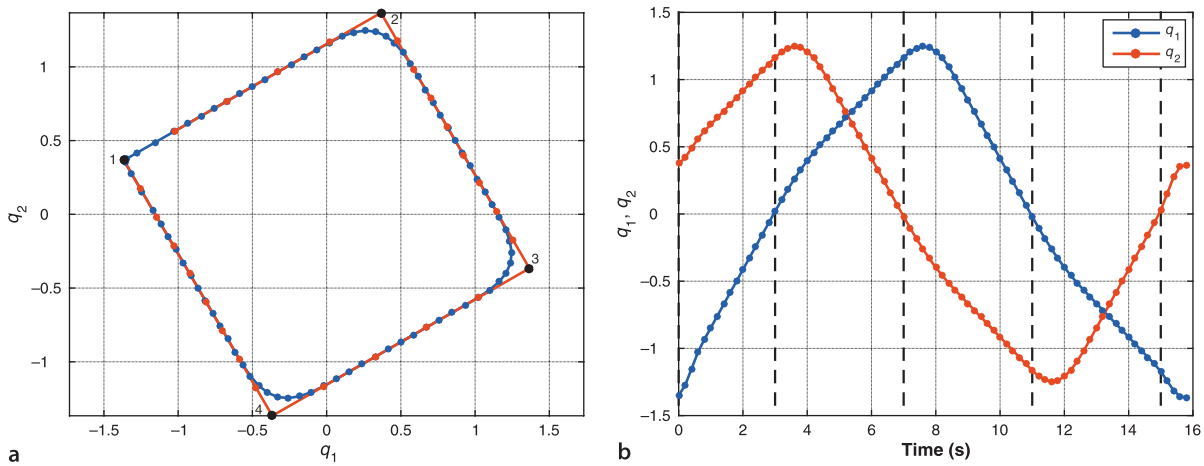


Fig. 3.6. Multi-segment multi-axis trajectories: **a** configuration of robot (tool position) for acceleration time of $t_{acc} = 0$ s (red) and $t_{acc} = 2$ s (blue), the via points are indicated by solid black markers; **b** configuration versus time with segment transitions ($t_{acc} = 2$ s) indicated by dashed black lines. The discrete-time points are indicated by dots

durations for each segment, the initial configuration, the sample time step, and the acceleration time. The function `mstraj` returns a matrix with one row per time step and the columns correspond to the axes. We can plot q_2 against q_1 to see the path of the robot

```
>> plot(q0(:,1), q0(:,2))
```

and is shown by the red path in Fig. 3.6a. If we increase the acceleration time

```
>> q2 = mstraj(via(:,[2 3 4 1])', [2,1], [], via(:,1)', 0.2, 2);
```

the trajectory becomes more rounded (blue path) as the polynomial blending functions do their work. The smoother trajectory also takes more time to complete.

```
>> [numrows(q0) numrows(q2)]
ans =
   28   80
```

Only one of the maximum axis speed or time per segment can be specified, the other is set to MATLAB's empty matrix `[]`.

Acceleration time if given is rounded up internally to a multiple of the time step.

The configuration variables as a function of time are shown in Fig. 3.6b. This function also accepts optional initial and final velocity arguments and t_{acc} can be a vector specifying different acceleration times for each of the N blends.

Keep in mind that this function simply interpolates pose represented as a vector. In this example the vector was assumed to be Cartesian coordinates, but this function could also be applied to Euler or roll-pitch-yaw angles but this is not an ideal way to interpolate rotation. This leads us nicely to the next section where we discuss interpolation of orientation.

3.3.4 Interpolation of Orientation in 3D

In robotics we often need to interpolate orientation, for example, we require the end-effector of a robot to smoothly change from orientation $\boldsymbol{\xi}_0$ to $\boldsymbol{\xi}_1$ in $\mathbf{SO}(3)$. We require some function $\boldsymbol{\xi}(s) = \sigma(\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, s)$ where $s \in [0, 1]$ which has the boundary conditions $\sigma(\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, 0) = \boldsymbol{\xi}_0$ and $\sigma(\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, 1) = \boldsymbol{\xi}_1$ and where $\sigma(\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, s)$ varies *smoothly* for intermediate values of s. How we implement this depends very much on our concrete representation of $\boldsymbol{\xi}$.

If pose is represented by an orthonormal rotation matrix, $\boldsymbol{\xi} \sim \boldsymbol{R} \in \mathbf{SO}(3)$, we might consider a simple linear interpolation $\sigma(\boldsymbol{R}_0, \boldsymbol{R}_1, s) = (1 - s)\boldsymbol{R}_0 + s\boldsymbol{R}_1$ but this would not, in general, be a valid orthonormal matrix which has strict column norm and inter-column orthogonality constraints.

A workable and commonly used approach is to consider a 3-angle representation such as Euler or roll-pitch-yaw angles, $\boldsymbol{\xi} \sim \boldsymbol{\Gamma} \in \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1$ and use linear interpolation

$$\sigma(\boldsymbol{\Gamma}_0, \boldsymbol{\Gamma}_1, s) = (1 - s)\boldsymbol{\Gamma}_0 + s\boldsymbol{\Gamma}_1$$

and converting the interpolated angles back to a rotation matrix always results in a valid form. For example we define two orientations

```
>> R0 = SO3.Rz(-1) * SO3.Ry(-1);
>> R1 = SO3.Rz(1) * SO3.Ry(1);
```

and find the equivalent roll-pitch-yaw angles

```
>> rpy0 = R0.torpy();  rpy1 = R1.torpy();
```

and create a trajectory between them over 50 time steps

```
>> rpy = mtraj(@tpoly, rpy0, rpy1, 50);
```

which is most easily visualized as an animation▶

```
>> SO3.rpy( rpy ).animate;
```

`rpy` is a 50×3 matrix and the result of `SO3.rpy` is a 1×50 vector of `SO3` objects, and their `animate` method is then called.

For large orientation changes we see that the axis around which the coordinate frame rotates changes along the trajectory. The motion, while smooth, sometimes looks uncoordinated. There will also be problems if either $\boldsymbol{\xi}_0$ or $\boldsymbol{\xi}_1$ is close to a singularity in the particular 3-angle system being used. This particular trajectory passes very close to the singularity, at around steps 24 and 25, and a symptom of this is the very rapid rate of change of roll-pitch-yaw angles at this point. The frame is not rotating faster at this point – you can verify that in the animation – the rotational parameters are changing very quickly and this is consequence of the particular representation.

Interpolation of unit-quaternions is only a little more complex than for 3-angle vectors and produces a change in orientation that is a rotation around a *fixed* axis in space. Using the Toolbox we first find the two equivalent quaternions

```
>> q0 = R0.UnitQuaternion;  q1 = R1.UnitQuaternion;
```

and then interpolate them

```
>> q = interp(q0, q1, 50);
>> about(q)
q [UnitQuaternion] : 1x50 (1.7 kB)
```

which results in a vector of 50 `UnitQuaternion` objects which we can animate by

```
>> q.animate
```

Quaternion interpolation is achieved using spherical linear interpolation (*slerp*) in which the unit quaternions follow a great circle path on a 4-dimensional hypersphere. The result in 3-dimensions is rotation about a fixed axis in space.

3.3.4.1 Direction of Rotation

When traveling on a circle we can move clockwise or counter-clockwise to reach the goal – the result is the same but the distance traveled may be different. On a sphere or hypersphere the principle is the same but now we are traveling on a great circle▶. In this example we animate a rotation about the z-axis, from an angle of -2 radians to $+2$ radians

```
>> q0 = UnitQuaternion.Rz(-2);  q1 = UnitQuaternion.Rz(2);
>> q = interp(q0, q1, 50);
>> q.animate()
```

A great circle on a sphere is the intersection of the sphere and a plane that passes through the center. On Earth the equator and all lines of longitude are great circles. Ships and aircraft prefer to follow great circles because they represent the shortest path between two points on the surface of a sphere.

but this is taking the long way around the circle, moving 4 radians when we could travel $2\pi - 4 \approx 2.28$ radians in the opposite direction. The `'shortest'` option requests the rotational interpolation to select the shortest path

```
>> q = interp(q0, q1, 50, 'shortest');
>> q.animate()
```

and the animation clearly shows the difference.

3.3.5 Cartesian Motion in 3D

Another common requirement is a smooth path between two poses in SE(3) which involves change in position as well as in orientation. In robotics this is often referred to as Cartesian motion.

We represent the initial and final poses as homogeneous transformations

```
>> T0 = SE3([0.4, 0.2, 0]) * SE3.rpy(0, 0, 3);
>> T1 = SE3([-0.4, -0.2, 0.3]) * SE3.rpy(-pi/4, pi/4, -pi/2);
```

The `SE3` object has a method `interp` that interpolates between two poses for normalized distance $s \in [0, 1]$ along the path, for example the midway pose between `T0` and `T1` is

```
>> interp(T0, T1, 0.5)
ans =
    0.0975   -0.7020    0.7055         0
    0.7020    0.5510    0.4512         0
   -0.7055    0.4512    0.5465    0.1500
         0         0         0         0
```

where the translational component is linearly interpolated and the rotation is spherically interpolated using the unit-quaternion interpolation method `interp`.

A trajectory between the two poses in 50 steps is created by

```
>> Ts = interp(T0, T1, 50);
```

This could also be written as `T0.interp(T1, 50)`.

where the arguments are the initial and final pose and the trajectory length.◀ The resulting trajectory `Ts` is a vector of `SE3` objects

```
>> about(Ts)
Ts [SE3] : 1x50 (6.5 kB)
```

representing the pose at each time step. The homogeneous transformation for the first point on the path is

```
>> Ts(1)
ans =
   -0.9900   -0.1411         0    0.4000
    0.1411   -0.9900         0    0.2000
         0         0    1.0000         0
         0         0         0    1.0000
```

and once again the easiest way to visualize this is by animation

```
>> Ts.animate
```

which shows the coordinate frame moving and rotating from pose `T0` to pose `T1`.

The translational part of this trajectory is obtained by◀

```
>> P = Ts.transl;
```

The `.t` property applied to a vector of `SE3` objects returns a MATLAB comma-separated list of translation vectors. The `.transl` method returns the translations in a more useful matrix form.

which returns the Cartesian position for the trajectory in matrix form

```
>> about(P)
P [double] : 50x3 (1.2 kB)
```

which has one row per time step that is the corresponding position vector. This is plotted

```
>> plot(P);
```

in Fig. 3.7 along with the orientation in roll-pitch-yaw format

```
>> rpy = Ts.torpy;
>> plot(rpy);
```

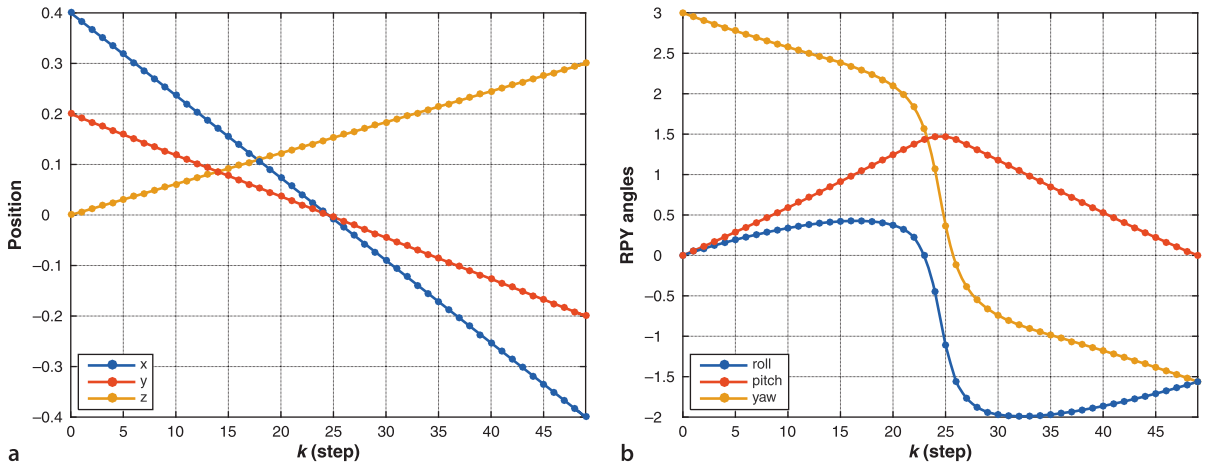


Fig. 3.7. Cartesian motion. **a** Cartesian position versus time, **b** roll-pitch-yaw angles versus time

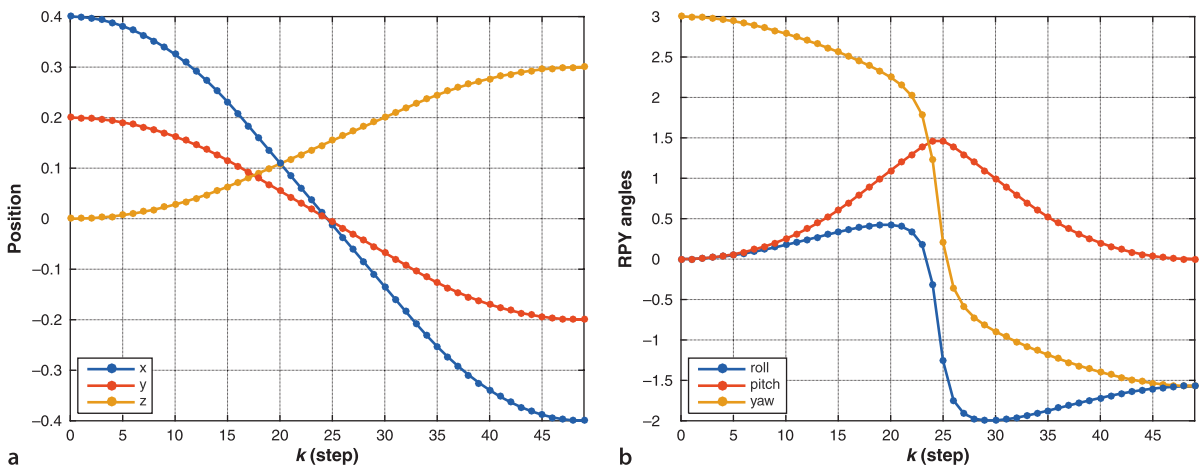


Fig. 3.8. Cartesian motion with LSPB path distance profile. **a** Cartesian position versus time, **b** roll-pitch-yaw angles versus time

We see that the position coordinates vary smoothly and linearly with time and that orientation varies smoothly with time.

However the motion has a velocity and acceleration *discontinuity* at the first and last points. While the path is smooth in space the distance s along the path is not smooth in time. Speed along the path jumps from zero to some finite value and then drops to zero at the end – there is no initial acceleration or final deceleration. The scalar functions `tpoly` and `lspb` discussed earlier can be used to generate s so that motion *along* the path is smooth. We can pass a vector of normalized distances along the path as the second argument to `interp`

The roll-pitch-yaw angles do not vary linearly with time because they represent a nonlinear transformation of the linearly varying quaternion.

```
>> Ts = T0.interp(T1, lspb(0, 1, 50) );
```

The trajectory is unchanged but the coordinate frame now accelerates to a constant speed along the path and then decelerates and this is reflected in smoother curves for the trajectory shown in Fig. 3.8. The Toolbox provides a convenient shorthand `ctraj` for the above

```
>> Ts = ctraj(T0, T1, 50);
```

where the arguments are the initial and final pose and the number of time steps.

Fig. 3.9. a SPIRE (Space Inertial Reference Equipment) from 1953 was 1.5 m in diameter and weighed 1200 kg. **b** A modern inertial navigation system the LORD MicroStrain 3DM-GX4-25 has triaxial gyroscopes, accelerometers and magnetometer, a pressure altimeter, is only $36 \times 24 \times 11$ mm and weighs 16 g (image courtesy of LORD MicroStrain); **c** 9 Degrees of Freedom IMU Breakout (LSM9DS1-SEN-13284 from SparkFun Electronics), the chip itself is only 3.5×3 mm

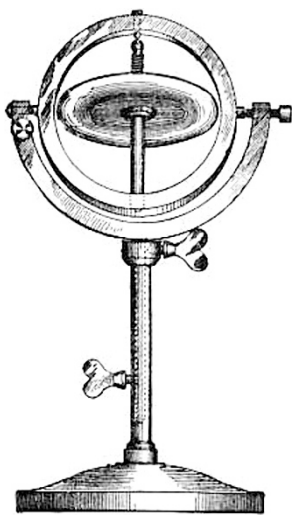
3.4 Application: Inertial Navigation

An inertial navigation system or INS is a "black box" that estimates its velocity, orientation and position by measuring accelerations and angular velocities and integrating them over time. Importantly it has no external inputs such as radio signals from satellites. This makes it well suited to applications such as submarine, spacecraft and missile guidance where it is not possible to communicate with radio navigation aids or which must be immune to radio jamming. These particular applications drove development of the technology during the cold war and space race of the 1950s and 1960s. Those early systems were large, see Fig. 3.9a, extremely expensive and the technical details were national secrets. Today INSs are considerably cheaper and smaller as shown in Fig. 3.9b; the sensor chips shown in Fig. 3.9c can cost as little as a few dollars and they are built into every smart phone.

An INS estimates its pose with respect to an inertial reference frame which is typically denoted {0} and fixed to some point on the Earth's surface – the world coordinate frame.◀ The frame typically has its z-axis upward or downward and the x- and y-axes establish a local tangent plane. Two common conventions have the x-, y- and z-axes respectively parallel to north-east-down (NED) or east-north-up (ENU) directions. The coordinate frame $\{B\}$ is attached to the moving vehicle or robot and is known as the body- or body-fixed frame.

As discussed in Sect. 3.2.3 the Earth's surface is not an inertial reference frame, but for most robots with nonmilitary grade sensors this is a valid assumption.

3.4.1 Gyroscopes



Any sensor that measures the rate of change of orientation is known, for historical reasons, as a gyroscope.

3.4.1.1 How Gyroscopes Work

The term gyroscope conjures up an image of a childhood toy – a spinning disk in a round frame that can balance on the end of a pencil. Gyroscopes are confounding devices – you try to turn them one way but they resist and turn (precess) in a different direction. This unruly behavior is described by a simplified version of Eq. 3.10

$$\tau = \boldsymbol{\omega} \times \boldsymbol{h} \tag{3.15}$$

where $\boldsymbol{h}$ is the angular momentum of the gyroscope, a vector parallel to the rotor's axis of spin and with magnitude $\|\boldsymbol{h}\| = J\varpi$, where J is the rotor's inertia and ϖ its rotational speed. It is the cross product in Eq. 3.15 that makes the gyroscope move in a contrary way.

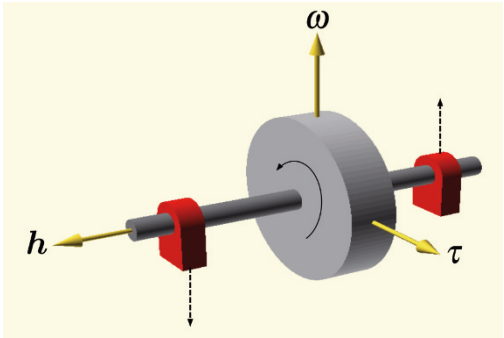


Fig. 3.10.
Gyroscope in strapdown configuration. Angular velocity ω induces a torque τ which can be sensed as forces at the bearings shown in red

If no torque is applied to the gyroscope its angular momentum remains constant in the inertial reference frame which implies that the axis will maintain a *constant direction* in that frame. Two gyroscopes with orthogonal axes form a stable platform that will maintain a *constant orientation* with respect to the inertial reference frame – fixed with respect to the universe. This was the principle of many early spacecraft navigation systems such as that shown in Fig. 2.15 – the vehicle was able to rotate about the stable platform and the spacecraft's orientation could be measured with respect to the platform.

The challenge was to create a mechanism that allowed the vehicle to rotate around the stable platform without exerting any torque on the gyroscopes. This required exquisitely engineered low-friction gimbals and bearing systems.

Alternatively we can fix the gyroscope to the vehicle in the strapdown configuration as shown in Fig. 3.10. If the vehicle rotates with an angular velocity ω the attached gyroscope will *resist* and exert an orthogonal torque τ which can be measured. If the magnitude of h is high then this kind of sensor is very sensitive – a very small angular velocity leads to an easily measurable torque.

Typically by strain gauges attached to the bearings of the rotor shaft.

Over the last few decades this rotating disk technology has been eclipsed by sensors based on optical principles such as the ring-laser gyroscope (RLG) and the fiber-optic gyroscope (FOG). These are high quality sensors but expensive and bulky. The low-cost sensors used in mobile phones and drones are based on micro-electro-mechanical systems (MEMS) fabricated on silicon chips. Details of the designs vary but all contain a mass vibrating at high frequency in a plane, and rotation about an axis normal to the plane causes an orthogonal displacement within the plane that is measured capacitively.

Typically over 10 kHz.

Gyroscopic angular velocity sensors measure rotation about a single axis. Typically three gyroscopes are packaged together and arranged so that their sensitive axes are orthogonal. The three outputs of such a triaxial gyroscope are the components of the angular velocity vector ${}^{B}\boldsymbol{\omega}^{\#}$ measured in the body frame $\{B\}$, and we introduce the # superscript to explicitly indicate a sensor measurement.

Interestingly, nature has invented gyroscopic sensors. All vertebrates have angular velocity sensors as part of their vestibular system. In each inner ear we have three semi-circular canals – fluid filled organs that measure angular velocity. They are arranged orthogonally, just like a triaxial gyroscope, with two measurement axes in a vertical plane and one diagonally across the head.

3.4.1.2 Estimating Orientation

If we assume that ${}^{B}\boldsymbol{\omega}$ is constant over a time interval δ_t the equivalent rotation at the timestep k is

$$ {}^{B}\xi_{\Delta}\langle k\rangle \sim \mathrm{e}^{\left[{}^{B}\omega^{\#}\right]_{\times}\delta_t} \tag{3.16} $$

If the orientation of the sensor frame is initially ξ_B then the evolution of estimated pose can be written in discrete-time form as

$$ \hat{\xi}_B\langle k{+}1\rangle \leftarrow \hat{\xi}_B\langle k\rangle \oplus {}^{B}\xi_{\Delta}\langle k\rangle \tag{3.17} $$

Much important development was undertaken by the MIT Instrumentation Laboratory under the leadership of Charles Stark Draper. In 1953 the feasibility of inertial navigation for aircraft was demonstrated in a series of flight tests with a system called SPIRE (Space Inertial Reference Equipment) shown in Fig. 3.9a. It was 1.5 m in diameter and weighed 1 200 kg. SPIRE guided a B-29 bomber on a 12 hour trip from Massachusetts to Los Angeles without the aid of a pilot and with Draper aboard. In 1954 the first self-contained submarine navigation system (SINS) was introduced to service. The Instrumentation Lab also developed the Apollo Guidance Computer, a one-cubic-foot computer that guided the Apollo Lunar Module to the surface of the Moon in 1969.

Today high-performance inertial navigation systems based on fiber-optic gyroscopes are widely available and weigh around one 1 kg while low-cost systems based on MEMS technology can weigh just a few grams and cost a few dollars.

where we use the hat notation to explicitly indicate an estimate of pose and $k \in \mathbb{Z}^+$ is the index of the time step. In concrete terms we can compute this *update* using $\mathbf{SO}(3)$ rotation matrices or unit-quaternions as discussed in Sect. 3.1.3 and taking care to normalize the rotation after each step.

We will demonstrate this integration using unit quaternions and simulated angular velocity data for a tumbling body. The script

```
>> ex_tumble
```

creates a matrix `w` whose columns represent consecutive body-frame angular velocity measurements with corresponding times given by elements of the vector `t`. We choose the initial pose to be the null rotation

```
>> attitude(1) = UnitQuaternion();
```

and then for each time step we update the orientation and keep the orientation history in a vector of quaternions

```
>> for k=1:numcols(w)-1
       attitude(k+1) = attitude(k) .* UnitQuaternion.omega( w(:,k)*dt );
   end
```

The `omega` method creates a unit-quaternion corresponding to a rotation angle and axis given by the magnitude and direction of its argument. The `.*` operator performs quaternion multiplication and normalizes the product, ensuring the result has a unit norm.◀ We can animate the changing orientation of the body frame

The `.increment` method of the UnitQuaternion class does this in a single call.

```
>> attitude.animate('time', t)
```

or view the roll-pitch-yaw angles as a function of time

```
>> mplot(t, attitude.torpy() )
```

3.4.2 Accelerometers

Accelerometers are sensors that measure acceleration. Even when not moving they sense the acceleration due to gravity which defines the direction we know as *downward*. Gravitational acceleration is a function of the material in the Earth beneath us and our distance from the Earth's center. The Earth is not a perfect sphere◀ and points in the equatorial region are further from the center. Gravitational acceleration can be approximated by

The technical term is an oblate spheroid, it bulges out at the equator because of centrifugal acceleration due to the Earth's rotation. The equatorial diameter is around 40 km greater than the polar diameter.

$$g \approx 9.780327\left(1 + 0.0053024\sin^2\theta - 0.0000058\sin^2 2\theta\right) - 0.000003086h$$

where θ is the angle of latitude and h is height above sea level. A map of gravity showing the effect of latitude and topography is shown in Fig. 3.11.

Charles Stark (Doc) Draper (1901–1987) was an American scientist and engineer, often referred to as "the father of inertial navigation." Born in Windsor, Missouri, he studied at the University of Missouri then Stanford where he earned a B.A. in psychology in 1922, then at MIT an S.B. in electro-chemical engineering and an S.M. and Sc.D. in physics in 1928 and 1938 respectively. He started teaching while at MIT and became a full professor in aeronautical engineering in 1939. He was the founder and director of the MIT Instrumentation Laboratory which made important contributions to the theory and practice of inertial navigation to meet the needs of the cold war and the space program.

Draper was named one of Time magazine's Men of the Year in 1961 and inducted to the National Inventors Hall of Fame in 1981. The Instrumentation lab was renamed Charles Stark Draper Laboratory (CSDL) in his honor. (Photo courtesy of The Charles Stark Draper Laboratory Inc.)

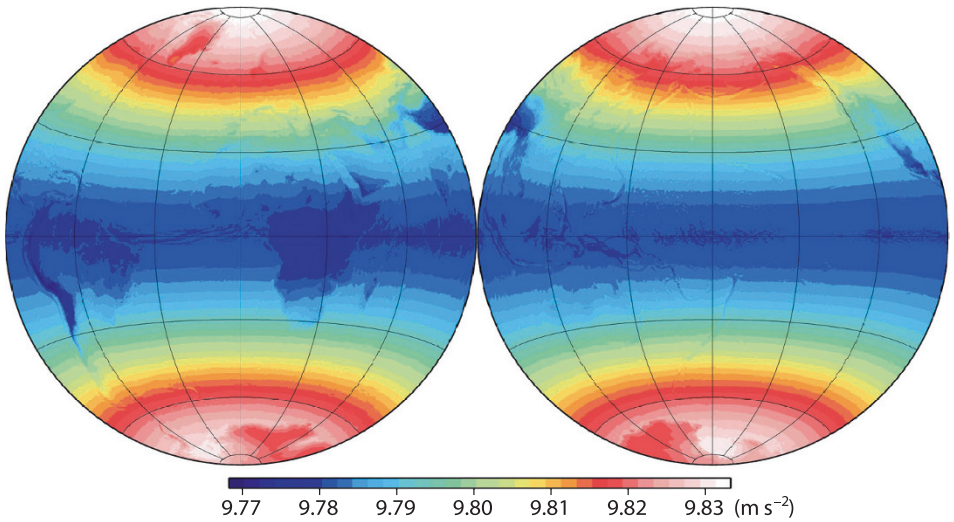


Fig. 3.11. Variation in Earth's gravitational acceleration, continents and mountain ranges are visible. The hemispheres shown are centerd on the prime (left) and anti (right) meridian respectively (from Hirt et al. 2013)

3.4.2.1 How Accelerometers Work

An accelerometer is conceptually a very simple device comprising a mass, known as the proof mass, supported by a spring as shown in Fig. 3.12. In the inertial reference frame Newton's second law for the proof mass is

$$m\ddot{x}_m = F_s - mg \tag{3.18}$$

and for a spring with natural length l_0 the relationship between force and extension d is

$$F_s = kd$$

The various displacements are related

$$x_b - (l_0 + d) = x_m$$

and taking the double derivative then substituting Eq. 3.18 gives

$$\ddot{x}_b - \ddot{d} = \tfrac{1}{m}(kd - mg)$$

The quantity we wish to measure is the acceleration of the accelerometer $a = \ddot{x}_b$[▶] and the relative displacement of the proof mass

$$d = \frac{m}{k}(a + g)$$

We assume that $\ddot{d} = 0$ in steady state. Typically there would be a damping element to increase friction and stop the proof mass oscillating. This adds a term $-B\dot{x}_m$ to the right-hand side of Eq. 3.18.

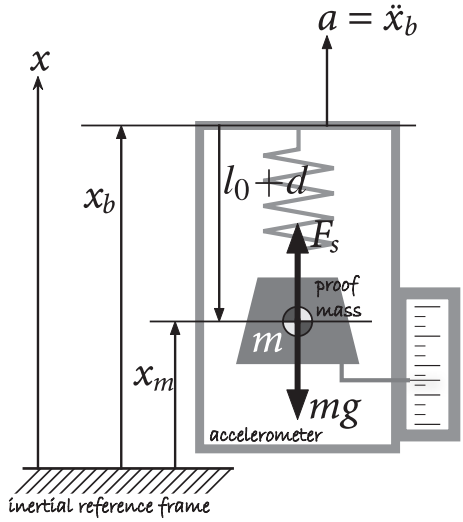


Fig. 3.12. The essential elements of an accelerometer and notation

is linearly related to that acceleration. In an accelerometer the displacement is measured and scaled by k/m so that the output of the sensor is

$$a^{\#} = a + g \ \text{m s}^{-2}$$

If this accelerometer is stationary then $a = 0$ yet the measured acceleration would be $a^{\#} = 0 + g = g$ in the upward direction. This is because our model has included the Newtonian gravity force $m\boldsymbol{g}$, as discussed in Sect. 3.2.3. Accelerometer output is sometimes referred to as specific, inertial or proper acceleration.

> The fact that a stationary accelerometer indicates an upward acceleration of 1 g is unintuitive since the accelerometer is clearly stationary and not accelerating. Intuition would suggest, that if anything, the acceleration should be in the downward direction where the device would accelerate if dropped. However the reality is that an accelerometer at rest in a gravity field reports upward acceleration.◀

A number of iPhone sensor apps incorrectly report acceleration in the *downward* direction when the phone is stationary.

Accelerometers measure acceleration along a single axis. Typically three accelerometers are packaged together and arranged so that their sensitive axes are orthogonal. The three outputs of such a triaxial accelerometer are the components of the acceleration vector ${}^{B}\boldsymbol{a}^{\#}$ measured in the body frame $\{B\}$.

Nature has also invented the accelerometer. All vertebrates have acceleration sensors called ampullae as part of their vestibular system. We have two in each inner ear: the saccule which measures vertical acceleration, and the utricle which measures front-to-back acceleration, and they help us maintain balance.◀ The proof mass in the ampullae is a collection of calcium carbonate crystals called otoliths, literally ear stones, on a gelatinous substrate which serves as the spring and damper. Hair cells embedded in the substrate measure the displacement of the otoliths due to acceleration.

Inconsistency between motion sensed in our ears and motion perceived by our eyes is the root cause of motion sickness.

3.4.2.2 Estimating Pose and Body Acceleration

In frame $\{0\}$ with its z-axis vertically upward, the gravitational acceleration vector is

$${}^{0}\boldsymbol{a} = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$

where g is the local gravitational acceleration from Fig. 3.11. In a body-fixed frame $\{B\}$ at an arbitrary orientation expressed in terms of ZYX roll-pitch-yaw angles

We could use any 3-angle sequence.

$$ {}^0\xi_B = \mathcal{R}_z(\theta_y) \oplus \mathcal{R}_y(\theta_p) \oplus \mathcal{R}_x(\theta_r) $$

the gravitational acceleration will be

$$ {}^B\boldsymbol{a} = \left(\ominus {}^0\xi_B\right) \bullet {}^0\boldsymbol{a} = \begin{pmatrix} -g\sin\theta_p \\ g\cos\theta_p \sin\theta_r \\ g\cos\theta_p \cos\theta_r \end{pmatrix} \tag{3.19} $$

The *measured* acceleration vector from the sensor in frame $\{B\}$ is

$$ {}^B\boldsymbol{a}^{\#} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} $$

and equating this with Eq. 3.19 we can solve for the roll and pitch angles

$$ \sin\hat{\theta}_p = \frac{-a_x}{g} \tag{3.20} $$

$$ \tan\hat{\theta}_r = \frac{a_y}{a_z}, \quad \theta_p \neq \pm\frac{\pi}{2} \tag{3.21} $$

These angles are sufficient to determine whether a phone, tablet or camera is in portrait or landscape orientation.

and we use the hat notation to indicate that these are estimates of the angles. Notice that there is no solution for the yaw angle and in fact θ_y does not even appear in Eq. 3.19. The gravity vector is parallel to the vertical axis and rotating around that axis, yaw rotation, will not change the measured value at all.

Another way to consider this is that we are essentially measuring the direction of the gravity vector with respect to the frame $\{B\}$ and a vector provides only two unique *pieces* of directional information, since one component of a unit vector can be written in terms of the other two.

> We have made a very strong assumption that the measured acceleration ${}^B\boldsymbol{a}^{\#}$ is only due to gravity. On a robot the sensor will experience additional acceleration as the vehicle moves and this will introduce an error in the estimated orientation.

Frequently we want to estimate the motion of the vehicle in the inertial frame, and the total measured acceleration in $\{0\}$ is due to gravity *and* motion

$$ {}^0\boldsymbol{a}^{\#} = {}^0\boldsymbol{g} + {}^0\boldsymbol{a}_v $$

We observe acceleration in the body frame so the vehicle acceleration in the world frame is

$$ {}^0\hat{\boldsymbol{a}}_v = {}^0\hat{\boldsymbol{R}}_B \, {}^B\boldsymbol{a}^{\#} - {}^0\boldsymbol{g} \tag{3.22} $$

and we assume that ${}^0\hat{\boldsymbol{R}}_B$ and $\boldsymbol{g}$ are both known. Integrating that with respect to time

The first assumption is a strong one and problematic in practice. Any error in the rotation matrix results in incorrect cancellation of the gravity component of $\boldsymbol{a}^{\#}$ which leads to an error in the estimated body acceleration.

$$ {}^0\hat{\boldsymbol{v}}_v(t) = \int {}^0\hat{\boldsymbol{a}}_v(t)\,\mathrm{d}t \tag{3.23} $$

gives the velocity of the vehicle, and integrating again

$$ {}^0\hat{\boldsymbol{p}}_v(t) = \int {}^0\hat{\boldsymbol{v}}_v(t)\,\mathrm{d}t \tag{3.24} $$

gives its position. Note that we can assume vehicle acceleration is zero and estimate attitude, or assume attitude and estimate vehicle acceleration. We cannot estimate both since there are more unknowns than measurements.

3.4.3 Magnetometers

The Earth is a massive but weak magnet. The poles of this geomagnet are the Earth's north and south magnetic poles which are constantly moving and located quite some distance from the planet's rotational axis.

At any point on the planet the magnetic flux lines can be considered a vector $\boldsymbol{m}$ whose magnitude and direction can be accurately predicted and mapped as shown in Fig. 3.13. We describe the vector's direction in terms of two angles: declination and inclination. A horizontal projection of the vector $\boldsymbol{m}$ points in the direction of magnetic north and the declination angle D is measured from true north◀ clockwise to that projection. The inclination angle I of the vector is measured in a vertical plane downward◀ from horizontal to $\boldsymbol{m}$. The length of the vector, the magnetic field intensity, is measured by a magnetometer in units of Tesla (T) and for the Earth this varies from 25−65 μT◣ as shown in Fig. 3.13a.

The direction of the Earth's north rotational pole, where the rotational axis intersects the surface of the northern hemisphere.

In the Northern hemisphere inclination is positive, that is, the vector points into the ground.

By comparison a modern MRI machine has a magnetic field strength of 4-8 T.

3.4.3.1 How Magnetometers Work

The key element of most modern magnetometers is a Hall-effect sensor, a semiconductor device which produces a voltage proportional to the magnetic field intensity in a direction normal to the current flow. Typically three Hall-effect sensors are packaged together and arranged so that their sensitive axes are orthogonal. The three outputs of such a triaxial magnetometer are the components of the Earth's magnetic field intensity vector ${}^{B}\boldsymbol{m}^{\#}$ measured in the body frame $\{B\}$.

Yet again nature leads, and creatures from bacteria to turtles and birds are known to sense magnetic fields. The effect is particularly well known in pigeons and there is debate about whether or not humans have this sense. The actual biological sensing mechanism has not yet been discovered.

3.4.3.2 Estimating Heading

Consider an inertial coordinate frame $\{0\}$ with its z-axis vertically upward and its x-axis pointing toward *magnetic* north. The magnetic field intensity vector therefore lies in the xz-plane

$$ {}^{0}m = B \begin{pmatrix} \cos I \\ 0 \\ \sin I \end{pmatrix} $$

where B is the magnetic field intensity and I the inclination angle which are both known from Fig. 3.13. In a body-fixed frame $\{B\}$ at an arbitrary orientation expressed in terms of roll-pitch-yaw angles◀

We could use any 3-angle sequence.

$$ {}^{0}\xi_{B} = \mathcal{R}_{z}(\theta_{y}) \oplus \mathcal{R}_{y}(\theta_{p}) \oplus \mathcal{R}_{x}(\theta_{r}) $$

Edwin Hall (1855–1938) was an American physicist born in Maine. His Ph.D. research in physics at the Johns Hopkins University in 1880 discovered that a magnetic field exerts a force on a current in a conductor. He passed current through thin gold leaf and in the presence of a magnetic field normal to the leaf was able to measure a very small potential difference between the sides of the leaf. This is now known as the Hall effect. While it was then known that a magnetic field exerted a force on a current carrying conductor it was believed the force acted on the conductor not the current itself – electrons were yet to be discovered. He was appointed as professor of physics at Harvard in 1895 where he worked on thermoelectric effects.

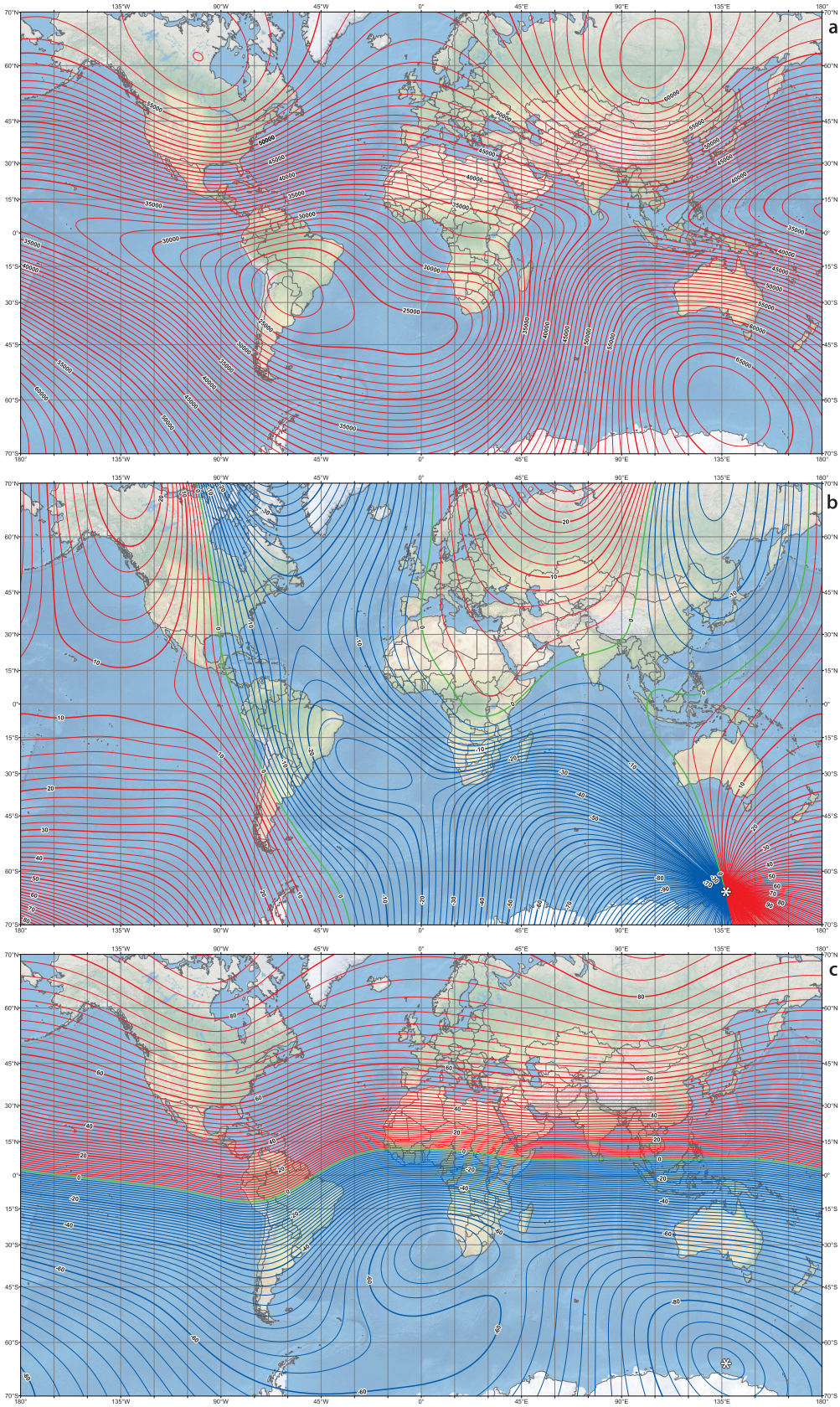


Fig. 3.13.
A predicted model of the Earth magnetic field parameters for 2015. **a** Magnetic field intensity (nT); **b** magnetic declination (degrees); **c** magnetic inclination (degrees). Magnetic poles indicated by *asterisk* (maps by NOAA/NGDC and CIRES http://ngdc.noaa.gov/geomag/WMM, published Dec 2014)

the magnetic field intensity will be

$$^{B}\boldsymbol{m} = \left(\ominus^{0}\xi_{B}\right)^{0} \bullet \boldsymbol{m} \tag{3.25}$$

The measured magnetic field intensity vector from the sensor in frame $\{B\}$ is

$$^{B}\boldsymbol{m}^{\#} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix}$$

and equating this with Eq. 3.25 we can solve for the yaw angle

$$\hat{\theta}_y = \tan^{-1} \frac{\cos\theta_p \left(m_z \sin\theta_r - m_y \cos\theta_r\right)}{m_x + B \sin I \sin\theta_p}$$

Many triaxial Hall-effect sensor chips also include a triaxial accelerometer for just this purpose.

assuming that the roll and pitch angles have been determined, perhaps using measured acceleration and Eq. 3.21.◀

Typically in vehicle navigation the *x*-axis points forward and the yaw angle is also called the heading angle.

We defined yaw angle as the orientation of the frame $\{B\}$ x-axis▶ with respect to *magnetic* north. To obtain the heading angle with respect to true-north we subtract the local declination angle

$$^{\text{tn}}\hat{\theta}_y = \hat{\theta}_y - D$$

Magnetometers are great in theory but problematic in practice. Firstly, our modern world is full of magnets and electromagnets. Buildings contain electrical wiring and robots themselves are full of electric motors, batteries and electronics. These all add to, or overwhelm, the local geomagnetic field. Secondly, many objects in our world contain ferromagnetic materials such as the reinforcing steel in buildings or the steel bodies of cars or ships. These distort the geomagnetic field leading to local changes in its direction. These effects are referred to respectively as hard- and soft-iron distortion of the magnetic field.◀

These can be calibrated out but the process requires that the sensor is rotated by 360 degrees.

3.4.4 Sensor Fusion

An inertial navigation system uses the devices we have just discussed to determine the pose of a vehicle – its position and its orientation. Early inertial navigation systems, such as shown in Fig. 2.15, used mechanical gimbals to keep the accelerometers at a constant attitude with respect to the stars using a gyro-stabilized platform. The acceleration measured on this platform is by definition referred to the inertial frame and simply needs to be integrated to obtain the velocity of the platform, and integrated again to obtain its position. In order to achieve accurate position estimates over periods of hours or days the gimbals and gyroscopes had to be of extremely high quality so that the stable platform did not drift, and the acceleration sensors needed to be extremely accurate.

The modern strapdown inertial measurement configuration uses no gimbals. The angular velocity, acceleration and magnetic field sensors are rigidly attached to the vehicle. The collection of inertial sensors is referred to as an inertial measurement unit or IMU. A 6-DOF IMU comprises triaxial gyroscopes and accelerometers while a 9-DOF IMU comprises triaxial gyroscopes, accelerometers and magnetometers.◀ A system that only determines attitude is called an attitude and heading reference system or AHRS.

Increasingly these sensor packages also include a barometric pressure sensor to measure changes in altitude.

The sensors we use, particularly the low-cost ones in phones and drones, are far from perfect. Consider any sensor value – gyroscope, accelerometer or magnetometer – the measured signal

$$x^{\#} = sx + b + \varepsilon$$

is related to the unknown true value x by a scale factor s, offset or bias b and random noise ε. s is usually specified by the manufacturer to some tolerance, perhaps $\pm 1\%$, and for a particular sensor this can be determined by some calibration procedure. Bias b is ideally equal to zero but will vary from device to device. Bias that varies over time is often called sensor drift. Scale factor and bias are typically both a function of temperature.

We assume a linear relationship but check the fine print in a datasheet to understand what a sensor really does.

Some sensors also exhibit cross-sensitivity. They may give a weak response to a signal in an orthogonal direction or from a different mode, quite commonly low-cost gyroscopes respond to vibration and acceleration as well as rotation.

In practice bias is the biggest problem because it varies with time and temperature and has a very deleterious effect on the estimated pose and position. Consider a positive bias on the output of a gyroscopic sensor – the output is higher than it should be. At each time step in Eq. 3.17 the incremental rotation will be bigger than it should be, which means that the orientation error will grow linearly with time.

If we use Eq. 3.22 to estimate the vehicle's acceleration then the error in attitude means that the measured gravitation acceleration is incorrectly canceled out and will be indistinguishable from *actual* vehicle acceleration. This offset in acceleration becomes a linear time error in velocity and a quadratic time error in position. Given that the pose error is already linear in time we end up with a cubic time error in position, and this is ignoring the effects of accelerometer bias. Sensor bias is problematic! A rule of thumb is that gyroscopes with bias stability of 0.01 deg h^{-1} will lead to position error growing at a rate of 1 nmi h^{-1} (1.85 km h^{-1}). Military grade systems have very impressive stability, for missiles <0.00002 deg h^{-1} which is in stark contrast to consumer grade devices which are in the range $0.01-0.2$ deg per *second*.

The effect of an attitude error is dangerous on something like a quadrotor. For example, if the estimated pitch angle is too high then the vehicle control system will pitch down by the same amount to keep the craft "level", and this will cause it to accelerate forward.

A simple approach to this problem is to estimate bias by leaving the IMU stationary for a few seconds and computing the average value of all the sensors. This value is then subtracted from future sensor readings. This is really only valid over a short time period because the bias is not constant.

A lot of hobby drones do this just before they take off.

A more sophisticated approach is to estimate the bias online, but to do this we need to combine information from different sensors – an approach known as sensor fusion. We rely on the fact that different sensors have complementary characteristics. Bias on angular rate sensors causes the attitude estimate error to grow with time, but for accelerometers it will only cause an attitude offset. However accelerometers respond to motion of the vehicle while good gyroscopes do not. Magnetometers provide partial information about roll, pitch and yaw, are immune to acceleration, but do respond to stray magnetic fields and other distortions. There are many ways to achieve this kind of fusion. A common approach is to use an estimation tool called an extended Kalman filter described in Appendix H. Given a full nonlinear mathematical model that relates the sensor signals and their biases to the vehicle pose and knowledge about the noise (uncertainty) on the sensor signals, the filter gives an optimal estimate of the pose and bias that best explain the sensor signals.

Our brain has an online mechanism to cancel out the bias in our vestibular gyroscopes. It uses the recent average rotation as the bias, based on the reasonable assumption that we do not undergo prolonged rotation. If we do, then that angular rate becomes the new normal so that when we stop rotating we perceive rotation in the opposite direction. We call this dizziness.

Here we will consider a simple but still very effective alternative called the explicit complementary filter. The rotation update step is performed using Eq. 3.17 but compared to Eq. 3.16 the incremental rotation is more complex

$$ {}^{B}\xi_{\Delta}\langle k\rangle = \mathrm{e}^{\left[{}^{B}\boldsymbol{\omega}^{\#}\langle k\rangle - \hat{\boldsymbol{b}}\langle k\rangle + k_P\boldsymbol{\sigma}_R\langle k\rangle\right]_{\times}\delta_t} \tag{3.26} $$

The key differences are that the estimated bias $\hat{\boldsymbol{b}}$ is subtracted from the sensor measurement and a term based on the orientation error $\boldsymbol{\sigma}_R$ is added. The estimated bias changes with time according to

$$ \hat{\boldsymbol{b}}\langle k+1\rangle \leftarrow \hat{\boldsymbol{b}}\langle k\rangle - k_I\boldsymbol{\sigma}_R\langle k\rangle \tag{3.27} $$

and also depends on the orientation error $\boldsymbol{\sigma}_R$. $k_P > 0$ and $k_I > 0$ are both well chosen constants.

The orientation error is derived from N *vector measurements* ${}^{0}\boldsymbol{v}_i^{\#}$

$$ \boldsymbol{\sigma}_R\langle k+1\rangle = \sum_{i=1}^{N} k_i\, {}^{0}\boldsymbol{v}_i \times {}^{0}\boldsymbol{v}_i^{\#}\langle k\rangle $$

where ${}^0\boldsymbol{v}_i$ is the known value of a vector signal in the inertial frame (for example gravitational acceleration) and

$$ {}^0\boldsymbol{v}_i^{\#}\langle k\rangle = {}^0\hat{\xi}_B\langle k\rangle \bullet {}^B\boldsymbol{v}^{\#}\langle k\rangle $$

is the value measured in the body-fixed frame and rotated into the inertial frame by the estimated orientation ${}^0\hat{\xi}_B$. Any error in direction between these vectors will yield a nonzero cross-product which is the axis around which to rotate one vector into the other. The filter uses this difference – the innovation – to improve the orientation estimate by feeding it back into Eq. 3.26. This filter allows an unlimited number of vectorial measurements ${}^0\boldsymbol{v}_i$ to be fused together; for example we could add magnetic field or any other kind of direction data such as the altitude and azimuth of visual landmarks, stars or planets.

The script

```
>> ex_tumble
```

provides simulated "measured" gyroscope, accelerometer and magnetometer data organized as columns of the matrices `wm`, `gm` and `mm` respectively and all include a fixed bias. Corresponding times are given by elements of the vector `t`. Firstly we will repeat the example from page 81 but now with sensor bias

```
>> attitude(1) = UnitQuaternion();
>> for k=1:numcols(wm)-1
      attitude(k+1) = attitude(k) .* UnitQuaternion.omega( wm(:,k)*dt );
   end
```

To see the effect of bias on the estimated attitude we will compare it to the true attitude `truth` that was also computed by the script. As a measure of error we plot the angle between the corresponding unit quaternions in the sequence

```
>> plot(t, angle(attitude, truth), 'r' );
```

which is shown as the red line in Fig. 3.14a. We can clearly see growth in angular error over time. Now we implement the explicit complementary filter with just a few extra lines of code

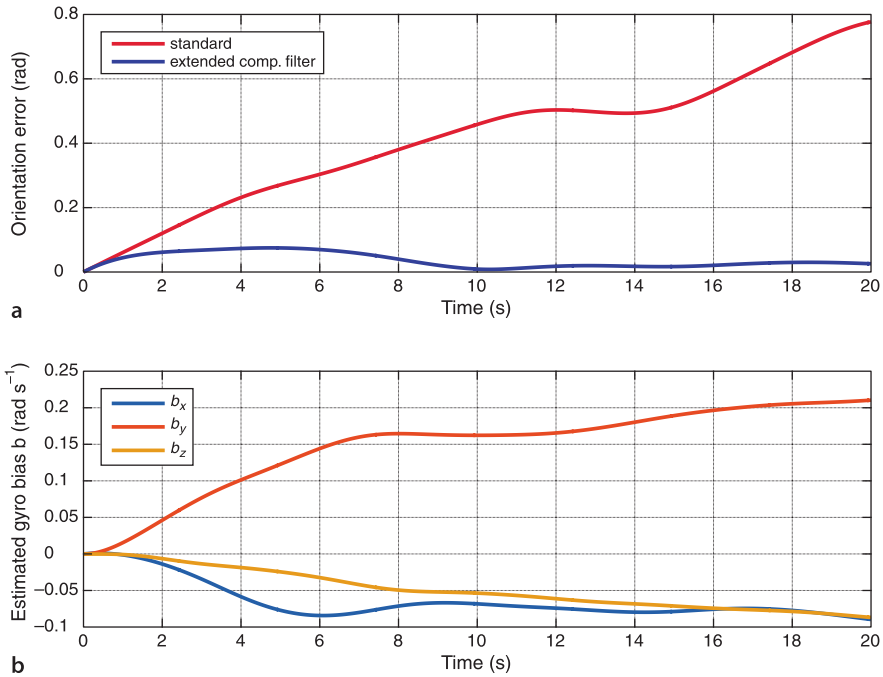


Fig. 3.14.
a Effect of gyroscope bias on naive INS (red) and explicit complementary filter (blue); **b** estimated gyroscope bias from the explicit complementary filter

```
>> kI = 0.2; kP = 1;
>> b = zeros(3, numcols(w));
>> attitude_ecf(1) = UnitQuaternion(); b = [0 0 0]';
>> for k=1:numcols(wm)-1
      invq = inv( attitude_ecf(k) );
      sigmaR = cross(gm(:,k), invq*g0) + cross(mm(:,k), invq*m0);
      wp = wm(:,k) - b(:,k) + kP*sigmaR;
      attitude_ecf(k+1) = attitude_ecf(k) .* UnitQuaternion.omega( wp*dt );
      b(:,k+1) = b(:,k) - kI*sigmaR*dt;
   end
```

and plot the angular difference between the estimated and the attitude as a blue line

```
>> plot(t, angle(attitude_ecf, truth), 'b' );
```

Bringing together information from multiple sensors has checked the growth in attitude error, despite all the sensors having a bias. The estimated gyroscope bias is shown in Fig. 3.14b and we can see the bias estimates converging on their true value.

3.5 Wrapping Up

In this chapter we have considered pose that varies as a function of time from several perspectives.

Firstly we took a calculus perspective and showed that the temporal derivative of an orthonormal rotation matrix or a quaternion is a function of the angular velocity of the body – a concept from mechanics. The skew-symmetric matrix appears in the rotation matrix case and we should no longer be surprised about this given its intimate connection to rotation via Lie group theory. We then looked at finite time differences as an approximation to the derivative and showed how these lead to computationally cheap methods to update rotation matrices and quaternions given knowledge of angular velocity. We also discussed the dynamics of moving bodies that translate and rotate under the influence of forces and torques, inertial and noninertial reference frames and the notion of fictitious forces.

The second perspective was to create motion – a sequence of poses, a trajectory, that a robot can follow. An important characteristic of a trajectory is that it is *smooth* – the position and orientation changes smoothly with time. We started by discussing how to generate smooth trajectories in one dimension and then extended that to the multi-dimensional case and then to piecewise-linear trajectories that visit a number of intermediate points. Smoothly varying rotation was achieved by interpolating roll-pitch-yaw angles and quaternions.

With all this under our belt we were then able to tackle an application, the important problem of inertial navigation. Given imperfect measurements from sensors on a moving body we are able to estimate the pose of that moving body.

Further Reading

The earliest work on manipulator Cartesian trajectory generation was by Paul (1972, 1979) and Taylor (1979). The multi-segment trajectory is discussed by Paul (1979, 1981) and the concept of segment transitions or blends is discussed by Lloyd and Hayward (1991). These early papers, and others, are included in the compilation on Robot Motion (Brady et al. 1982). Polynomial and LSPB trajectories are described in detail by Spong et al. (2006) and multi-segment trajectories are covered at length in Siciliano et al. (2009) and Craig (2005).

The book *Digital Apollo* (Mindell 2008) is a very readable story of the development of the inertial navigation system for the Apollo Moon landings. The article by Corke et al. (2007) describes the principles of inertial sensors and the functionally equivalent sensors located in the inner ear of mammals that play a key role in maintaining balance.

There is a lot of literature related to the theory and practice of inertial navigation systems. The thesis of Achtelik (2014) describes a sophisticated extended Kalman filter for estimating the pose, velocity and sensor bias for a small flying robot. The explicit complementary filter used in this chapter is described by Hua et al. (2014). The recently revised book Groves (2013) covers inertial and terrestrial radio and satellite navigation and has a good coverage of Kalman filter state estimation techniques. Titterton and Weston (2005) provides a clear and concise description of the principles underlying inertial navigation with a focus on the sensors themselves but is perhaps a little dated with respect to modern low-cost sensors. Data sheets on many low-cost inertial and magnetic field sensing chips can be found at https://www.sparkfun.com in the Sensors category.

Exercises

1. Express the incremental rotation ${}^{B}\boldsymbol{R}_{\Delta}$ as an exponential series and verify Eq. 3.7.
2. Derive the unit-quaternion update equation Eq. 3.8.
3. Make a simulation with a particle moving at constant velocity and a rotating reference frame. Plot the position of the particle in the inertial and the rotating reference frame and observe how the motion changes as a function of the inertial frame velocity.
4. Redo the quaternion-based angular velocity integration on page 81 using rotation matrices.
5. Derive the expression for fictitious forces in a rotating reference frame from Sect. 3.2.3.
6. At your location determine the magnitude and direction of centripetal acceleration you would experience. If you drove at 100 km h^{-1} due east what is the magnitude and direction of the Coriolis acceleration you would experience? What about at 100 km h^{-1} due north? The vertical component is called the Eötvös effect, how much lighter does it make you?
7. For a `tpoly` trajectory from 0 to 1 in 50 steps explore the effects of different initial and final velocities, both positive and negative. Under what circumstances does the quintic polynomial overshoot and why?
8. For a `lspb` trajectory from 0 to 1 in 50 steps explore the effects of specifying the velocity for the constant velocity segment. What are the minimum and maximum bounds possible?
9. For a trajectory from 0 to 1 and given a maximum possible velocity of 0.025 compare how many time steps are required for each of the `tpoly` and `lspb` trajectories?
10. Use `animate` to compare rotational interpolation using quaternions, Euler angles and roll-pitch-yaw angles. Hint: use the quaternion `interp` method and `mtraj`.
11. Repeat the example of Fig. 3.7 for the case where:
    a) the interpolation does *not* pass through a singularity. Hint – change the start or goal pitch angle. What happens?
    b) the final orientation is at a singularity. What happens?
12. Develop a method to quantitatively compare the performance of the different orientation interpolation methods. Hint: plot the locus followed by $\hat{\boldsymbol{z}}$ on a unit sphere.
13. For the `mstraj` example (page 75)
    a) Repeat with different initial and final velocity.
    b) Investigate the effect of increasing the acceleration time. Plot total time as a function of acceleration time.
14. Modify `mstraj` so that acceleration limits are taken into account when determining the segment time.
15. There are a number of iOS and Android apps that display sensor data from gyros, accelerometers and magnetometers. You could also use MATLAB, see http://mathworks.com/hardware-support/iphone-sensor.html. Run one of these and explore how the sensor signals change with orientation and movement. What happens when you throw the phone into the air?

16. Consider a gyroscope with a 20 mm diameter steel rotor that is 4 mm thick and rotating at 10 000 rpm. What is the magnitude of h? For an angular velocity of 5 deg s^{-1}, what is the generated torque?
17. Using Eq. 3.15 can you explain how a toy gyroscope is able to balance on a single point with its spin axis horizontal? What holds it up?
18. A triaxial accelerometer has fallen off the table, ignoring air resistance what value does it return as it falls?
19. Implement the algorithm to determine roll and pitch angles from accelerometer measurements.
    a) Devise an algorithm to determine if you are in portrait or landscape orientation?
    b) Create a trajectory for the accelerometer using `tpoly` to generate motion in either the x- or y-direction. What effect does the acceleration along the path have on the estimated angle?
    c) Calculate the orientation using quaternions rather than roll-pitch-yaw angles.
20. You are in an aircraft flying at 30 000 feet over your current location. How much lighter are you?
21. Determine the Euler angles as a function of the measured acceleration. If you have the Symbolic Math Toolbox™ you might like to use that.
22. Determine the magnetic field strength, declination and inclination at your location. Visit the website http://www.ngdc.noaa.gov/geomag-web.
23. Using the sensor reading app from above, orient the phone so that the magnetic field vector has only a z-axis component, where is the magnetic field vector with respect to your phone?
24. Using the sensor reading app from above log some inertial sensor data from a phone while moving it around. Use that data to estimate the changing attitude or full pose of the phone. Can you do this in real time?
25. Experiment with varying the parameters of the explicit complementary filter on page 90. Change the bias or add Gaussian noise to the simulated sensor readings.