The Mathematics of Havok's Solver

Contents

Notation

Vectors

$\mathbb{R}$	The real number field.
x	A column vector in $\mathbb{R}^n$, usually in $\mathbb{R}^3$.
x^T	The transpose of vector x.
x_i	The i^{th} element of x.
$x.y$	The dot product of vectors x and y.
$x \times y$	The cross product of vector x and y.
$\parallel x \parallel$	The length of vector x in $\mathbb{R}$.
sx, $s.x$	The multiplication of vector x by the real scalar$s$.
$\text{Proj}_x y$	The orthogonal projection of vector y on vector x.
$\text{Rep}_B x$	The representation of vector x in basis B.

Matrices

$\mathcal{M}_{m\times n}$	The vector space of $m \times n$ real matrices.
BA	The product of matrices A and B. (A followed by B).
$E_{n\times n}$, E,I	The $n \times n$ identity matrix.
$Z_{m\times n}$, Z	The $m \times n$ zero matrix.

Quaternions

$SO(3)$	The group of rotations about the origin in Euclidean space $\mathbb{R}^3$.
$\mathbb{H}$	The quaternion algebra.
q	The $\mathbb{R}^4$ vector representation of a quaternion.
$\begin{pmatrix} q_s \\ q_v \end{pmatrix}$	A notation for the quaternion which splits it into a scalar q_s and a vector q_v
pq	The product of the quaternions p and q. (q followed by p).
q^c, q^*	The conjugate of quaternion q.

Time

t	Absolute time.
h	A time step, a finite time interval.
a^t, $a[t]$	The value of a at time.
a^k,$a[k]$	The value of a at at discrete time $k.h$.
$\dot{a} = \frac{da}{dt}$	The time derivative of a.

Rigid bodies

p	The cartesian coordinates of the center of mass of a body in $\mathbb{R}^3$.
r	The rotation of a body in $SO(3)$ parameterized as a unit quaternion.

$\dot{p}$	The linear velocity of a body in $\mathbb{R}^3$.
w	The angular velocity of a body in $\mathbb{R}^3$.
$\overline{w}$	The angular velocity of a body in the body's (local) frame of reference.
$\begin{pmatrix} p \\ r \end{pmatrix}$, q	The six degrees of freedom configuration vector of a body in $\mathbb{R}^3 \times SO(3)$, also simply called the position.
$\begin{pmatrix} \dot{x} \\ \dot{r} \end{pmatrix}$, $\dot{q}$	The time derivative of $\begin{pmatrix} x \\ r \end{pmatrix}$ in $\mathbb{R}^3 \times \mathbb{R}^4$.
$\begin{pmatrix} \dot{x} \\ w \end{pmatrix}$, v	The velocity vector of a body in $\mathbb{R}^3 \times \mathbb{R}^3$.
m	The mass of a body.
$I(t)$	The rotational inertia matrix of a body in $\mathcal{M}_{3\times3}$ relative to the center of mass.
$\begin{pmatrix} m.E_{3\times3} & Z \\ Z & I(t) \end{pmatrix}$, $M(t)$	The rigid body mass matrix in $\mathcal{M}_{6\times6}$.
$\begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix}$, $\bar{I}$	The body frame rotational inertia matrix in $\mathcal{M}_{3\times3}$ with I_x, I_y, I_z as the prinicpal moments of intertia. Unlike $I(t)$, $\bar{I}$ is not time dependent.
$C_{WB}(t)$, C_B^{-1}, C_B^T	The change of basis matrix from world to body frame of reference in $\mathcal{M}_{3\times3}$.
$C_{BW}(t)$, C_B, B	The change of basis matrix from body to world frame of reference in $\mathcal{M}_{3\times3}$.
f, τ	Force and torque.
$\cdots_i$	In a system of bodies, the subscript $\cdots_i$ relates $\cdots$ to body i.

Part I
Havok's Solver

1 Havok's Constraints

1.1 Integration Scheme

In this section we try to characterize the integration scheme used in the Havok engine. For constraints that use Lagrange multipliers, the update rule for the velocity vector is

$$v[t+h] = v[t] + M^{-1}J^T\lambda_G.$$

From the integration scheme point of view, this means that

$$\dot{v}(t) = \frac{1}{h}(M^{-1}J^T\lambda_G).$$

Now $\dot{v}(t)$[1] is a function of time, and so are M, J and λ_G. The latter is a function of B, J, M, F_e. We assume F_e be time independent, while all of B, M and J are usually a function of the position q which makes them time dependent.
In Havok's engine, position is always updated using $q[t+h] = q[t]+h.Sv[t+h]$; this rules out Explicit Euler as a characterization. In contrast, the velocity update rule varies. For some constraints, all structures used to compute $\dot{v}(t)$ are evaluated at time t (Semi-implicit). For others it is a mix of evaluations at t and approximate evaluations at $t+h$, resulting in a hybrid integrator that one could name "almost implicit".

[1]$\dot{v}(t)$ merely indicates that $\dot{v}$ is a function of time, while $\dot{v}[t]$ is the value of $\dot{v}$ time t. (see Notation)

1.2 Sub-Stepping

1.2.1 Approximation of Quaternion Composition

1.3 Joints

1.3.1 Linear 1D

1.3.2 Angular 1D

1.4 User

1.5 Contacts

1.6 Destruction

1.7 Chain

1.8 Rope

Part II
Fundamentals

1 Rigid Body Dynamics

Equations of motion

$\dot{r} = \frac{1}{2}Q(w)r = \frac{1}{2}rQ(\overline{w})$

The time derivative of a rigid body's quaternion rotation,

where $Q(w)$ is the quaterion $\begin{pmatrix} 0 \\ w \end{pmatrix}$.

$$\mathcal{B}(q) = \begin{pmatrix} -q_0 & -q_1 & -q_2 \\ q_3 & q_2 & -q_1 \\ -q_2 & q_3 & q_1 \\ q_2 & -q_0 & q_3 \end{pmatrix}$$

A helper matrix for quaternion calculus in $\mathcal{M}_{4\times 3}$.

$\dot{r} = \frac{1}{2}\mathcal{B}(r)Q(w)$

$$\dot{q} = \begin{pmatrix} E_{3\times 3} & Z \\ Z & \frac{1}{2}\mathcal{B}(r) \end{pmatrix} \begin{pmatrix} \dot{p} \\ w \end{pmatrix} = Sv$$

The linear relation between the time derivative of q to its velocity vector v. The S matrix in $\mathcal{M}_{7\times 6}$.

$$\begin{pmatrix} f \\ \tau \end{pmatrix} = \begin{pmatrix} m\ddot{p} \\ I\dot{w} + w \times Iw \end{pmatrix}$$

The Newton-Euler (N-E) equations of motion for a single body.

$$\begin{pmatrix} f \\ \tau \end{pmatrix} \simeq \begin{pmatrix} m\ddot{p} \\ I\dot{w} \end{pmatrix} = M.\begin{pmatrix} \ddot{p} \\ \dot{w} \end{pmatrix}$$

Approximate Newton-Euler (A-N-E) equations. Inertial torque is ignored and consequently, angular velocity is conserved instead of momentum.

Change of reference frame

$I(t) = C_{BW}(t)\,\overline{I}\,C_{WB}(t) = B\overline{I}B^{-1}$

$\bar{z} = B^{-1}(z)$

Where z is a vector (in $\mathbb{R}^3$) in world frame,$\overline{z}$ is that vector in body frame, e.g: w and $\bar{w}$.

$\bar{M} = B^{-1}(M)B$

Where M is a matrix (in $\mathcal{M}_{3\times 3}$) in world frame, $\overline{M}$ is that matrix in body frame, e.g: I and $\bar{I}$.

1.1 Rigid body Ordinary Differential Equations

The Approximate Newton-Euler equations can be expanded to two coupled first order ordinary differential equations (ODE):

$$\begin{aligned}\dot{q} =& Sv\\ \dot{v} =& M^{-1}\begin{pmatrix} f \\ \tau \end{pmatrix},\end{aligned}$$

and this extends naturally to a system of bodies.

2 Simulation of Rigid Body Dynamics

2.1 Numerical Integration Schemes

In this section we summarize common techniques for numerically integrating the rigid body ODEs; which means computing a new state for the next discrete time step $t + h$. We will illustrate each scheme by its update rule and briefly comment on it.

For all schemes, both velocity and position are discretized. Roughly speaking, an ODE of the form $\dot{x} = y$ becomes

$$\frac{x[t+h] - x[t]}{h} = y(t),$$

which leads to the update rule

$$x[t+h] = x[t] + h.y(t).$$

Explicit Euler

$$\begin{aligned}v[t+h] =& v[t] + h.\dot{v}[t]\\ q[t+h] =& q[t] + h.Sv[t].\end{aligned}$$

Explicit Euler integration is known for its instability for even the simplest[2] systems.

[2]One such system is the *simple harmonic oscillator*.

Implicit Euler

$$\begin{aligned} v[t+h] =& v[t] + h.\dot{v}[t+h] \\ q[t+h] =& q[t] + h.Sv[t+h]. \end{aligned}$$

Implicit Euler has strong stability, but it comes at the cost of visibly distorting the physical behavior. Apart from that, evaluating $\dot{v}[t+h]$ proves to be a challenge.

Semi-implicit Euler

$$\begin{aligned} v[t+h] =& v[t] + h.\dot{v}[t] \\ q[t+h] =& q[t] + h.Sv[t+h]. \end{aligned}$$

Semi-implicit[3] Euler has adequate stability, good physical behavior, and is easy to compute.

Implicit midpoint

$$\begin{aligned} v[t+{}^{h}\!/_{2}] =& v[t] + {}^{h}\!/_{2}.\dot{v}[t] \\ q[t+{}^{h}\!/_{2}] =& q[t] + {}^{h}\!/_{2}.Sv[t] \\ v[t+h] =& v[t] + h.\dot{v}[t+{}^{h}\!/_{2}] \\ q[t+h] =& q[t] + h.Sv[t+{}^{h}\!/_{2}]. \end{aligned}$$

The implicit midpoint scheme combines the strong stability of implicit Euler with the good physical behavior of Semi-explicit Euler.

2.2 Linear Time Dynamics Using Lagrange Multipliers

To derive this common formulation, we consider a system of n bodies subject to m velocity constraints. We require that the system satisfies both the approximate Newton-Euler equations discretized to first order, and the velocity constraints equations.
We assume that we are dealing with Jacobian constraints, so

$$Jv = B,$$

[3]Also called *Symplectic Euler.*

where J is in $\mathcal{M}_{s\times 6n}$ and β in $\mathbb{R}^m$. Let us call the aggregate of the individual total forces acting on each body the total force vector f.

$$f = \begin{pmatrix} f_1 \\ \tau_1 \\ . \\ . \\ f_n \\ \tau_n \end{pmatrix}$$

It is the sum of two other force vectors: the internal constraint forces f_c and the external forces f_e:

$$f = f_c + f_e$$

The constraint forces do no work because they are internal to the system which implies that

$$f_c = J^T \lambda_f,$$

where λ_f is a vector of Lagrange multipliers in $\mathbb{R}^m$. Acceleration formulations exhibit known solvability problems[4], so before discretizing we switch to an impulse based formulation with g as an impulse:

$$\begin{aligned} g =& f.h \\ g_c =& J^T \lambda_g. \end{aligned}$$

We discretize the system by evaluating it at two consecutive discrete time steps t^1,t^2, which results in a linear system.

$$\begin{aligned} v^2 =& J^{-1}B \\ M(\frac{v^2 - v^1}{h}) =& f_c^1 + f_e^1 \\ M(\frac{v^2 - v^1}{h}) =& \frac{1}{h}(J^T\lambda_g + g_e) \\ M(J^{-1}B - v^1) =& (J^T\lambda_g + g_e) \\ J^{-1}B - v^1 =& M^{-1}J^T\lambda_g + M^{-1}g_e \\ M^{-1}J^T\lambda_G =& J^{-1}B - v^1 - M^{-1}g_e \\ JM^{-1}J^T\lambda_G =& B - J(v^1 + M^{-1}g_e). \end{aligned}$$

[4]In particular due to the Painlevé paradox.

For convenience, we set

$$\begin{aligned}\Gamma =& JM^{-1}J^T\\ v_e =& M^{-1}g_e\\ \eta =& B - J(v^1 + v_e),\end{aligned}$$

and we call Γ the inverse virtual mass matrix,η the right-hand-side (rhs) of the Lagrange multiplier equation., while v_e is the external velocity vector at the beginning of the time step. This gives the following concise version of the equation

$$\Gamma\lambda_g = \eta.$$

The matrix Γ is positive semi-definite, so if the constraints are consistent, it is invertible.

2.3 The Sequential Impulse Method

3 Constraints

A rigid body constraint C is described with a constraint function

$$C(t, q, \dot{q}),$$

which is then turned into a position constraint equation $C(t, q, \dot{q}) = 0$ or inequality $C(t, q, \dot{q}) \geq 0$.
When the equation holds, its time derivatives $\dot{C} = 0$ and $\ddot{C} = 0$ hold as well[5] and they are called the velocity and acceleration constraint equation respectively. Constraints that are described by equations are called bilateral, while the inequality constraints are called unilateral. Constraints that are bilateral and independent of $\dot{q}$ are called holonomic:

$$C(t, q) = 0,$$

and all others are called nonholonomic.

3.1 Constraint Jacobians

For constraints that are of the simple form $C(q)$, The velocity formulation takes on a special form

$$\frac{dC}{dt} = \frac{\partial C}{\partial q}\frac{dq}{dt} = \frac{\partial C}{\partial q}Sv = Jv,$$

[5]The same applies to inequalities.

where J is called the Jacobian matrix and has dimensions $1 \times 6n$ for a constraint involving n bodies.

It is also useful to consider as "Jacobian constraints" the bilateral constraints of the form $C(q) = l(t)$, $l(t) = Bt + A$. Here, the velocity equation becomes

$$\frac{dC}{dt} = Jv = B,$$

where B is called the constraint bias.

3.2 Inequality Constraints

3.3 Correction and Relaxation

3.3.1 Baumgarte Stabilization

The Baumgarte stabilization method consists in trying to satisfy all three constraint formulations instead of just one (e.g: velocity) using

$$\ddot{C} + \alpha\dot{C} + \beta C = 0,$$

where $\alpha, \beta > 0$.
For a velocity based simulation we can use the simpler $\dot{C} + \beta C = 0$. When the constraints are of the simple form $C(q) = 0$, we get $Jv = -\beta C$, which we can plug into the the bias of the Lagrange multipliers equation, so:

$$B = -\beta C(q).$$

The consequence of using this bias is that the resulting impulse not only acts to satisfy $\dot{C}$ but also partially combat the inevitable positional drift.

3.3.2 The Error Reduction Parameter (ERP)

When using Baumgarte stabilization, experience shows that simply setting $\beta = 1$ can destabilize the simulation. Instead, values less than one are used, but a methodical way to find the optimal setting (if possible at all) has so far been elusive. On the other hand, understanding the effect of setting $\beta = k < 1$ on the drift itself is easier.

In the non-discretized case, the solution for the differential equation $\dot{C}+\beta C = 0$ is well known: $C(t) = C(0)e^{-\beta t}$, with $C(t)$ being the drift at time t and β acting as a damping parmeter.

For the discrete case, we have the recurrence relation $C(k+1) = C(k)(1-\beta)$, and so $C(k+1) = C(0)(1-\beta)^k$. For convergence, β has to be within the range $(0, 2)$. A more intuitve parameter than β is ERP: the error reduction within one second. Per example, an ERP of 0.75 would mean that we wish to reduce the error by 75% within one second. Since one second elapses in $n = 1/h$ steps, we get:

$$\begin{aligned}(1-ERP).C(0) =& C(0).(1-\beta)^n \\ \beta =& 1 - \sqrt[n]{1-ERP}.\end{aligned}$$

3.3.3 Predicting Error

While Baumgarte stabilization uses the positional error $C(t)$ at time t to compute the constraint impulses, it is also possible to predict the error at time $t+h$ and use that instead. It makes sense to combine this with an approximation of the whole state at $t+h$. ??????? Having approximated $C(t+h)$ we linearize $\dot{C}$ and get

$$\dot{C}(t+h) \approx \frac{C(t+h) - C(t)}{h}.$$

Using the resulting $\dot{C}(t+h)$ instead of the more common $\dot{C}(t)$ turns a symplectic solver into an approximate implicit one. ??????

3.3.4 Post Stabilization

3.3.5 Constraint Force Mixing (CFM)

3.3.6 Scaling the Velocity Error

4 Successive Over Relaxation

5 Projected Gauss-Seidel with S.O.R

Formally, The complementary problem (MLCP) is the following:

$$r = \Gamma\lambda - \eta$$

$$\lambda^- \leq \lambda \leq \lambda^+$$

$$r_i = 0 \leftrightarrow \lambda_i^- \leq \lambda_i \leq \lambda_i^+$$

6 Rotation

Every rotation about the origin of three dimensional Euclidean space $\mathbb{R}^3$ is by definition a transformation that preserves distance and handeness. Rotation is also assumed to leave the origin unchanged to distinguish it from translation. The motivation for this definition comes from what we naturally (geometrically) understand by it. The condition of preserving handedness is necessary to distinguish rotation from reflection (what we see when we look into a mirror) which is also distance preserving[6]. Additionally, the composition of two rotations cannot result in anything else but yet another rotation. The abstract algebraic stucture called the rotation group, denoted by SO(3), captures these properties and is the most general way to mathematically desribe three dimensional rotation without commiting to a specific parameterization. SO(3) abbreviates 'Special Orthogonal Group in 3 Dimensions', referring to orthogonal 3×3 matrices with unit determinant.

6.1 Parameterization of Rotation

SO(3) is a manifold, and can be charted in more than one way. The multiple charts set up rival coordinate systems none of which is the most natural or obvious one. In what follows, we summarize the most relevant ones.

6.1.1 Rotation Matrices

Rotations are distance preserving by definition, they are therefore linear maps and can be represented by matrices[7]. For our purposes, sticking to $\mathbb{R}^3$ and Cartesian coordinates, we only consider rotations represented with respect to the natrual basis E_3 and it then follows[7] that Rotation matrices are determined by nine direction cosines:

$$R = \begin{pmatrix} r_1[0] & r_2[0] & r_3[0] \\ r_1[1] & r_2[1] & r_3[1] \\ r_1[2] & r_2[2] & r_3[2] \end{pmatrix},$$

where r_i is the rotated vector e_i of E_3, and $r_i[j]$ is the cosine of the angle between r_i and e_j. Furthermore, they are orthonormal with a determinant of one, hence preserving signed volume. Because of these restrictions, only three of the nine cosines are linearly independent and that is why parameterizations with lower dimensionality such as the ones we consider below are possible.

[6]This kinship results in reflection also being called *improper rotation* as opposed to rotation *proper.*

[7]See the appendix on rotations.

6.1.2 Axial Rotation

According to Euler's rotation theorem, any rotation in three dimensional space admits an axis[7]. The defining feature of this axis is that the rotation leaves it unchanged. It turns out that to fully describe the rotation, all that is needed is to associate the axis with a unique angle, leading to an *axis and angle* parameterization.Using a unit length vector for the axis, we can multiply it by the angle to obtain an axial rotation vector in $\mathbb{R}^3$ which is valid if when the angle is zero since in that case, there is no rotation involved to begin with. It is however important to note that the resulting mathematical structure is merely a *pseudo vector* in the sense that adding two of them does not necessarily result in another representing the composition of the two involved rotations.
Finally, we note that to convert (back and forth) between a rotation matrix and axial rotation, the Rodrigues's rotation formula[7] proves invaluable.

6.1.3 Sequential Rotations (Euler Angles)

It was once more Euler who showed that rotation can be parametrized with merely three numbers[1]. The essense of such a paramterization is twofold. Firstly, that given a known axis, angle alone determines a rotation, and secondly, that any rotation can be achieved with a fixed sequence of three *elementary* ones, each about a known axis. It hence falls out that the three correpsonding angles fully determine the rotation.
There is an intuitive way to find such a sequence, using a cartesian frame (x, y, z) - or (thumb, index, middle finder) to help mental visualization - to be rotated to an arbitrary target frame (x', y', z'). The idea is that once a principal axis (e.g x) is aligned to its target, only one additional elementary rotation suffices to align the remaining axes; a consequence of preservation of orthogonality and handedness. Aligning the principal axis is easily achieved with two rotations[1].

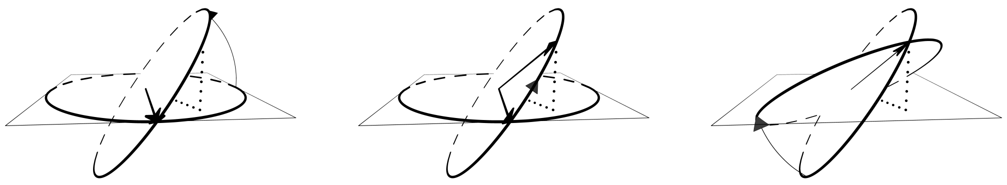

Figure 1: The three steps of a sequential rotation. The first two steps show the evolution of the x axis toward x' whose projection is shown. First by rotating the plane (x, y) such that it contains both x and x', and then rotating in that plane. The third step rotates both remaining axes about x'.

6.1.4 Rotation Quaternions

6.2 The Exponential Map

6.3 Rate of Rotation

6.3.1 Angular Velocity

6.3.2 Quaterion Calculus

6.3.3 Rates of Sequential Rotations

Part III
Appendix

1 Rotations

1.1 A Distance Preserving Transformation is Linear

Proof. Let $t : \mathbb{R}^n \to \mathbb{R}^n$ be a length preserving transformation that maps the vector zero to itself . By definition t is linear iff. $t(au + bv) = at(a) + bt(v)$ for all $u, v \in \mathbb{R}^n$, $a, b \in \mathbb{R}$. By contradiction, assume that t is not linear, but length preserving. In other words $\| t(au) \|=\| au \|$, $\| t(bv) \|=\| bv \|$,

$\| t(au+bv) \| = \| au+bv \|$ but $t(au+bv) \neq at(u)+bt(v)$. This leads to

$$\begin{aligned}
t(au+bv) &\neq at(u)+bt(v) \\
\| t(au+bv) \| &\neq \| at(u)+bt(v) \| \\
\| t(au+bv) \|^2 &\neq \| at(u)+bt(v) \|^2 \\
\| t(au+bv) \|^2 &\neq (a.t(u))^2 + (bt(v))^2 + 2.ab.t(u).t(v) \\
\| t(au+bv) \|^2 &\neq \| au \|^2 + \| bv \|^2 + 2.ab.t(u).t(v).
\end{aligned}$$

But $\| t(au+bv) \| = \| au+bv \|$ implies that $\| t(au+bv) \|^2 = \| au \|^2 \| bv \|^2 + 2.ab.(u.v)$. Subtracting both equations we get

$$\begin{gathered}
2.ab.(u.v) \neq 2.ab.t(u).t(v) \\
u.v \neq t(u).t(v).
\end{gathered}$$

Geometrically, this is equivalent to saying that angle cosines are not invariant under the considered maps, which of course is false and we will complete the proof by showing just that.
So far we have treated t as length preserving and worked with the consequences. We now note that there is a difference between length and distance preservation: A transformation that merely preserves vector lengths could map its whole domain to a set of colinear vectors of different (but preserved) lengths, in effect collapsing shapes to lines. But by distance preservation we further require that the distances between the 'tips' of vectors also remain unchanged; this means that for two vectors u and v, the difference $u-v$ must have equal length to $t(u)-t(v)$. This reduces to angle cosine invariance:

$$\begin{gathered}
\| u-v \|^2 = \| t(u)-t(v) \|^2 \\
\| u \|^2 + \| v \|^2 - 2.(u.v) = \| t(u) \|^2 + \| t(v) \|^2 - 2.(t(u).t(v)) \\
\| u \|^2 + \| v \|^2 - 2.(u.v) = \| u \|^2 + \| v \|^2 - 2.(t(u).t(v)) \\
u.v = t(u).t(v),
\end{gathered}$$

which shows that $u.v \neq t(u).t(v)$ was false and t is therefore linear. QED

1.2 Properties of Rotation Matrices

Represented with respect to the natural basis E_3, rotation matrices have column vectors assembled from direction cosines, are orthonormal and have a determinant of one.

Proof. A linear map is determined by its action on a basis so for a rotation map r, the matrix representation is

$$R = \left(\operatorname{Rep}_{E_3}(r(e_1)) \quad \operatorname{Rep}_{E_3}(r(e_2)) \quad \operatorname{Rep}_{E_3}(r(e_3)) \right).$$

Because E_3 is orthogonal, the components of a vector's representation are its projections on the e_i vectors:

$$\text{Rep}_{E_3}(r(e_i)) = \begin{pmatrix} r(e_i).e_1 \\ r(e_i).e_2 \\ r(e_i).e_3 \end{pmatrix},$$

and R takes the form

$$\begin{pmatrix} r(e_1).e_1 & r(e_2).e_1 & r(e_3).e_1 \\ r(e_1).e_2 & r(e_2).e_2 & r(e_3).e_2 \\ r(e_1).e_3 & r(e_2).e_3 & r(e_3).e_3 \end{pmatrix}.$$

Since all vectors involved are unit length, these projections are simply the cosines of the related angles; for this reason, the entries of R are called the *direction cosines*.
To obtain a more succint form of R we simply set $r(e_i) = r_i$ and write r_i as $\begin{pmatrix} r_i[x] & r_i[y] & r_i[z] \end{pmatrix}^T$ to get

$$R = \begin{pmatrix} r_1[x] & r_2[x] & r_3[x] \\ r_1[y] & r_2[y] & r_3[y] \\ r_1[z] & r_2[z] & r_3[z] \end{pmatrix},$$

where r_i is the rotated vector e_i of E_3.
Since the rotated vectors $r(e_i)$ keep their lengths and unsigned angles[8], they remain orthogonal unit vectors and R is orthonormal. QED

In general, the transpose of an orthormal matrix is equal to its inverse. This is clear since for an orthormal matrix M with vectors m_i, the product $P = M.M^T$ has entries $p_{ij} = m_i.m_j$ but $m_i.m_j = 0$ when $i \neq j$ by orthogonality and $m_i.m_j = 1$ when $i = j$ since $m_i.m_i = \| m_i \|^2 = 1$.
P is therefore the identity matrix and by the definition of matrix inverses, we have that $M^T = M^{-1}$. From this it follows that $1 = \det(M.M^T) = \det(M.M) = \det(M)^2$, and so $\det(M) = \pm 1$.
It turns out that, as far as their matrix representations are concerned, the distinction between rotation and reflection is the sign of the determinant in the following way: A distance preserving map with representation R is a rotation if and only if

$$\det(R) = 1,$$

and a reflection otherwise. A proof [2] is beyond the scope of this document but intuitively, the preservation handedness ultimately relates to permutations of E_3 basis vectors, reducing (like geometric handedness) to two cases

[8]See: A Distance Preserving Transformation is Linear.

when we consider them relatively to e_1: (e_1, e_2, e_3) which is by convention labelled right handed, and (e_1, e_3, e_2) or $(e_1, e_2, -e_3)$ labelled left handed. This labelling is arbitrary and what really matters is the fact that a matrix with determinant of one does not act through an odd number of permutations. Futhermore, it is easy to check that

$$\det(R) = r_3.(r_1 \times r_2)$$

which provides a geometrically intuitive formula to check whether R is a rotation or a reflection, or equivalently, whether R is a rotation of (e_1, e_2, e_3) or $(e_1, e_2, -e_3)$.

1.3 Every Rotation Admits an Axis

Setting Euler's geometric proof[9] of this remarkable fact aside and using linear algebra, we remark that since rotations are linear maps and the axis we seek is by definition invariant (under the rotation), it is natural to describe it in eigenvector terms: as an eigenvector related to the eignevalue of one. We shall now prove that such a vector always exists.

Proof. Every eigenvalue has at least one eigenvector, so all that is needed is to prove that every rotation matrix has 1 as one of its eigenvalues. This is true iff. for any rotation matrix R, we have that $\mid R - 1.I \mid = 0$.
We know that $\mid R \mid = 1$, but $R.R^{-1} = I$, so $\mid R^{-1} \mid = 1$. Now

$$\mid R - I \mid = \mid (R - I)^T \mid = \mid R^T - I \mid,$$

which is equal $\mid R^{-1} - I \mid$ since R is orthonormal. Hence we have:

$$\mid R - I \mid = \mid R^{-1} - I \mid = \mid R^{-1}(I - R) \mid = \mid R^{-1} \mid\mid I - R \mid .$$

But for any matrix M in $\mathcal{M}_{3\times3}$ we have $\mid -M \mid = - \mid M \mid$ because M has odd column count. So $\mid R^{-1} \mid\mid I - R \mid = - \mid R - I \mid$ and we finally conclude that

$$\mid R - I \mid = - \mid R - I \mid$$

implying that $\mid R - I \mid = 0$. QED

[9]Euler stated that "In whatever way a sphere is turned about its centre, it is always possible to assign a diameter, whose direction in the translated state agrees with that of the initial state." [1]

1.4 Rodrigues's Rotation Formula

Given a rotation matrix R that admits an invariant axis along vector a (with unit length) and rotates an arbitrary vector x to y, Rodrigues's rotation formula states that

$$R.x = a(a.x) + (cos\alpha)[x - a(a.x)] + (sin\alpha)[a \times x],$$

where α is the angle between the projections of x and y on the plane orthogonal to a.

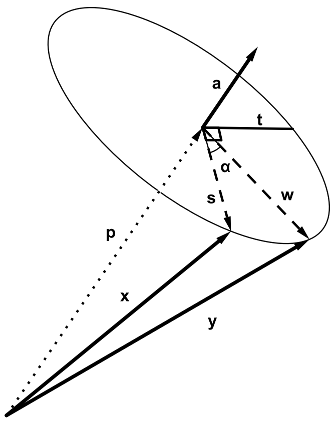

Figure 2: Rotation of the vector x around the axis a by an angle α.

Proof. By the properties of rotation, the lengths of x and y are equal and so are their angles relative to a. Hence, they both share the same orthogonal projection p on the axis and we have that $x = p + s$ and $y = p + w$ with

$$p = a.(a.x).$$

Turning our attention to w we note that both s and w are orthogonal to a and so they form a plane of which a is the normal. Since α is the angle between them, it satisfies $cos\alpha = \frac{s.w}{\|s\| \cdot \|w\|}$, while the projection of w on s is $\text{Proj}_s w = \frac{s.w}{\|s\| \cdot \|s\|}.s$. But it is clear that$\| s \| = \| w \|$ and so

$$\text{Proj}_s w = cos\alpha.s.$$

We can complete w by adding to $\text{Proj}_s w$ the rest vector u orthogonal to s. The direction of u is the same as that of the vector $t = a \times s$, while its length is given by the $\| u \| = sin\alpha \| w \|$ (considering the right triangle formed by w, s and u). We can thus write $u = \frac{\|w\|}{\|t\|}.sin\alpha.(a \times s)$ which simplifies to $u = sin\alpha.(a \times s)$ and finally gives

$$y = a.(a.x) + cos\alpha.s + sin\alpha.(a \times s).$$

We continue by substituting the vector s with $x - p$ to obtain

$$R.x = a.(a.x) + (cos\alpha)[x - a(a.x)] + (sin\alpha)[a \times x].$$

QED

1.5 Computing a Rotation Matrix from an Axis and Angle

One useful variant of the Rodrigues rotation formula is obtained by rearranging the right-hand-side to obtain an explicit expression for R independently of any vector x. To do this we put $a(a.x)$ and $[a \times x]$ into matrix form:

$$a(a.x) = \begin{pmatrix} a_1a_1 & a_1a_2 & a_1a_3 \\ a_2a_1 & a_2a_2 & a_2a_3 \\ a_3a_1 & a_3a_2 & a_3a_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$a \times x = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

We can then factor these sub-expressions and rewrite the formula as
$R.x = cos\alpha.x + a.(a.x)[1 - cos\alpha] + sin\alpha[a \times x]$ to find that

$$R = cos\alpha.I + (1 - cos\alpha).\begin{pmatrix} a_1a_1 & a_1a_2 & a_1a_3 \\ a_2a_1 & a_2a_2 & a_2a_3 \\ a_3a_1 & a_3a_2 & a_3a_3 \end{pmatrix} + sin\alpha.\begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}.$$

1.6 Computing an Axis and Angle from a Rotation Matrix

From the Rotation Matrix formula above, we notice that the diagonal of R is a function of $cos\alpha$ and the axis. We can make it a function of the angle alone with a change of basis to a B consisting of e_u, e_v and a (in that order), where e_u and e_v are natural basis vectors chosen appropriately. Finding such a basis is always possible by replacing in E_3 one of the vectors which is linearly dependent on a by a itself. Then, assuming R represents the linear map r, we have

$$R = \text{Rep}_{E_3,E_3}(r) = \text{Rep}_{B,E_3}(id).\text{Rep}_{B,B}(r).\text{Rep}_{E_3,B}(id),$$

with $\text{Rep}_B(a) = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$ and therefore

$$\text{Rep}_{B,B}(r) = cos\alpha.I + (1 - cos\alpha).\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + sin\alpha.\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Setting $S = \mathrm{Rep}_{B,B}(r)$ we find that $\mathrm{tr}S = 2.cos\alpha{+}1$. Since the trace operator is invariant under matrix similarity and R is similar to S we can also write

$$cos\alpha = \frac{\mathrm{tr}R - 1}{2}.$$

This determines the angle α up to a choice of sign and multiples of 2π. Of course, there are multiple equivalent axis and angle parameterizations for any given rotation. Per example, negating the angle and flipping the axis direction results in an equivalent rotation. Consequently, it is fine to arbitrarily choose any sign for the cosine inverse, because the axis can then be computed accordingly and so it remains to find a formula for the axis given the angle.

Looking once more at the Rotation matrix formula, we notice that the skew symmetric matrix part of R provides a good opportunity to find the axis components. Indeed, it is obvious that $\mathrm{Skw}R = sin\alpha. \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$ while at the same time $\mathrm{Skw}R = \frac{1}{2}(R - R^T)$ in general, therefore we have

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \frac{1}{2.sin\alpha} \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix}.$$

1.7 Quaternions

As an extension of the complex numbers, quaterions form a vector space in $\mathbb{R}^4$, with a standard choice of basis elements: $1, i, j, k$. A quaternion a can therefore be represented as the vector:

$$(a_0, a_1, a_2, a_3)^T = a_0.1 + a_1.i + a_2.j + a_3.k,$$

and addition and scalar multiplication follow:

$$\begin{aligned} a + b &= (a_0 + b_0).1 + (a + b_1).i + (a + b_2).j + (a + b_3).k \\ s.a &= (sa_0).1 + (sa_1).i + (sa_2).j + (sa_3).k. \end{aligned}$$

Quaternions are also a noncommutative algebra, governed by the mulitplication rule of the basis elements:

$$i^2 = j^2 = k^2 = ijk = -1.$$

This rule is sufficient to determine all basis products:

$$ij = k,\ ji = -k,\ jk = i,\ kj = -i,\ ki = j,\ ik = -j.$$

along with quaternion multiplication:

$$\begin{aligned}
a.b &= (a_0.1 + a_1.i + a_2.j + a_3.k).(b_0.1 + b_1.i + b_2.j + b_3.k)\\
&= (a_1b_0 + a_0b_1).i + (a_2b_0 + a_0b_2).j + (a_3b_0 + b_3a_0)k\\
&\quad + (a_0b_0).1 + (a_1b_1).i^2 + (a_2b_2).j^2 + (a_3b_3).k^2\\
&\quad + (a_1b_2 + a_2b_1).ij + (a_1b_3 + a_3b_1).ik + (a_2b_3 + a_3b_2).jk\\
&= (a_0b_0 - a_1b_1 - a_2b_2 - a_3b_3).1\\
&\quad + (a_0b_1 + a_1b_0 + a_2b_3 - a_3b_2).i\\
&\quad + (a_0b_2 - a_1b_3 + a_2b_0 + a_3b_1).j\\
&\quad + (a_0b_3 + a_1b_2 - a_2b_1 + a_3b_0).k.
\end{aligned}$$

Quaternion multiplication can be expressed more concisely by splitting the representation of a quaternion a into a scalar a_s and a vector a_v in $\mathbb{R}^3$ such that $a = (a_s, a_v^T)^T$. Less formally, we shall write $a = (a_s, a_v)$ and using this notation, multiplication becomes:

$$(a_s, \overrightarrow{a_v}).(b_s, \overrightarrow{b_v}) = (a_sb_s - \overrightarrow{a_v}.\overrightarrow{b_v},\ a_s\overrightarrow{b_v} + b_s\overrightarrow{a_v} + \overrightarrow{a_v} \times \overrightarrow{b_v}).$$

The norm (or length) of a quaternion is calculated using conjugates. In this way, it is compatible with generalzing Euclidean distance and is obtained by:

$$\| a \|^2 = a.a^c = a_0^2 + a_1^2 + a_2^2 + a_3^2,$$

where a^c is the conjugate of a and $a^c = \frac{1}{2}(a + iqi + jqj + kqk)$. In split notation this reduces to the simple identity:

$$a^c = (a_s, -a_v).$$

Quaternion norm is multiplicative for both scalars and quaternions, that is, where b is either a scalar or a quaternion, we have that:

$$\| b.a \| = \| b \| . \| a \| .$$

For any non-zero quaternion a, it is possible to define the unit quaternion using u as:

$$u = \frac{a}{\| a \|}.$$

Furthermore, the conjugate can be used to define the reciprocal a^{-1} such that:

$$a^{-1} = \frac{a^c}{\| a \|^2},$$

which unlike general multiplication, is commutative:

$$a^{-1}.a^c = a^c.a^{-1} = 1.$$

References

[1] L. Euler. Formulae generales pro translatione quacunque corporum rigidorum. *Novi Commentarii Acad. Petropolitanae*, 20:189–207, 1776.

[2] Bob Palais, Richard Palais, and Stephen Rodi. A disorienting look at euler's theorem on the axis of a rotation. *The American Mathematical Monthly*, 116(10):892–909, 2009.