Rotation Matrices

James D Emery

Edition: 10/2/15

Contents

1 Rotation Matrix Defined by Axis and Angle

Let a unit vector $\mathbf{n}$ specify a rotation axis, and let α be a rotation angle in the right hand rule sense. We shall show that the rotation of a vector $\mathbf{x}$ to a vector $\mathbf{y}$, around an axis in the direction of a vector $\mathbf{n}$, by an angle α, can be accomplished by multiplying $\mathbf{x}$ by a rotation matrix $\mathbf{M}$. To show this we shall write vectors as column vectors. So for example we shall write vector $\mathbf{n}$ as

$$\mathbf{n} = n_1\mathbf{i} + n_2\mathbf{j} + n_3\mathbf{k},$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the standard basis unit vectors directed along the cartesian axes. We shall also write each such vector as a 3 by 1 matrix known as a column vector,

$$\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}.$$

We shall find a matrix $\mathbf{M}$, so that

$$\mathbf{y} = \mathbf{M}\mathbf{x}.$$

(This derivation is a solution to exercise 12.51, p 421, **Applied Linear Algebra**, Ben Noble, 1969, Prentice Hall.)

Let the origin be O. Refer to the figures called **R3 View** and **Plane View**. The figure **Plane View** shows the plane perpendicular to the rotation axis, defined by vector $\mathbf{n}$, with points P, Q, R and S in this plane.
Vector $\mathbf{x}$ is the vector $\vec{OP}$ from the origin O in 3-space to point P.

$$\mathbf{x} = \vec{OP}.$$

Vector $\mathbf{y}$ is the vector $\vec{OR}$ from the origin to point R.

$$\mathbf{y} = \vec{OR}.$$

Vector $\vec{OQ}$ is the rotation axis, the direction of $\mathbf{n}$.

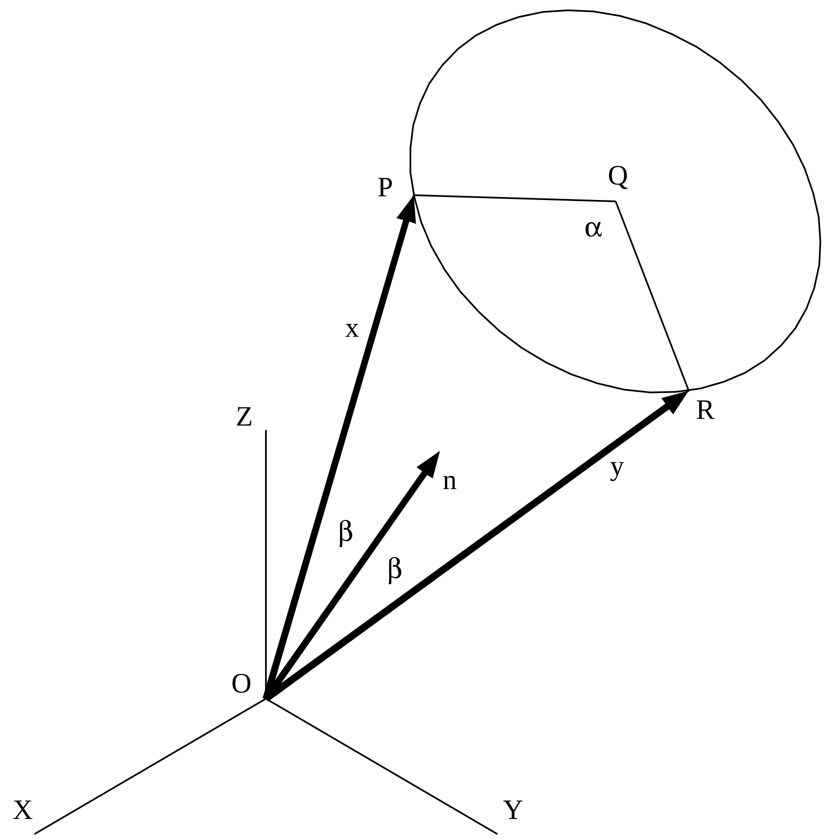

Figure 1: **R3 View.** In the three dimensional view we see that vector **n** is in the direction of the rotation axis directed toward the center of the rotation circle, which is point Q. Vector **x**, which is $\vec{OP}$, is rotated by angle α to vector **y**, which is $\vec{OR}$. Angle β is the measure of both angle POQ and ROQ.

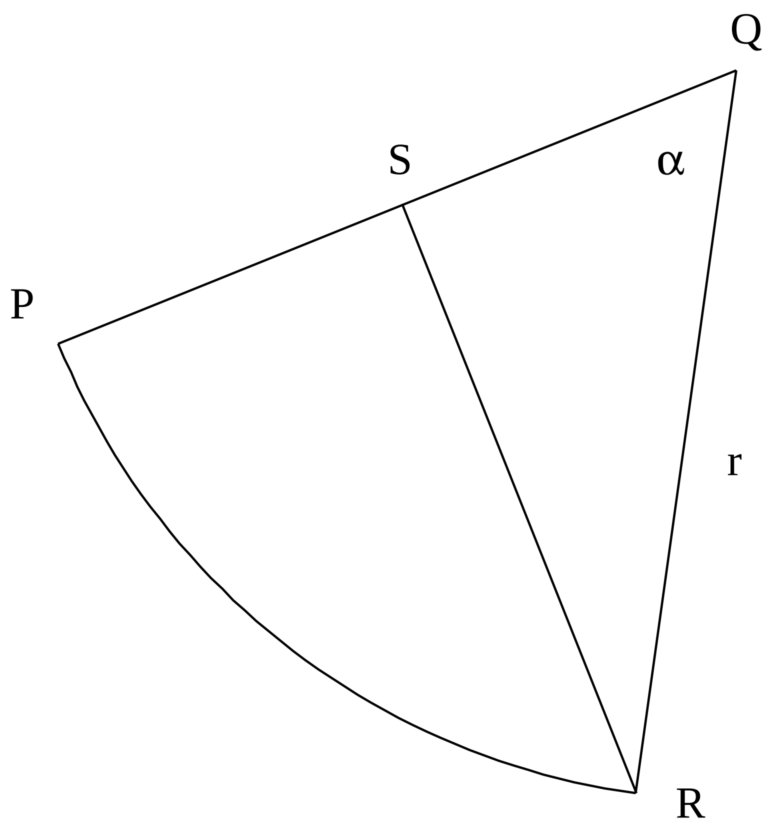

Figure 2: **Plane View.** The plane is perpendicular to the rotation axis through Q. P is rotated to R. The rotation angle is α. The length of $\vec{SQ}$ is $r\cos(\alpha)$, so the length of $\vec{PS}$ is $r(1-\cos(\alpha))$.

Let us project point P to the axis defined by $\mathbf{n}$, getting point Q. Then

$$\vec{PQ} = (\mathbf{x}\cdot\mathbf{n})\mathbf{n} - \mathbf{x}.$$

We have a plane triangle PQR, where the measure of angle PQR is α. Although the derivation is valid for any angle α, we have drawn the figure so that $\alpha < \pi/2$. We construct a point S on line PQ, so that SR is perpendicular to PQ. Then vector y is

$$\mathbf{y} = \vec{OP} + \vec{PS} + \vec{SR}.$$

We shall show that each of these vectors is a product of a matrix and $\mathbf{x}$. Then the matrix we are looking for is the sum of these matrices.

Let β be the angle between x and n. Let r be the length of PQ and of QR. We have

$$r = \|\mathbf{x}\|\sin(\beta).$$

Considering the triangle SQR, we see that the length of $\vec{SQ}$ is $r\cos(\alpha)$. Thus

$$\vec{PS} = \frac{r(1-\cos(\alpha))}{r}\vec{PQ} = (1-\cos(\alpha))((\mathbf{x}\cdot\mathbf{n})\mathbf{n} - \mathbf{x}).$$

Vector $\vec{SR}$ is perpendicular to the plane containing $\mathbf{x}$ and $\mathbf{n}$. Such a unit vector is

$$\frac{\mathbf{n}\times\mathbf{x}}{\|\mathbf{x}\|\sin(\beta)} = \frac{\mathbf{n}\times\mathbf{x}}{r}.$$

From the figure the length of SR is $r\sin(\alpha)$. Hence

$$\vec{SR} = \mathbf{n}\times\mathbf{x}\sin(\alpha).$$

So

$$\mathbf{y} = \vec{OP} + \vec{PS} + \vec{SR} = \mathbf{x} + (1-\cos(\alpha))((\mathbf{x}\cdot\mathbf{n})\mathbf{n} - \mathbf{x}) + \sin(\alpha)\mathbf{n}\times\mathbf{x}$$

$$= \cos(\alpha)\mathbf{x} + (1-\cos(\alpha))(\mathbf{x}\cdot\mathbf{n})\mathbf{n}) + \sin(\alpha)\mathbf{n}\times\mathbf{x}.$$

Each of these terms is a linear transformation of vector $\mathbf{x}$. It can be written as a matrix equation.

Indeed we have

$$\mathbf{n} \times \mathbf{x} = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Let

$$\mathbf{N} = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

Because $\mathbf{x}$ and $\mathbf{n}$ are column vectors, we have

$$(\mathbf{x} \cdot \mathbf{n})\mathbf{n} = \mathbf{n}(\mathbf{n} \cdot \mathbf{x}) = \mathbf{n}\mathbf{n}^T\mathbf{x}.$$

Then we have

$$\mathbf{n}\mathbf{n}^T = \begin{bmatrix} n_1n_1 & n_1n_2 & n_1n_3 \\ n_2n_1 & n_2n_2 & n_2n_3 \\ n_3n_1 & n_3n_2 & n_3n_3 \end{bmatrix}.$$

We may write the equation as

$$\mathbf{y} = [\cos(\alpha)\mathbf{I} + (1 - \cos(\alpha))\mathbf{n}\mathbf{n}^T + \sin(\alpha)\mathbf{N}]\mathbf{x}.$$

We may simplify this a little further, because

$$\mathbf{N}^2 = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -n_3^2 - n_2^2 & n_1n_2 & n_1n_3 \\ n_2n_1 & -n_2^2 - n_1^2 & n_2n_3 \\ n_3n_1 & n_3n_2 & -n_2^2 - n_1^2 \end{bmatrix}$$

$$= \begin{bmatrix} n_1n_1 - 1 & n_1n_2 & n_1n_3 \\ n_2n_1 & n_2n_2 - 1 & n_2n_3 \\ n_3n_1 & n_3n_2 & n_3n_3 - 1 \end{bmatrix}$$

$$= \mathbf{n}\mathbf{n}^T - \mathbf{I}.$$

We have shown that

$$\mathbf{N}^2 + I = \mathbf{n}\mathbf{n}^T.$$

So

$$\mathbf{y} = [\cos(\alpha)I + (1-\cos(\alpha))\mathbf{n}\mathbf{n}^T + \sin(\alpha)\mathbf{N}]\mathbf{x}$$
$$= [\cos(\alpha)\mathbf{I} + (1-\cos(\alpha))(\mathbf{N}^2 + \mathbf{I}) + \sin(\alpha)\mathbf{N}]\mathbf{x}$$
$$= [\mathbf{I} + (1-\cos(\alpha))\mathbf{N}^2 + \sin(\alpha)\mathbf{N}]\mathbf{x}.$$

So our rotation matrix

$$\mathbf{M} = \mathbf{I} + (1-\cos(\alpha))\mathbf{N}^2 + \sin(\alpha)\mathbf{N}.$$

Notice that $\mathbf{Nn}$ is really the cross product of two parallel vectors, so is zero. Explicitely

$$\mathbf{Nn} = \begin{bmatrix} -n_3n_2 + n_2n_3 \\ n_3n_1 - n_1n_3 \\ -n_2n_1 + n_1n_2 \end{bmatrix} = 0$$

Hence multiplying

$$\mathbf{N}^2 + I = \mathbf{n}\mathbf{n}^T.$$

by $\mathbf{N}$, gives

$$\mathbf{N}^3 + \mathbf{N} = \mathbf{N}\mathbf{n}\mathbf{n}^T = 0,$$

so

$$\mathbf{N}^3 = -\mathbf{N}.$$

Continuing we find

$$\mathbf{N}^4 = -\mathbf{N}^2,$$
$$\mathbf{N}^5 = \mathbf{N},$$
$$\mathbf{N}^6 = \mathbf{N}^2,$$
$$\mathbf{N}^7 = -\mathbf{N},$$
$$\mathbf{N}^8 = -\mathbf{N}^2$$
$$.......$$

and so on. We will use this later to show that the rotation matrix $\mathbf{M}$ is an exponential. Let us show explicitly that $\mathbf{M}$ is an orthogonal matrix. Using the facts that

$$\mathbf{N}^T = -\mathbf{N},$$

and

$$\mathbf{N}^4 = -\mathbf{N}^2,$$

we have

$$\mathbf{M}\mathbf{M}^T = [\mathbf{I} + (1 - \cos(\alpha))\mathbf{N}^2 + \sin(\alpha)\mathbf{N}][\mathbf{I} + (1 - \cos(\alpha))\mathbf{N}^2 - \sin(\alpha)\mathbf{N}].$$

Expanding this expression we find that

$$\mathbf{M}\mathbf{M}^T = \mathbf{I}.$$

See the references: Jay P Filmore, **A Note On Rotation Matrices**, IEEE Computer Graphics and Applications, February 1984, the orthogonal matrix subroutine **orthgm** in libraries **emerylib.ftn** and **emerylib.c**, as well as **axisang**, and **v2rot**, and the program **trnsf.c**. This latter program allows one to construct a general transformation matrix in steps, and then to apply it to a file containing points. See the batch files, **viewffun.bat**, **viewcfun.bat**, **getffun.bat** , and **getcfun.bat**, which allow one to see the subroutines and functions available in the libraries, and to extract such functions and subroutines so that they may be incorporated into programs. The library **emerylib.c** contains both C functions and C++ functions.

2 Axis and Angle of a Proper Rotation Matrix

Suppose we have a set of orthogonal basis vectors u_1, u_2, u_3. Clearly the matrix of a rotation transformation with respect to this basis, about axis u_3 by angle θ, is

$$M = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The trace of this matrix is the sum of the diagonal elements. The trace of M is

$$trace(M) = 2\cos(\theta) + 1.$$

The matrix of this rotation with respect to any other orthogonal basis is

$$M' = PMP^{-1},$$

where P is the change of basis matrix.

The trace has the property that for n by n matrices A and B,

$$trace(AB) = trace(BA).$$

This follows because

$$\begin{aligned} trace(AB) &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ji} \\ &= \sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}b_{ji} \\ &= \sum_{j=1}^{n}\sum_{i=1}^{n} b_{ji}a_{ij} \\ &= trace(BA). \end{aligned}$$

Then

$$trace(M') = trace(PMP^{-1}) = trace(P^{-1}PM) = trace(M) = 2\cos(\theta) + 1.$$

Hence the trace of a rotation matrix determines the rotation angle. A rotation matrix is an orthogonal matrix in which the column vectors (also the row vectors) are unit orthogonal vectors. Hence

$$MM^T = I,$$

That is M^T is the inverse of M.

$$1 = det(I) = det(MM^T) = det(M)det(M^T) = det(M)^2.$$

It follows that

$$|det(M)| = 1.$$

For a proper orthogonal matrix $det(M) = 1$.

A vector v in the direction of the rotation axis is transformed to itself. Thus it is an eigenvector with eigenvalue $\lambda = 1$, that is

$$Mv = \lambda v.$$

Clearly this is the only eigenvector if M is not the identity. For any other vector that is not in the direction of the axis, is rotated. Hence the axis of the rotation may be obtained by calculating an eigenvector of M. Given a proper orthogonal transformation M, we can use the explicit formula for M calculated above to find an eigenvector of M.

We have

$$M = I + (1 - \cos(\alpha))N^2 + \sin(\alpha)N$$

and

$$M^T = I + (1 - \cos(\alpha))N^2 - \sin(\alpha)N.$$

So

$$\frac{1}{2}(M + M^T) = I + (1 - \cos(\alpha))N^2.$$

We have

$$N^2 = nn^T - I,$$

so

$$\frac{1}{2}(M + M^T) = I + (1 - \cos(\alpha))(nn^T - I)$$
$$= \cos(\alpha)I + (1 - \cos(\alpha))nn^T$$

Now

$$\cos(\alpha) = \frac{trace(M) - 1}{2},$$

and

$$(1 - \cos(\alpha)) = 1 - \frac{trace(M) - 1}{2} = \frac{3 - trace(M)}{2}.$$

Thus

$$(M + M^T) - (trace(M) - 1)I = (3 - trace(M))nn^T.$$

Any row or column of matrix nn^T is a multiple of vector n. So any row or column of matrix

$$(M + M^T) - (trace(M) - 1)I$$

is an eigenvector for eigenvalue $\lambda = 1$, and gives the rotation axis. Here is a Fortran subroutine called **axisang** for finding the axis and angle of a rotation matrix:

```
c+ axisang axis and angle of a rotation matrix, January 2004
      subroutine axisang(a,x,t)
      implicit real*8(a-h,o-z)
c Input:
c  a  3 by 3 orthogonal rotation matrix
c Output:
c  x  unit vector in the direction of the rotation axis
c  t  rotation angle, 0 <= t <= pi (right hand rule)
c References:
c  (1) Rotations, James Emery, January 2004, (rotations.tex)
c  (2) A Note on Rotation Matrices, Jay P Fillmore,
c      IEEE Computer Graphics and Applications, February, 1984.
c  (3) Applied Linear Algebra, B. Noble, 1969.
      real*8 a(3,3),b(3,3),x(3),y(3),z(3),w(3)
      real*8 c(3,3)
      zero=0.
c     compute the trace of a.
      trc=a(1,1)+a(2,2)+a(3,3)
c     Compute the positive angle of rotation
      cs=(trc-1.0d0)/2.0d0
      sn=sqrt(1.0d0 - cs*cs)
      t=atan2(sn,cs)
c     Find the transpose of a
      call mattrn(a,3,3,3,b,3)
c     Add a to its transpose.
      call mata(a,3,3,3,b,3,c,3)
      s=trc-1
      do i=1,3
       c(i,i)=c(i,i)-s
      enddo
c     We have computed a matrix c whose row and column vectors are
c     multiples of the required eigenvector.
      amax=0.
      do i=1,3
       anorm=c(i,1)**2+c(i,2)**2+c(i,3)**2
       if(anorm .gt. amax)then
        k=i
        amax=anorm
       endif
      enddo
      anorm=sqrt(amax)
      xmax=0.
      do j=1,3
       y(j)=1.
       x(j)=c(k,j)/anorm
```

```
         if(abs(x(j)).gt.xmax)then
          m=j
          xmax=abs(x(j))
         endif
        enddo
        y(m)=0.
        if(m .eq. 1)then
         y(2)=x(3)
         y(3)=-x(2)
        endif
        if(m .eq. 2)then
         y(1)=x(3)
         y(3)=-x(1)
        endif
        if(m .eq. 3)then
         y(1)=x(2)
         y(2)=-x(1)
        endif
c       We have found a unit eigenvector x and
c       we have found a vector y perpendicular to x.
c       Rotate y to z,  z=a*y.
        do i=1,3
         z(i)=0.
         do j=1,3
          z(i)=z(i) + a(i,j)*y(j)
         enddo
        enddo
c       Compute the cross product of y and z, w=y cross z
        call crsspr(y,z,w)
c       If the right hand rule is satisfied, w should be in the
c       direction of the axis x.
        s=dotpr(x,w)
c       If the right hand rule is not satisfied, reverse the direction
c       of the axis.
        if(s .lt. zero)then
         do j=1,3
          x(j)=-x(j)
         enddo
        endif
        return
        end
```

3 Obtaining the Rotation As The Exponential of an Element of a Banach Algebra

A Banach Algebra is a normed linear vector space in which a multiplication is defined, and which is complete. A complete space is one in which every Cauchy sequence converges to an element of the space. Let B be an element of the space of n by n matrices. One can take as norm the sum of the absolute values of the elements of the matrix. A norm satisfies the triangle inequality.

$$\|B_1 + B_2\| \leq \|B_1\| + \|B_2\|.$$

Also for a Banach Algebra the norm satisfies

$$\|B_1 B_2\| \leq \|B_1\|\|B_2\|.$$

We can take the exponential of the norm of B

$$e^{\|B\|} = \sum_{n=0}^{\infty} \frac{\|B\|^n}{n!}.$$

This converges for every B because the real exponential function is an entire function. Since it converges, the partial sums are a Cauchy sequence. This means that given some $\epsilon > 0$ there exists some integer N so that for every $m, n > N$, say $n > m$, the partial sums S_m, S_n differ by less than ϵ, that is

$$\frac{\|B\|^{m+1}}{(m+1)!} + \frac{\|B\|^{m+2}}{(m+2)!} + ... + \frac{\|B\|^n}{n!} < \epsilon$$

But using the norm inequalities this shows that

$$\left\| \frac{B^{m+1}}{(m+1)!} + \frac{B^{m+2}}{(m+2)!} + ... + \frac{B^n}{n!} \right\| < \epsilon.$$

It follows that the partial sums of the series

$$e^B = \sum_{n=0}^{\infty} \frac{B^n}{n!}$$

form a Cauchy sequence in the Banach space. Since the Banach space is complete, the series converges to some element, here a n by n matrix. So the exponential of the matrix (operator) is defined. Let us take the 3 by 3 matrix N above defined by our unit axis vector n. That is,

$$N = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

We use the properties of N found above, namely

$$N^3 = -N.$$

$$N^4 = -N^2,$$

$$N^5 = N,$$

$$N^6 = N^2,$$

$$N^7 = -N,$$

$$N^8 = -N^2$$

....

Then we write

$$e^{tN} = \sum_{n=0}^{\infty} \frac{t^n N^n}{n!}$$

$$= I + (tN + \frac{t^3}{3!}N^3 + \frac{t^5}{5!}N^5 + ...)$$

$$+(\frac{t^2}{2!}N^2 + \frac{t^4}{4!}N^4 + \frac{t^6}{6!}N^6 + ...)$$

$$= I + (tN - \frac{t^3}{3!}N + \frac{t^5}{5!}N + ...)$$

$$+(\frac{t^2}{2!}N^2 - \frac{t^4}{4!}N^2 + \frac{t^6}{6!}N^2 + ...)$$

$$= I + (t - \frac{t^3}{3!} + \frac{t^5}{5!} + ...)N$$

$$+(\frac{t^2}{2!} - \frac{t^4}{4!} + \frac{t^6}{6!} + ...)N^2$$

$$= I + \sin(t)N + (1 - \cos(t))N^2.$$

This is the formula for the rotation matrix M derived above. Hence the rotation matrix for rotation about axis n by angle α is the exponential

$$M = e^{\alpha N}.$$

4 Properties of The Exponential of a Matrix

If a matrix A is upper triangular

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ 0 & a_{22} & a_{23} & ... & a_{2n} \\ 0 & 0 & a_{33} & .... & a_{3n} \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & a_{nn} \end{bmatrix}$$

Then A^n is an upper triangular matrix of the form

$$\begin{bmatrix} a_{11}^n & ... & ... & .... & ... \\ 0 & a_{22}^n & ... & ... & ... \\ 0 & 0 & a_{33}^n & .... & ... \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & a_{nn}^n, \end{bmatrix}$$

This becomes apparent by just calculating an example such as say A^2 for a 3 by 3 upper triangular matrix. Now if a sequence of matrices converges to a matrix M, then a sequence consisting of say the ijth element of each matrix in the sequence, converges to the ijth element of M. It follows that if A is upper triangular, then

$$e^A = \begin{bmatrix} e^{a_{11}} & ... & ... & .... & ... \\ 0 & e^{a_{22}} & ... & ... & ... \\ 0 & 0 & e^{a_{33}} & .... & ... \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & e^{a_{nn}} \end{bmatrix}$$

So

$$det(A) = e^{a_{11}} e^{a_{22}} ... e^{a_{nn}}.$$

$$= e^{a_{11}+a_{22}+...+a_{nn}}$$

$$= e^{trace(A)}.$$

If A is a general matrix, then there exists a matrix T such that

$$T^{-1}AT = B,$$

and B is upper triangular. For a simple proof, see Bellman Page 21. Note that T might be a complex matrix. We have

$$T^{-1}(e^A)T = I+\frac{1}{1!}T^{-1}AT+\frac{1}{2!}(T^{-1}AT)(T^{-1}AT)+\frac{1}{3!}(T^{-1}AT)(T^{-1}AT)(T^{-1}AT)+...$$

$$= e^{T^{-1}AT}.$$

Hence because $T^{-1}AT$ is upper triangular, we have

$$det(e^A) = det(T^{-1}e^AT) = det(e^{T^{-1}AT}) = e^{trace(T^{-1}AT)} = e^{trace(A)}.$$

We have proven:

Proposition For any square matrix

$$det(e^A) = e^{trace(A)}.$$

For our matrix N above,

$$N = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

the trace is 0. It follows that

$$det(e^N) = e^{trace(N)} = e^0 = 1.$$

Because the transpose of N is - N, we see that

$$(e^{\alpha N})^T = e^{-\alpha N}.$$

It may be shown that

$$e^{A+B} = e^A e^B$$

if and only if

$$AB = BA.$$

So we find that

$$e^{\alpha N}(e^{\alpha N})^T = e^{\alpha N}(e^{-\alpha N}) = I.$$

So we see directly that
$$e^{\alpha N}$$
is a proper orthogonal matrix, and so represents a rotation. Also directly $n = (n_1, n_2, n_3)$ is an eigenvector because $Nn = 0$ and the only nonzero term in the exponential series for $e^{\alpha N}n$ is $In = n$. So n is the rotation axis. Because
$$N^2 = \begin{bmatrix} n_1n_1 - 1 & n_1n_2 & n_1n_3 \\ n_2n_1 & n_2n_2 - 1 & n_2n_3 \\ n_3n_1 & n_3n_2 & n_3n_3 - 1 \end{bmatrix}$$
$$trace(N^2) = \|n\|^2 - 3 = -2.$$
Hence
$$trace(e^{\alpha N}) = trace(I + \sin(\alpha)N + (1 - \cos(\alpha))N^2)$$
$$= 3 + 0 + (1 - \cos(\alpha))(-2) = 1 + 2\cos(\alpha),$$
So we see directly that α is the rotation angle.

5 A Test Program

Here is a test program using subroutines axisang and orthgm:

```
c orthgm.ftn Compute orthogonal rotation matrix
c 9/30/15
c 1/7/04
c Test of revised orthgm and axisang subroutine
c Compute rotation matrix from axis and angle
c Compute axis and angle from orthogonal matrix
      implicit real*8(a-h,o-z)
      dimension a(3,3)
      dimension x(3)
      dimension ain(3)
      dimension tmp(3)
      pi=4.0d0*atan(1.0d0)
      a(1,1)=0.
      a(1,2)=0.
      a(1,3)=-1.
      a(2,1)=0.
      a(2,2)=-1.
      a(2,3)=0.
      a(3,1)=-1.
      a(3,2)=0.
      a(3,3)=0.
      write(*,*)' Enter an angle t (degrees), -180 <= t <= 180 '
      call readr(nf,ain, nr)
```

```
      t=ain(1)*pi/180.0d0
      write(*,'(a,g15.8,a)')' Angle  = ', ain(1),' degrees'
      write(*,*)' Enter a direction vector '
      call readr(nf,ain, nr)
      do i=1,3
       x(i)=ain(i)
      enddo
      write(*,'(a,3(1x,g15.8))')' Direction= ',(x(k),k=1,3)
      call orthgm(x,t,a)
      write(*,*)' rotation matrix a = '
      do i=1,3
       write(*,'(3(1x,g15.8))')(a(i,j),j=1,3)
      enddo
      ia=3
      call det3(a,ia,det)
      write(*,'(1x,a,g22.14)')' determinant=',det
      call axisang(a,x,t)
      write(*,*)' Computed rotation axis and angle of matrix a'
      write(*,'(1x,a,g22.14,g22.14,g22.14)')' axis=',x(1),x(2),x(3)
      write(*,'(1x,a,g22.14)')' angle= ',t*180./pi
      write(*,*)
      write(*,*)' Now we interchange columns 1 and 2 of matrix a'
      write(*,*)' to create an orthogonal matrix,'
      write(*,*)' which is an improper rotation matrix. '
      do i=1,3
       tmp(i)=a(i,1)
       a(i,1)=a(i,2)
       a(i,2)=tmp(i)
      enddo
c
      write(*,*)' a = '
      do i=1,3
       write(*,'(3(1x,g15.8))')(a(i,j),j=1,3)
      enddo
      call axisang(a,x,t)
      call det3(a,ia,det)
      write(*,'(1x,a,g22.14)')' determinant=',det
      write(*,*)' Attempt to compute an axis and angle'
      write(*,*)' Computed axis and angle of this improper rotation'
      write(*,'(1x,a,g22.14,g22.14,g22.14)')' axis=',x(1),x(2),x(3)
      write(*,'(1x,a,g22.14)')' angle= ',t*180./pi
      end
c+ axisang axis and angle of a rotation matrix, January 2004
      subroutine axisang(a,x,t)
      implicit real*8(a-h,o-z)
c Input:
c  a  3 by 3 orthogonal rotation matrix
c Output:
c  x  unit vector in the direction of the rotation axis
c  t  rotation angle, 0 <= t <= pi (right hand rule)
c References:
c  (1) Rotations, James Emery, January 2004, (rotations.tex)
c  (2) A Note on Rotation Matrices, Jay P Fillmore,
c      IEEE Computer Graphics and Applications, February, 1984.
c  (3) Applied Linear Algebra, B. Noble, 1969.
```

```
      real*8 a(3,3),b(3,3),x(3),y(3),z(3),w(3)
      real*8 c(3,3)
      zero=0.
c     compute the trace of a.
      trc=a(1,1)+a(2,2)+a(3,3)
c     Compute the positive angle of rotation
      cs=(trc-1.0d0)/2.0d0
      sn=sqrt(1.0d0 - cs*cs)
      t=atan2(sn,cs)
c     Find the transpose of a
      call mattrn(a,3,3,3,b,3)
c     Add a to its transpose.
      call mata(a,3,3,3,b,3,c,3)
      s=trc-1
      do i=1,3
       c(i,i)=c(i,i)-s
      enddo
c     We have computed a matrix c whose row and column vectors are
c     multiples of the required eigenvector.
      amax=0.
      do i=1,3
       anorm=c(i,1)**2+c(i,2)**2+c(i,3)**2
       if(anorm .gt. amax)then
        k=i
        amax=anorm
       endif
      enddo
      anorm=sqrt(amax)
      xmax=0.
      do j=1,3
       y(j)=1.
       x(j)=c(k,j)/anorm
       if(abs(x(j)).gt.xmax)then
        m=j
        xmax=abs(x(j))
       endif
      enddo
      y(m)=0.
      if(m .eq. 1)then
       y(2)=x(3)
       y(3)=-x(2)
      endif
      if(m .eq. 2)then
       y(1)=x(3)
       y(3)=-x(1)
      endif
      if(m .eq. 3)then
       y(1)=x(2)
       y(2)=-x(1)
      endif
c     We have found a unit eigenvector x and
c     we have found a vector y perpendicular to x.
c     Rotate y to z,  z=a*y.
      do i=1,3
       z(i)=0.
```

```
       do j=1,3
        z(i)=z(i) + a(i,j)*y(j)
       enddo
      enddo
c     Compute the cross product of y and z, w=y cross z
      call crsspr(y,z,w)
c     If the right hand rule is satisfied, w should be in the
c     direction of the axis x.
      s=dotpr(x,w)
c     If the right hand rule is not satisfied, reverse the direction
c     of the axis.
      if(s .lt. zero)then
       do j=1,3
        x(j)=-x(j)
       enddo
      endif
      return
      end
c+ mattrn   matrix transpose
      subroutine mattrn(a,ia,m,n,b,ib)
        implicit real*8(a-h,o-z)
c arguments
c   a-matrix
c  ia-row dimension of a in calling program
c   m-number of rows in a
c   n-number of columns in a
c   b-transpose of a
c  ib-row dimension of b in calling program
c
      dimension a(ia,*),b(ib,*)
      do 10 i=1,m
      do 10 j=1,n
10    b(j,i)=a(i,j)
      return
      end
c+ mata matrix addition
      subroutine mata(a,ia,m,n,b,ib,c,ic)
        implicit real*8(a-h,o-z)
c arguments
c  a-matrix
c ia-row dimension of a in calling program
c  m-number of rows
c  n-number of columns
c  b-matrix
c ib-row dimension of b in calling program
c  c-sum matrix: c=a*b
c ic-row dimension of c in calling program
      dimension a(ia,*),b(ib,*),c(ic,*)
c     c=a+b
      do 10 i=1,m
      do 10 j=1,n
      c(i,j)=a(i,j)+b(i,j)
10    continue
      return
      end
```

```
c+ crsspr  vector cross product.
      subroutine crsspr(a,b,c)
      implicit real*8(a-h,o-z)
c     c=product of a and b
      dimension a(3),b(3),c(3)
      c(1)=a(2)*b(3)-a(3)*b(2)
      c(2)=a(3)*b(1)-a(1)*b(3)
      c(3)=a(1)*b(2)-a(2)*b(1)
      return
      end
c+ dotpr scalar product of 3-space vectors
      function dotpr(a,b)
      implicit real*8(a-h,o-z)
c     2/5/97
      dimension a(*),b(*)
      s=0.
      do i=1,3
       s=s+a(i)*b(i)
      enddo
      dotpr=s
      return
      end
c+ readr read a row of numbers and return in double precision array
      subroutine readr(nf, a, nr)
      implicit real*8(a-h,o-z)
c Input:
c nf    unit number of file to read
c       nf=0 is the standard input file (keyboard)
c Output:
c a     array containing double precision numbers found
c nr    number of values in returned array,
c     or 0 for empty or blank line,
c       or -1 for end of file on unit nf.
c   Numbers are separated by spaces.
c   Examples of valid numbers are:
c   12.13 34 45e4 4.78e-6 4e2,5.6D-23,10000.d015
c requires subroutine valsub and function lenstr
c a semicolon and all characters following are ignored.
c This can be used for comments.
c modified 6/16/97 added semicolon feature
      dimension a(*)
      character*200  b
      character*200  c
      character*1 d
        c=' '
      if(nf.eq.0)then
         read(*,'(a)',end=99)b
      else
         read(nf,'(a)',end=99)b
      endif
      nr=0
      lsemi=index(b,';')
      if(lsemi .gt. 0)then
       if(lsemi .gt. 1)then
        b=b(1:(lsemi-1))
```

```
        else
         return
        endif
       endif
       l=lenstr(b)
       if(l.ge.200)then
         write(*,*)' error in readr subroutine '
         write(*,*)' record is too long '
       endif
       do 1 i=1,l
        d=b(i:i)
        if (d.ne.' ') then
         k=lenstr(c)
         if (k.gt.0)then
            c=c(1:k)//d
         else
            c=d
         endif
        endif
        if( (d.eq.' ').or.(i.eq.l)) then
         if (c.ne.' ') then
           nr=nr+1
            call valsub(c,a(nr),ier)
           c=' '
         endif
        endif
1      continue
       return
99     nr=-1
       return
       end
c+ valsub converts string to floating point number (r*8)
       subroutine valsub(s,v,ier)
       implicit real*8(a-h,o-z)
c      examples of valid strings are: 12.13 34 45e4 4.78e-6 4E2
c      the string is checked for valid characters,
c      but the string can still be invalid.
c       s-string
c       v-returned value
c       ier- 0 normal
c            1 if invalid character found, v returned 0
c
       logical p
       character s*(*),c*50,t*50,ch*15
       character z*1
       data ch/'1234567890+-.eE'/
       v=0.
       ier=1
       l=lenstr(s)
       if(l.eq.0)return
       p=.true.
       do 10 i=1,l
       z=s(i:i)
       if((z.eq.'D').or.(z.eq.'d'))then
           s(i:i)='e'
```

```
      endif
      p=p.and.(index(ch,s(i:i)).ne.0)
 10   continue
      if(.not.p)return
      n=index(s,'.')
      if(n.eq.0)then
          n=index(s,'e')
          if(n.eq.0)n=index(s,'E')
          if(n.eq.0)n=index(s,'d')
          if(n.eq.0)n=index(s,'D')
          if(n.eq.0)then
             s=s(1:l)//'.'
          else
            t=s(n:l)
            s=s(1:(n-1))//'.'//t
          endif
          l=l+1
      endif
      write(c,'(a30)')s(1:l)
      read(c,'(g30.23)')v
      ier=0
      return
      end
c+ lenstr   nonblank length of string
      function lenstr(s)
c  length of the substring of s obtained by deleting all
c  trailing blanks from s.  thus the length of a string
c  containing only blanks will be 0.
      character   s*(*)
      lenstr=0
      n=len(s)
      do 10 i=n,1,-1
      if(s(i:i) .ne. ' ')then
         lenstr=i
         return
      endif
10    continue
      return
      end
c+ orthgm  generate a rotation matrix (orthogonal) from axis and angle
      subroutine orthgm(x,t,a)
      implicit real*8(a-h,o-z)
c   a is the rotation matrix for column vectors
c   x-axis vector
c   t-rotation angle
c   a-output 3 by 3 matrix
c References:
c  (1) Rotations, James Emery, January 2004, (rotations.tex)
c  (2) A Note on Rotation Matrices, Jay P Fillmore,
c      IEEE Computer Graphics and Applications, February, 1984.
c  (3) Applied Linear Algebra, B. Noble, 1969.
      real*8 id(3,3),l(3,3),l2(3,3),a(3,3),x(3)
      real*8 lambda
      data (id(1,j),j=1,3)/1.d0,0.d0,0.d0/
      data (id(2,j),j=1,3)/0.d0,1.d0,0.d0/
```

```
      data (id(3,j),j=1,3)/0.d0,0.d0,1.d0/
      lambda=sqrt(x(1)**2+x(2)**2+x(3)**2)
      do i=1,3
       l(i,i)=0.
      enddo
      l(1,2)=-x(3)
      l(1,3)=x(2)
      l(2,3)=-x(1)
      l(2,1)=-l(1,2)
      l(3,1)=-l(1,3)
      l(3,2)=-l(2,3)
      call matm(l,3,3,3,l,3,3,l2,3)
      c1=sin(t)/lambda
      c2=(1.-cos(t))/(lambda**2)
      do i=1,3
       do j=1,3
        a(i,j)=id(i,j)+c1*l(i,j)+c2*l2(i,j)
       enddo
      enddo
      return
      end
c+ matm  matrix multiplication
      subroutine matm(a,ia,m,n,b,ib,l,c,ic)
        implicit real*8(a-h,o-z)
c arguments
c  a-matrix
c ia-row dimension of a in calling program
c  m-number of rows of a
c  n-number of columns of a
c  b-matrix
c ib-row dimension of b in calling program
c  l-number of columns of b
c  c-product matrix: c=a*b
c ic-row dimension of c in calling program
c
      dimension a(ia,*),b(ib,*),c(ic,*)
c     c=a*b
      do 10 i=1,m
      do 10 j=1,l
      c(i,j)=0.
      do 5 k=1,n
5     c(i,j)=c(i,j)+a(i,k)*b(k,j)
10    continue
      return
      end
c
c+ det3   determinant of a 3 by 3 matrix
       subroutine det3(b,ib,det)
c b-3 by 3 matrix
c ib-row dimension of b
c det-computed determinant
       implicit real*8(a-h,o-z)
       dimension b(ib,*)
       det=b(1,1)*(b(2,2)*b(3,3)-b(2,3)*b(3,2))
       det=det-b(1,2)*(b(2,1)*b(3,3)-b(2,3)*b(3,1))
```

```
      det=det+b(1,3)*(b(2,1)*b(3,2)-b(2,2)*b(3,1))
      return
      end
```

6 Running Some Examples

Example 1:

We will compute the rotation matrix for a 30 degree rotation about the z axis. This is the matrix

$$\mathbf{A} = \begin{bmatrix} \cos(\pi/6) & -\sin(\pi/6) & 0 \\ \sin(\pi/6) & \cos(\pi/6) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We also shall show that if we interchange a pair of column vectors, the matrix becomes an improper rotation with determinant -1. This matrix is improper because it will convert a basis for a right handed coordinate system into a left handed coordinate basis.

We run program **orthgm** and compute:

```
 Enter an angle t (degrees), -180 <= t <= 180
Angle  =   30.000000     degrees
 Enter a direction vector
Direction=    .00000000       .00000000       1.0000000
 rotation matrix a =
  .86602540      -.50000000       .00000000
  .50000000       .86602540       .00000000
  .00000000       .00000000       1.0000000
 determinant=   1.0000000000000
 Computed rotation axis and angle of matrix a
 axis=   .00000000000000       .00000000000000       1.0000000000000
 angle=    30.000000000000

 Now we interchange columns 1 and 2 of matrix a
 to create an orthogonal matrix,
 which is an improper rotation matrix.
 a =
 -.50000000       .86602540       .00000000
  .86602540       .50000000       .00000000
  .00000000       .00000000       1.0000000
 determinant=  -1.0000000000000
 Attempt to compute an axis and angle
 Computed axis and angle of this improper rotation
 axis=  -.50000000000000       .86602540378444       .00000000000000
 angle=    90.000000000000
```

Although the computation on the improper matrix to compute a rotation angle and axis is carried out, the result has no meaning.

Example 2:

Now we shall compute an example where the result is not so obvious We shall compute the rotation matrix for a 65 degree rotation about the axis in vector direction $\mathbf{i} + \mathbf{j} + \mathbf{k}$. We run the program and compute:

```
 Enter an angle t (degrees), -180 <= t <= 180
Angle  =   65.000000     degrees
 Enter a direction vector
Direction=    1.0000000       1.0000000       1.0000000
 rotation matrix a =
  .61507884      -.33079647       .71571762
  .71571762       .61507884      -.33079647
 -.33079647       .71571762       .61507884
 determinant=   1.0000000000000
 Computed rotation axis and angle of matrix a
 axis=   .57735026918963       .57735026918963       .57735026918963
 angle=    65.000000000000

 Now we interchange columns 1 and 2 of matrix a
 to create an orthogonal matrix,
 which is an improper rotation matrix.
 a =
 -.33079647       .61507884       .71571762
  .61507884       .71571762      -.33079647
  .71571762      -.33079647       .61507884
 determinant=  -1.0000000000000
 Attempt to compute an axis and angle
 Computed axis and angle of this improper rotation
 axis=  -.33079646539450       .61507884116047       .71571762423403
 angle=    90.000000000000
```

Exercise: Write a program that computes a matrix, which rotates some specified vector to the x-axis. This can be used in mapping a coordinate bases to a standard basis

7 Bibliography

These items are included because, either they are referred to, or they have material on exponentials of operators or exponentials of matrices.

[1] Bachman George, **Elements of Abstract Harmonic Analysis**, Academic Press, 1964.

[2] Bellman Richard, **Stability Theory of Differential Equations**, Dover, 1953

[3] Brauer Fred, Nohel John A, **The Qualitative Theory of Ordinary Differential Equations**, Dover, 1969.

[4] Douglas Ronald G, **Banach Algebra Techniques In Operator Theory**, Academic Press, 1972.

[5] Emery James D, **Topics In Linear Algebra and Its Applications**, 2015, www.stemsoc.org/je/lineara.tex, www.stemsoc.org/je/lineara.pdf.

[5] Filmore Jay, **A Note On Rotation Matrices**, IEEE Computer Graphics and Applications, February 1984, (Zerox Copy: Emery Papers, number 365).

[6] Kolmogorov and Fomin, **Functional Analysis**.

[7] Moler Clive, Van Loan Charles, **Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twentyfive Years Later**, SIAM Review Vol. 54, No. 1, pp3-49, March 2003.

[8] Naimark, **Normed Rings**.

[9] Noble Ben, **Applied Linear Algebra**, Prentice-Hall, 1969.

[10] Taylor A E, **An Introduction to Functional Analysis**.