Matrix and Vector Algebra (matrix.c)

Technical Note

Document XXX
Version: Draft
Authors: Mark Pedley and Michael Stanley
Date: xx Aug 2014

Table of Contents

Glossary

$\boldsymbol{A}, \boldsymbol{B}$	General matrices
$\boldsymbol{A}^T$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of matrix $\boldsymbol{A}$
$\lvert\boldsymbol{A}\rvert$	Determinant of matrix $\boldsymbol{A}$
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{R}$	Rotation matrix
$\boldsymbol{R_n}(\eta)$	Rotation matrix about axis $\hat{\boldsymbol{n}}$ by angle η
$\boldsymbol{\beta}_i$	*i-th* eigenvector
η, ζ	Rotation angles
$\Delta\eta, \Delta\zeta$	Small rotation angles
λ_i	*i-th* eigenvalue
$\boldsymbol{\Lambda}$	Diagonal eigenvalue matrix
$\boldsymbol{X}$	Matrix of column eigenvectors
ϕ	Jacobi rotation angle

1 Introduction

1.1 Summary

This Application Note provides an introduction to matrix algebra in section 2. With this introduction, the matrix functions in matrix.c should be self-explanatory with the exception of:

a) the eigensolver function eigencompute which uses the Jacobi rotation algorithm documented in section 3

b) the rotation matrix re-normalization function fmatrixAeqRenormRotA which is documented in section 4.

1.2 Functions

Function	Description
void f3x3matrixAeqI(float A[][3]);	Set the matrix A(3 by 3) to the identity matrix I(3)
void fmatrixAeqI(float *A[], int16 rc);	Set the matrix A(rc by rc) to the identity matrix I(rc)
void f3x3matrixAeqScalar(float A[][3], float Scalar);	Sets the matrix A(3 by 3) to the constant value Scalar. Always called in practice with Scalar=0.0F to zero the matrix.
void f3x3matrixAeqInvSymB(float A[][3], float B[][3]);	Sets A(3 by 3) to the inverse of the symmetric matrix B(3x3)
void f3x3matrixAeqAxScalar(float A[][3], float Scalar);	A(3 by 3) = A(3 by 3) * Scalar Used for normalization purposes.
void f3x3matrixAeqMinusA(float A[][3]);	Negates all elements of the matrix A(3 by 3)
float f3x3matrixDetA(float A[][3]);	Returns the determinant of the matrix A(3 by 3)
void eigencompute(float A[][10], float eigval[], float eigvec[][10], int8 n);	Computes the eigenvalues and eigenvectors of the square matrix m(n by n) stored in the upper left of a 10x10 array.
void fmatrixAeqInvA(float *A[], int8 iColInd[], int8 iRowInd[], int8 iPivot[], int8 isize);	Computes the in-site inverse of the matrix A(isize by isize)
void fmatrixAeqRenormRotA(float A[][3]);	Function removes rounding errors from the rotation matrix A(3 by 3) and re-constructs an exact orthonormal rotation matrix.

2 Primer on Matrix and Vector Algebra

2.1 Matrix Identities

For matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of arbitrary dimension:

$$(\boldsymbol{A}^T)^T = \boldsymbol{A} \quad \text{Eq 2.1.1}$$

$$(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T \quad \text{Eq 2.1.2}$$

$$(\boldsymbol{A}+\boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T \quad \text{Eq 2.1.3}$$

If the product $\boldsymbol{AB}$ is a scalar:

$$\boldsymbol{AB} = (\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T \quad \text{Eq 2.1.4}$$

If $\boldsymbol{A}$ and $\boldsymbol{B}$ are square and invertible, the inverse of their product satisfies:

$$(\boldsymbol{AB})^{-1}\boldsymbol{AB} = \boldsymbol{I} \Rightarrow (\boldsymbol{AB})^{-1}\boldsymbol{ABB}^{-1}\boldsymbol{A}^{-1} = \boldsymbol{IB}^{-1}\boldsymbol{A}^{-1} \Rightarrow (\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \quad \text{Eq 2.1.5}$$

The inverse and transpose operations commute for an invertible square matrix:

$$\boldsymbol{AA}^{-1} = \boldsymbol{I} \Rightarrow (\boldsymbol{AA}^{-1})^T = \boldsymbol{I}^T = \boldsymbol{I} \Rightarrow (\boldsymbol{A}^{-1})^T\boldsymbol{A}^T = \boldsymbol{I} \Rightarrow (\boldsymbol{A}^{-1})^T\boldsymbol{A}^T(\boldsymbol{A}^T)^{-1} = \boldsymbol{I}(\boldsymbol{A}^T)^{-1} \Rightarrow (\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1} \quad \text{Eq 2.1.6}$$

An orthogonal matrix is a real square matrix $\boldsymbol{A}$ whose columns and columns are orthonormal. A consequence is that the transpose of an orthonormal matrix is its inverse.

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I} \Rightarrow \boldsymbol{A}^T = \boldsymbol{A}^{-1} \quad \text{Eq 2.1.7}$$

2.2 Rotation Matrices

Rotation matrices are square matrices with determinant of +1 and with orthonormal columns and rows.

$$|\boldsymbol{R}| = 1 \quad \text{Eq 2.2.1}$$

$$\boldsymbol{RR}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I} \quad \text{Eq 2.2.2}$$

Rotation matrices about a given axis with specified angle can be defined as either the rotation matrix rotating the coordinate frame or rotating a vector in the coordinate frame. The two are related by having the same axis but opposite rotation angles. The convention used in Freescale's Sensor fusion Toolbox is that the coordinate system is rotated about the rotation vector.

Rotation matrices do not commute and the order of rotations must always be specified.

$$\boldsymbol{R}_1\boldsymbol{R}_2 \neq \boldsymbol{R}_2\boldsymbol{R}_1 \quad \text{Eq 2.2.3}$$

Euler's rotation theorem states that any sequence of rotation matrices can be represented by a single rotation matrix with axis and angle to be determined. The general matrix for a rotation of a coordinate frame about (normalized) axis $\hat{\boldsymbol{n}}$ by angle η is the Rodrigues matrix:

$$\boldsymbol{R}_{\boldsymbol{n}}(\eta) = \begin{pmatrix} \hat{n}_x{}^2 + (1-\hat{n}_x{}^2)cos\eta & \hat{n}_x\hat{n}_y(1-cos\eta) + \hat{n}_z sin\eta & \hat{n}_x\hat{n}_z(1-cos\eta) - \hat{n}_y sin\eta \\ \hat{n}_x\hat{n}_y(1-cos\eta) - \hat{n}_z sin\eta & \hat{n}_y{}^2 + (1-\hat{n}_y{}^2)cos\eta & \hat{n}_y\hat{n}_z(1-cos\eta) + \hat{n}_x sin\eta \\ \hat{n}_x\hat{n}_z(1-cos\eta) + \hat{n}_y sin\eta & \hat{n}_y\hat{n}_z(1-cos\eta) - \hat{n}_x sin\eta & \hat{n}_z{}^2 + (1-\hat{n}_z{}^2)cos\eta \end{pmatrix} \quad \text{Eq 2.2.4}$$

Inspection of equation 2.2.4 shows that the rotation matrix is symmetric when $sin\eta = 0$ which occurs at $\eta = 0$ and $sin\eta = \pi$ (180°) rotations.

$$\boldsymbol{R_n}(\eta = 0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 2.2.5}$$

$$\boldsymbol{R_n}(\eta = \pi) = \begin{pmatrix} 2\hat{n}_x{}^2 - 1 & 2\hat{n}_x\hat{n}_y & 2\hat{n}_x\hat{n}_z \\ 2\hat{n}_x\hat{n}_y & 2\hat{n}_y{}^2 - 1 & 2\hat{n}_y\hat{n}_z \\ 2\hat{n}_x\hat{n}_z & 2\hat{n}_y\hat{n}_z & 2\hat{n}_z{}^2 - 1 \end{pmatrix} \quad \text{Eq 2.2.6}$$

Evaluating $\boldsymbol{R}$ for $\hat{\boldsymbol{n}}$ aligned along the *x*, *y* and *z* axes produces the Cartesian rotation matrices:

$$\boldsymbol{R}_x(\eta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\eta & sin\eta \\ 0 & -sin\eta & cos\eta \end{pmatrix} \quad \text{Eq 2.2.7}$$

$$\boldsymbol{R}_y(\eta) = \begin{pmatrix} cos\eta & 0 & -sin\eta \\ 0 & 1 & 0 \\ sin\eta & -0 & cos\eta \end{pmatrix} \quad \text{Eq 2.2.8}$$

$$\boldsymbol{R}_z(\eta) = \begin{pmatrix} cos\eta & sin\eta & 0 \\ -sin\eta & cos\eta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 2.2.9}$$

For a small rotation angles $\Delta\eta$, the general rotation matrix simplifies to:

$$\boldsymbol{R_n}(\Delta\eta) = \begin{pmatrix} 1 & \hat{n}_z\Delta\eta & -\hat{n}_y\Delta\eta \\ -\hat{n}_z\Delta\eta & 1 & \hat{n}_x\Delta\eta \\ \hat{n}_y\Delta\eta & -\hat{n}_x\Delta\eta & 1 \end{pmatrix} \quad \text{Eq 2.2.10}$$

Rotation matrices do commute in the limit of small angles $\Delta\eta$ and $\Delta\zeta$:

$$\boldsymbol{R_n}(\Delta\eta)\boldsymbol{R_m}(\Delta\zeta) = \begin{pmatrix} 1 & \hat{n}_z\Delta\eta & -\hat{n}_y\Delta\eta \\ -\hat{n}_z\Delta\eta & 1 & \hat{n}_x\Delta\eta \\ \hat{n}_y\Delta\eta & -\hat{n}_x\Delta\eta & 1 \end{pmatrix} \begin{pmatrix} 1 & \hat{m}_z\Delta\zeta & -\hat{m}_y\Delta\zeta \\ -\hat{m}_z\Delta\zeta & 1 & \hat{m}_x\Delta\zeta \\ m_y\Delta\zeta & -\hat{m}_x\Delta\zeta & 1 \end{pmatrix} \quad \text{Eq 2.2.11}$$

$$= \begin{pmatrix} 1 & \hat{m}_z\Delta\zeta + \hat{n}_z\Delta\eta & -\hat{m}_y\Delta\zeta - \hat{n}_y\Delta\eta \\ -\hat{m}_z\Delta\zeta - \hat{n}_z\Delta\eta & 1 & \hat{m}_x\Delta\zeta + \hat{n}_x\Delta\eta \\ m_y\Delta\zeta + \hat{n}_y\Delta\eta & -\hat{m}_x\Delta\zeta - \hat{n}_x\Delta\eta & 1 \end{pmatrix} = \boldsymbol{R_m}(\Delta\zeta)\boldsymbol{R_n}(\Delta\eta) \quad \text{Eq 2.2.12}$$

2.3 Eigenvectors and Eigenvalues

The *i-th* column eigenvector $\boldsymbol{\beta}_i$ and eigenvalue λ_i of any square matrix $\boldsymbol{A}$ are defined as satisfying:

$$\boldsymbol{A}\boldsymbol{\beta}_i = \lambda_i\boldsymbol{\beta}_i \quad \text{Eq 2.3.1}$$

It is commonplace to create a eigenvector matrix $\boldsymbol{X}$ from the N individual column eigenvectors $\boldsymbol{\beta}_i$:

$$\boldsymbol{X} = (\boldsymbol{\beta}_0 \quad \boldsymbol{\beta}_1 \quad \ldots \quad \boldsymbol{\beta}_{N-1}) \quad \text{Eq 2.3.2}$$

Equation 2.3.1 can then be written in the form:

$$\boldsymbol{AX} = \boldsymbol{X\Lambda} \qquad \text{Eq 2.3.3}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_0 & 0 & \dots & 0 \\ 0 & \lambda_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_{N-1} \end{pmatrix} \qquad \text{Eq 2.3.4}$$

The rotation axis $\hat{\boldsymbol{n}}$ of a rotation matrix is, by definition, invariant under rotation and is therefore the normalized eigenvector corresponding to eigenvalue 1:

$$\boldsymbol{R\hat{n}} = \hat{\boldsymbol{n}} \qquad \text{Eq 2.3.5}$$

This can be confirmed for the general rotation matrix of equation 2.2.4:

$$\boldsymbol{R\hat{n}} = \begin{pmatrix} \hat{n}_x{}^2 + (1 - \hat{n}_x{}^2)cos\eta & \hat{n}_x\hat{n}_y(1 - cos\eta) + \hat{n}_z sin\eta & \hat{n}_x\hat{n}_z(1 - cos\eta) - \hat{n}_y sin\eta \\ \hat{n}_x\hat{n}_y(1 - cos\eta) - \hat{n}_z sin\eta & \hat{n}_y{}^2 + (1 - \hat{n}_y{}^2)cos\eta & \hat{n}_y\hat{n}_z(1 - cos\eta) + \hat{n}_x sin\eta \\ \hat{n}_x\hat{n}_z(1 - cos\eta) + \hat{n}_y sin\eta & \hat{n}_y\hat{n}_z(1 - cos\eta) - \hat{n}_x sin\eta & \hat{n}_z{}^2 + (1 - \hat{n}_z{}^2)cos\eta \end{pmatrix} \begin{pmatrix} \hat{n}_x \\ \hat{n}_y \\ \hat{n}_z \end{pmatrix} \qquad \text{Eq 2.3.6}$$

$$= \begin{pmatrix} \hat{n}_x(\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2) - (\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2 - 1) \\ \hat{n}_y(\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2) - (\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2 - 1) \\ \hat{n}_z(\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2) - (\hat{n}_x{}^2 + \hat{n}_y{}^2 + \hat{n}_z{}^2 - 1) \end{pmatrix} = \begin{pmatrix} \hat{n}_x \\ \hat{n}_y \\ \hat{n}_z \end{pmatrix} \qquad \text{Eq 2.3.7}$$

2.4 Vector Identities

Repeated application of Pythagoras' theorem relates the transpose product of any vector $\boldsymbol{a}$ with itself to its magnitude squared.

$$\boldsymbol{a}^T\boldsymbol{a} = |\boldsymbol{a}|^2 \qquad \text{Eq 2.4.1}$$

The transpose product of two vectors is often written as the scalar or dot product. The cosine rule of trigonometry relates the scalar product to the cosine of the angle α between the two vectors as:

$$|\boldsymbol{a} - \boldsymbol{b}|^2 = |\boldsymbol{a}|^2 + |\boldsymbol{b}|^2 - 2|\boldsymbol{a}||\boldsymbol{b}|cos\alpha \Rightarrow \boldsymbol{a}.\boldsymbol{a} + \boldsymbol{b}.\boldsymbol{b} - 2\boldsymbol{a}.\boldsymbol{b} = |\boldsymbol{a}|^2 + |\boldsymbol{b}|^2 - 2|\boldsymbol{a}||\boldsymbol{b}|cos \qquad \text{Eq 2.4.2}$$

$$\Rightarrow \boldsymbol{a}.\boldsymbol{b} = \boldsymbol{a}^T\boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}|cos\alpha \qquad \text{Eq 2.4.3}$$

The vector or cross product between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined to be a new vector with magnitude $absin\alpha$ in a direction $\hat{\boldsymbol{n}}$ orthogonal to both $\boldsymbol{a}$ and $\boldsymbol{b}$:

$$\boldsymbol{a} \times \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}|\hat{\boldsymbol{n}}sin\alpha \qquad \text{Eq 2.4.4}$$

In Cartesian coordinates, the vector product can be written as:

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{vmatrix} \hat{\boldsymbol{\imath}} & \hat{\boldsymbol{\jmath}} & \hat{\boldsymbol{k}} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix} \qquad \text{Eq 2.4.5}$$

An alternative formulation of the vector product is matrix multiplication by the 3x3 matrix $(\boldsymbol{a} \times)$ defined as:

$$(\boldsymbol{a}\times) = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \qquad \text{Eq 2.4.6}$$

A useful result for the vector product of three vectors is:

$$\boldsymbol{a}\times(\boldsymbol{b}\times\boldsymbol{c}) = (\boldsymbol{a}.\boldsymbol{c})\boldsymbol{b} - (\boldsymbol{a}.\boldsymbol{b})\boldsymbol{c} \qquad \text{Eq 2.4.7}$$

3 EigenAnalysis by Jacobi Algorithm

3.1 Introduction

This section documents the Jacobi algorithm underlying the function eigencompute which computes the eigenvalues and eigenvectors of a symmetric matrix by successive diagonalization by Givens rotation matrices.

The matrix of real eigenvectors $\boldsymbol{X}$ and diagonal matrix of real eigenvalues $\boldsymbol{\Lambda}$ of the real symmetric matrix $\boldsymbol{A}$ satisfy:

$$\boldsymbol{AX} = \boldsymbol{X\Lambda} \Rightarrow \boldsymbol{A} = \boldsymbol{X\Lambda X}^{-1} \Rightarrow \boldsymbol{\Lambda} = \boldsymbol{X}^{-1}\boldsymbol{AX} \qquad \text{Eq 3.1.1}$$

Pre-and post multiplying equation 3.1.1 by a sequence of plane rotation matrices ${\boldsymbol{R}_i}^{-1}$ and $\boldsymbol{R}_i$ (termed Givens rotation matrices) selected to zero the off-diagonal elements and diagonalize $\boldsymbol{A}$ gives:

$$\boldsymbol{\Lambda} = {\boldsymbol{R}_N}^{-1} \ldots {\boldsymbol{R}_2}^{-1}{\boldsymbol{R}_1}^{-1}\boldsymbol{A}\boldsymbol{R}_1\boldsymbol{R}_1\boldsymbol{R}_2 \ldots \boldsymbol{R}_N = {\boldsymbol{R}_N}^{-1} \ldots {\boldsymbol{R}_2}^{-1}{\boldsymbol{R}_1}^{-1}\boldsymbol{X\Lambda X}^{-1}\boldsymbol{R}_1\boldsymbol{R}_2 \ldots \boldsymbol{R}_N \qquad \text{Eq 3.1.2}$$

$$\Rightarrow \boldsymbol{X} = \boldsymbol{R}_1\boldsymbol{R}_2 \ldots \boldsymbol{R}_N \qquad \text{Eq 3.1.3}$$

The eigenvalues of the matrix $\boldsymbol{A}$ are then the elements of the diagonal matrix $\boldsymbol{\Lambda}$ derived by zeroing off-diagonal elements in $\boldsymbol{A}$ and the matrix of eigenvectors $\boldsymbol{X}$ is the product of the sequence of matrices $\boldsymbol{R}_i$ used to perform the diagonalization. This diagonalization strategy is used by the Jacobi rotation algorithm successively applying Givens rotation matrices.

3.2 Givens Rotation Matrix

The Givens matrix $\boldsymbol{R}_{pq}$ for rotation angle ϕ has form:

$$\boldsymbol{R}_{pq} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & \ldots & cos\phi & \ldots & sin\phi & \ldots & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & \ldots & -sin\phi & \ldots & cos\phi & \ldots & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{Eq 3.2.1}$$

All the diagonal elements are zero except for the two elements at positions p, p and q, q. All off-diagonal elements are zero except for the two elements at positions p, q and q, p. The Givens rotation matrix is orthonormal as required for a rotation matrix.

A general matrix $\boldsymbol{A}$ is transformed by pre- and post-multiplication by the Givens rotation matrix with non-zero elements in rows p and q as:

$$\boldsymbol{A}' = {\boldsymbol{R}_{pq}}^{T}\boldsymbol{A}\boldsymbol{R}_{pq} \qquad \text{Eq 3.2.2}$$

The elements in $\boldsymbol{A}$ changed by this operation are:

$$\boldsymbol{A}' = \begin{pmatrix} \cdots & \cdots & a'_{0,p} & \cdots & a'_{0,q} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a'_{p,0} & \cdots & a'_{p,p} & \cdots & a'_{p,q} & \cdots & a'_{p,n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a'_{q,0} & \cdots & a'_{q,p} & \cdots & a'_{q,q} & \cdots & a'_{q,n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & a'_{n-1,p} & \cdots & a'_{n-1,q} & \cdots & \cdots \end{pmatrix}$$ Eq 3.2.3

The changed elements of equation 3.2.3 are:

$$a'_{r,p} = a_{r,p}cos\phi - a_{r,q}sin\phi \ (r \neq p, r \neq q)$$ Eq 3.2.4

$$a'_{r,q} = a_{r,q}cos\phi + a_{r,p}sin\phi \ (r \neq p, r \neq q)$$ Eq 3.2.5

$$a'_{p,p} = a_{p,p}cos^2\phi + a_{q,q}sin^2\phi - 2a_{p,q}sin\phi cos\phi$$ Eq 3.2.6

$$a'_{q,q} = a_{p,p}sin^2\phi + a_{q,q}cos^2\phi + 2a_{p,q}sin\phi cos\phi$$ Eq 3.2.7

$$a'_{p,q} = a_{p,q}(cos^2\phi - sin^2\phi) + (a_{p,p} - a_{q,q})sin\phi cos\phi$$ Eq 3.2.8

3.3 Determining the Givens Rotation Angle

The required rotation angle ϕ is that which zeroes out element $a'_{p,q}$:

$$\frac{(cos^2\phi - sin^2\phi)}{sin\phi cos\phi} = \frac{(a_{q,q} - a_{p,p})}{a_{p,q}}$$

Eq 3.3.1

Standard trigonometry identities allow the cotangent of twice the rotation angle 2ϕ to be written as:

$$cot(2\phi) = \frac{cos(2\phi)}{sin(2\phi)} = \frac{cos^2\phi - sin^2\phi}{2sin\phi cos\phi}$$

Eq 3.3.2

Combining equations 3.3.1 and 3.3.2 gives the rotation angle as:

$$cot(2\phi) = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}$$

Eq 3.3.3

$$cot(2\phi) = \frac{cos^2\phi - sin^2\phi}{2sin\phi cos\phi} = \frac{1 - tan^2\,\phi}{2tan\phi} \Rightarrow tan^2\,\phi + 2cot(2\phi)\,tan\,\phi - 1 = 0$$

Eq 3.3.4

$$\Rightarrow tan\,\phi = -cot(2\phi) \pm \sqrt{cot^2(2\phi) + 1}$$

Eq 3.3.5

Taking the positive square root for $tan\,\phi$ gives:

$$tan\,\phi = -cot(2\phi) + \sqrt{cot^2(2\phi) + 1} = \frac{\left(-cot(2\phi) + \sqrt{cot^2(2\phi) + 1}\right)\left(-cot(2\phi) - \sqrt{cot^2(2\phi) + 1}\right)}{\left(-cot(2\phi) - \sqrt{cot^2(2\phi) + 1}\right)}$$

Eq 3.3.6

$$= \frac{-1}{-cot(2\phi) - \sqrt{cot^2(2\phi)+1}}$$

Eq 3.3.7

Taking the negative square root gives:

$$tan\,\phi = -cot(2\phi) - \sqrt{cot^2(2\phi)+1} = \frac{\left(-cot(2\phi) - \sqrt{cot^2(2\phi)+1}\right)\left(-cot(2\phi) + \sqrt{cot^2(2\phi)+1}\right)}{\left(-cot(2\phi) + \sqrt{cot^2(2\phi)+1}\right)}$$

Eq 3.3.8

$$= \frac{-1}{-cot(2\phi) + \sqrt{cot^2(2\phi)+1}}$$

Eq 3.3.9

For θ negative, the smaller magnitude of the two solutions is:

$$tan\,\phi = \frac{-1}{\left(-cot(2\phi) + \sqrt{cot^2(2\phi)+1}\right)} = \frac{sgn\left(cot(2\phi)\right)}{\left(|cot(2\phi)| + \sqrt{cot^2(2\phi)+1}\right)}$$

Eq 3.3.10

For θ positive, the smaller magnitude of the two solutions is:

$$tan\,\phi = \frac{1}{\left(cot(2\phi) + \sqrt{cot^2(2\phi)+1}\right)} = \frac{sgn\left(cot(2\phi)\right)}{\left(|cot(2\phi)| + \sqrt{cot^2(2\phi)+1}\right)}$$

Eq 3.3.11

In both cases:

$$tan\,\phi = \frac{sgn\left(cot(2\phi)\right)}{\left(|cot(2\phi)| + \sqrt{cot^2(2\phi)+1}\right)}$$

Eq 3.3.12

If θ is so large that $cot(2\phi)$ squared would overflow, the alternative is:

$$tan\,\phi = \frac{sgn\left(cot(2\phi)\right)}{2|cot(2\phi)|} = \frac{-1}{2cot(2\phi)}$$

Eq 3.3.13

A trigonometric identity used later is:

$$\frac{sin\phi}{1+cos\phi} = \frac{2sin\left(\frac{\phi}{2}\right)cos\left(\frac{\phi}{2}\right)}{2cos^2\left(\frac{\phi}{2}\right)} = tan\left(\frac{\phi}{2}\right)$$

Eq 3.3.14

3.4 The Jacobi Algorithm

To avoid roundoff error, iterative updates are used for equations 3.2.4 to 3.2.8:

Equation 3.2.8:

By definition, the rotation angle ϕ is selected so that equation 3.2.8 gives zero $a'_{p,q}$.

$$a'_{p,q} = 0$$

Eq 3.4.1

$$\Rightarrow a_{p,q}(cos^2\phi - sin^2\phi) + (a_{p,p} - a_{q,q})sin\phi cos\phi = 0$$ Eq 3.4.2

$$\Rightarrow a_{q,q}sin^2\phi = \frac{sin\phi\{a_{p,q}(cos^2\phi - sin^2\phi) + a_{p,p}sin\phi cos\phi\}}{cos\phi}$$

Eq 3.4.3

Equation 3.2.6:

$$a'_{p,p} = a_{p,p}cos^2\phi + a_{q,q}sin^2\phi - 2a_{p,q}sin\phi cos\phi$$ Eq 3.4.4

Substituting equation 3.4.3 gives:

$$a'_{p,p} = a_{p,p}cos^2\phi + sin\phi\left\{\frac{a_{p,q}(cos^2\phi - sin^2\phi) + a_{p,p}sin\phi cos\phi}{cos\phi}\right\} - 2a_{p,q}sin\phi cos\phi$$

Eq 3.4.5

$$= a_{p,p} + \left(\frac{a_{p,q}sin\phi(cos^2\phi - sin^2\phi) - 2a_{p,q}sin\phi cos^2\phi}{cos\phi}\right)$$

Eq 3.4.6

$$\Rightarrow a'_{p,p} = a_{p,p} - a_{p,q}sin\phi\left(\frac{sin^2\phi + cos^2\phi}{cos\phi}\right) = a_{p,p} - a_{p,q}\,tan\,\phi$$

Eq 3.4.7

Equation 3.2.7:

By definition of the rotation angle which zeroes equation 3.2.8:

$$a_{p,q}(cos^2\phi - sin^2\phi) + (a_{p,p} - a_{q,q})sin\phi cos\phi$$ Eq 3.4.8

$$\Rightarrow a_{p,p} = a_{q,q} - \left\{\frac{(cos^2\phi - sin^2\phi)a_{p,q}}{sin\phi cos\phi}\right\}$$

Substituting into equation 3.2.7 gives: Eq 3.4.9

$$a'_{q,q} = sin^2\phi\left\{a_{q,q} - \left\{\frac{(cos^2\phi - sin^2\phi)a_{p,q}}{sin\phi cos\phi}\right\}\right\} + a_{q,q}cos^2\phi + 2a_{p,q}sin\phi cos\phi$$

Eq 3.4.10

$$\Rightarrow a'_{q,q} = a_{q,q} + sin\phi\left(\frac{sin^2\phi + cos^2\phi}{cos\phi}\right)a_{p,q} = a_{q,q} + a_{p,q}\,tan\,\phi$$

Eq 3.4.11

Equation 3.2.4:

$$a'_{r,p} = a_{r,p}cos\phi - a_{r,q}sin\phi = a_{r,p} - a_{r,p}(1 - cos\phi) - a_{r,q}sin\phi$$ Eq 3.4.12

$$= a_{r,p} - sin\phi\left\{a_{r,q} + \frac{(1 - cos\phi)(1 + cos\phi)a_{r,p}}{sin\phi(1 + cos\phi)}\right\} = a_{r,p} - sin\phi\left\{a_{r,q} + \frac{a_{r,p}sin^2\phi}{sin\phi(1 + cos\phi)}\right\}$$

Eq 3.4.13

Substituting equation 3.3.14 gives:

$$a'_{r,p} = a_{r,p} - sin\phi\left(a_{r,q} + a_{r,p}tan\left(\frac{\phi}{2}\right)\right)$$

Eq 3.4.14

Equation 3.2.5:

$$a'_{r,q} = a_{r,q}cos\phi + a_{r,p}sin\phi = a_{r,q} - a_{r,q}(1 - cos\phi) + a_{r,p}sin\phi$$

Eq 3.4.15

$$= a_{r,q} + sin\phi\left\{a_{r,p} - \frac{(1 - cos\phi)(1 + cos\phi)a_{r,q}}{sin\phi(1 + cos\phi)}\right\} = a_{r,q} + sin\phi\left\{a_{r,p} - \frac{a_{r,q}sin^2\phi}{sin\phi(1 + cos\phi)}\right\}$$

Eq 3.4.16

Substituting equation 3.3.14 gives:

$$a'_{r,q} = a_{r,q} + sin\phi\left(a_{r,p} - a_{r,q}tan\left(\frac{\phi}{2}\right)\right)$$

Eq 3.4.17

4 Normalization of a Rotation Matrix

The rows and columns of a rotation matrix are orthonormal. After several thousand iterative rotations, a rotation matrix can become degraded as a result of numerical rounding errors. The simple method below permits the re-normalization of a rotation matrix and is used in function fmatrixAeqRenormRotA.

Given an initial rotation matrix $\boldsymbol{R}_0$ with column vectors $\boldsymbol{u}_0$, $\boldsymbol{v}_0$, $\boldsymbol{w}_0$:

$$\boldsymbol{R}_0 = (\boldsymbol{u}_0 \quad \boldsymbol{v}_0 \quad \boldsymbol{w}_0) \qquad \text{Eq 4.1.1}$$

Step 1: The column vector $\boldsymbol{u}$ is the vector $\boldsymbol{u}_0$ normalized:

$$\boldsymbol{u} = \frac{\boldsymbol{u}_0}{|\boldsymbol{u}_0|} \qquad \text{Eq 4.1.2}$$

Step 2: The column vector $\boldsymbol{v}$ is set to:

$$\boldsymbol{v} = \frac{\boldsymbol{v}_0 - (\boldsymbol{u}.\boldsymbol{v}_0)\boldsymbol{u}}{|\boldsymbol{v}_0 - (\boldsymbol{u}.\boldsymbol{v}_0)\boldsymbol{u}|} \qquad \text{Eq 4.1.3}$$

The column vector $\boldsymbol{v}$ is obviously normalized and is easily proved to be orthogonal to $\boldsymbol{u}$:

$$\boldsymbol{u}.\boldsymbol{v} = \frac{\boldsymbol{u}.\boldsymbol{v}_0 - (\boldsymbol{u}.\boldsymbol{v}_0)\boldsymbol{u}.\boldsymbol{u}}{|\boldsymbol{v}_0 - (\boldsymbol{u}.\boldsymbol{v}_0)\boldsymbol{u}|} = \frac{\boldsymbol{u}.\boldsymbol{v}_0(1 - \boldsymbol{u}.\boldsymbol{u})}{|\boldsymbol{v}_0 - (\boldsymbol{u}.\boldsymbol{v}_0)\boldsymbol{u}|} = 0 \qquad \text{Eq 4.1.4}$$

Step 3: The column vector $\boldsymbol{w}$ is set to the unit vector orthogonal to $\boldsymbol{u}$ and $\boldsymbol{v}$:

$$\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v} \qquad \text{Eq 4.1.5}$$

There is no need to explicitly normalize the vector $\boldsymbol{w}$ since $\boldsymbol{u}$ and $\boldsymbol{v}$ are unit vectors and $\boldsymbol{u}$ is orthogonal to $\boldsymbol{v}$.

The re-normalized rotation matrix $\boldsymbol{R}$ is:

$$\boldsymbol{R} = (\boldsymbol{u} \quad \boldsymbol{v} \quad \boldsymbol{w}) \qquad \text{Eq 4.1.6}$$