AN5020

Determining Matrix Eigenvalues and Eigenvectors by Jacobi Algorithm

Rev. 2.0 — 21 June 2016 **Application note**

Document information

Info	Content
Abstract	This application note documents the Jacobi rotation eigenanalysis algorithm in the NXP Sensor Fusion Library software.

Revision history

Document ID	Release date	Supercedes
AN5020 v2.0	20160606	AN5020 v1.0
Modifications:	• Minor changes • The format of this document has been redesigned to comply with the new identity guidelines of NXP Semiconductors. Legal texts have been adapted to the new company name where appropriate.	
AN5020 v1.0	2015 September	—

Contact information

For more information, please visit: http://www.nxp.com

1. Introduction

1.1 Summary

This application note documents the *Jacobi rotation* eigenanalysis algorithm in the *NXP Sensor Fusion Library* software and implemented by the two functions `eigencompute10` and `eigencompute4` in the file matrix.c.

The two functions are identical except for their function headers which specify 10x10 and 4x4 input matrices respectively. This is for software portability with early C standards in which C functions cannot be defined to handle arrays with variable numbers of columns.

The functions are used for a variety of mathematical solutions including magnetic hard and soft iron calibration, precision accelerometer calibration and for taking the square root of a symmetric matrix.

1.2 Terminology

Symbol	Definition
$\boldsymbol{A}$	General square matrix
$\boldsymbol{A}^T$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of matrix $\boldsymbol{A}$
$\boldsymbol{R}_i$	*i-th* Givens rotation matrix
$\boldsymbol{R}_{pq}$	Givens rotation matrix with non-zero elements in row p and column q
$\boldsymbol{X}$	Matrix of column eigenvectors
$\boldsymbol{\beta}_i$	*i-th* eigenvector
λ_i	*i-th* eigenvalue
$\boldsymbol{\Lambda}$	Diagonal eigenvalue matrix
ϕ	Jacobi rotation angle

1.3 Software Functions

Functions	Description	Reference
`void eigencompute10 (float A[][10], float eigval[], float eigvec[][10], int8 n)`	Computes the eigenvalues and eigenvectors of an n by n square matrix stored in the upper left of a 10x10 array.	2
`void eigencompute4 (float A[][4], float eigval[], float eigvec[][4], int8 n)`	Computes the eigenvalues and eigenvectors of an n by n square matrix stored in the upper left of a 4x4 array.	2

2. Eigenanalysis by Jacobi Algorithm

2.1 Introduction

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

$$\boldsymbol{A}\boldsymbol{\beta}_i = \lambda_i \boldsymbol{\beta}_i \tag{1}$$

The NxN matrix $\boldsymbol{X}$ formed from the N individual column eigenvectors $\boldsymbol{\beta}_i$ is:

$$\boldsymbol{X} = (\boldsymbol{\beta}_0 \quad \boldsymbol{\beta}_1 \quad \dots \quad \boldsymbol{\beta}_{N-1}) \tag{2}$$

Equation (1) can then be written in the form:

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda} \tag{3}$$

where $\boldsymbol{\Lambda}$ is the NxN matrix formed from the eigenvalues lying on the diagonal:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_0 & 0 & \dots & 0 \\ 0 & \lambda_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_{N-1} \end{pmatrix} \tag{4}$$

If the inverse matrix $\boldsymbol{X}^{-1}$ exists then equation (3) implies:

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1} \tag{5}$$

$$\boldsymbol{\Lambda} = \boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} \tag{6}$$

A diagonal matrix is unaffected by pre-and post-multiplication by any rotation matrix. Pre- and post-multiplying equation (6) by any inverse and forward rotation matrix $\boldsymbol{R}$ therefore gives:

$$\boldsymbol{R}^{-1}\boldsymbol{\Lambda}\boldsymbol{R} = \boldsymbol{\Lambda} = \boldsymbol{R}^{-1}\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}\boldsymbol{R} \tag{7}$$

The Jacobi algorithm underlying the functions `eigencompute10` and `eigencompute4` computes the eigenvalues and eigenvectors of a symmetric (and therefore square) NxN matrix $\boldsymbol{A}$ by successive pre- and post-multiplication by inverse and forward two-dimensional plane rotation matrices $\boldsymbol{R}_i$ termed Givens rotation matrices designed to obtain the diagonal matrix $\boldsymbol{\Lambda}$:

$$\boldsymbol{\Lambda} = \boldsymbol{R}_N{}^{-1} \dots \boldsymbol{R}_2{}^{-1}\boldsymbol{R}_1{}^{-1}\boldsymbol{A}\boldsymbol{R}_1\boldsymbol{R}_1\boldsymbol{R}_2 \dots \boldsymbol{R}_N \tag{8}$$

$$\Rightarrow \boldsymbol{\Lambda} = \boldsymbol{R}_N{}^{-1} \dots \boldsymbol{R}_2{}^{-1}\boldsymbol{R}_1{}^{-1}(\boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1})\boldsymbol{R}_1\boldsymbol{R}_2 \dots \boldsymbol{R}_N \tag{9}$$

$$\Rightarrow \boldsymbol{X} = \boldsymbol{R}_1\boldsymbol{R}_2 \dots \boldsymbol{R}_N \tag{10}$$

The eigenvectors of a symmetric matrix are orthogonal and another way of interpreting equations (8) through (10) is that the sequence of Givens rotations matrices rotates the matrix of eigenvectors to be aligned with the base vectors of the N dimensional coordinate system.

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.

2.2 Givens Rotation Matrix

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

$$\boldsymbol{R}_{pq} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & cos\phi & \dots & sin\phi & \dots & 0 \\ 0 & \dots & \dots & 1 & \dots & \dots & 0 \\ 0 & \dots & -sin\phi & \dots & cos\phi & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

All the diagonal elements are one 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:

$$\boldsymbol{A}' = \boldsymbol{R}_{pq}{}^T\boldsymbol{A}\boldsymbol{R}_{pq} \tag{12}$$

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

$$\boldsymbol{A}' = \begin{pmatrix} \dots & \dots & a'_{0,p} & \dots & a'_{0,q} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a'_{p,0} & \dots & a'_{p,p} & \dots & a'_{p,q} & \dots & a'_{p,n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a'_{q,0} & \dots & a'_{q,p} & \dots & a'_{q,q} & \dots & a'_{q,n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & a'_{n-1,p} & \dots & a'_{n-1,q} & \dots & \dots \end{pmatrix} \tag{13}$$

The changed elements of equation (13) are:

$$a'_{r,p} = a_{r,p}cos\phi - a_{r,q}sin\phi \; (r \neq p, r \neq q) \quad \textbf{(14)}$$

$$a'_{r,q} = a_{r,q}cos\phi + a_{r,p}sin\phi \; (r \neq p, r \neq q) \quad \textbf{(15)}$$

$$a'_{p,p} = a_{p,p}cos^2\phi + a_{q,q}sin^2\phi - 2a_{p,q}sin\phi cos\phi \quad \textbf{(16)}$$

$$a'_{q,q} = a_{p,p}sin^2\phi + a_{q,q}cos^2\phi + 2a_{p,q}sin\phi cos\phi \quad \textbf{(17)}$$

$$a'_{p,q} = a_{p,q}(cos^2\phi - sin^2\phi) + \left(a_{p,p} - a_{q,q}\right)sin\phi cos\phi \quad \textbf{(18)}$$

2.3 Determining the Givens Rotation Angle

The required rotation angle ϕ is that which zeroes out element $a'_{p,q}$ in equation (18):

$$a'_{p,q} = 0 \Rightarrow \frac{(cos^2\phi - sin^2\phi)}{sin\phi cos\phi} = \frac{\left(a_{q,q} - a_{p,p}\right)}{a_{p,q}} \quad \textbf{(19)}$$

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

$$cot(2\phi) = \frac{cos(2\phi)}{sin(2\phi)} = \frac{cos^2\phi - sin^2\phi}{2sin\phi cos\phi} \quad \textbf{(20)}$$

Combining equations (19) and (20) defines the rotation angle ϕ as:

$$cot(2\phi) = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}} \quad \textbf{(21)}$$

$$cot(2\phi) = \frac{cos^2\phi - sin^2\phi}{2sin\phi cos\phi} = \frac{1 - tan^2\phi}{2tan\phi} \Rightarrow tan^2\phi + 2cot(2\phi)\, tan\,\phi - 1 = 0 \quad \textbf{(22)}$$

$$\Rightarrow tan\,\phi = -cot(2\phi) \pm \sqrt{cot^2(2\phi) + 1} \quad \textbf{(23)}$$

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

$$tan\,\phi = -cot(2\phi) + \sqrt{cot^2(2\phi) + 1} \quad \textbf{(24)}$$

$$= \frac{\left(-cot(2\phi) + \sqrt{cot^2(2\phi) + 1}\right)\left(-cot(2\phi) - \sqrt{cot^2(2\phi) + 1}\right)}{\left(-cot(2\phi) - \sqrt{cot^2(2\phi) + 1}\right)}$$
$$= \frac{-1}{-cot(2\phi) - \sqrt{cot^2(2\phi) + 1}} \tag{25}$$

Taking the negative square root gives:

$$tan\,\phi = -cot(2\phi) - \sqrt{cot^2(2\phi) + 1} \tag{26}$$

$$= \frac{\left(-cot(2\phi) - \sqrt{cot^2(2\phi) + 1}\right)\left(-cot(2\phi) + \sqrt{cot^2(2\phi) + 1}\right)}{\left(-cot(2\phi) + \sqrt{cot^2(2\phi) + 1}\right)}$$
$$= \frac{-1}{-cot(2\phi) + \sqrt{cot^2(2\phi) + 1}} \tag{27}$$

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

$$tan\,\phi = \frac{-1}{\left(-cot(2\phi) + \sqrt{cot^2(2\phi) + 1}\right)} = \frac{sgn\left(cot(2\phi)\right)}{\left(|cot(2\phi)| + \sqrt{cot^2(2\phi) + 1}\right)} \tag{28}$$

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

$$tan\,\phi = \frac{1}{\left(cot(2\phi) + \sqrt{cot^2(2\phi) + 1}\right)} = \frac{sgn\left(cot(2\phi)\right)}{\left(|cot(2\phi)| + \sqrt{cot^2(2\phi) + 1}\right)} \tag{29}$$

In both cases:

$$tan\,\phi = \frac{sgn\left(cot(2\phi)\right)}{\left(|cot(2\phi)| + \sqrt{cot^2(2\phi) + 1}\right)} \tag{30}$$

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

$$tan\,\phi = \frac{sgn\left(cot(2\phi)\right)}{2|cot(2\phi)|} = \frac{-1}{2cot(2\phi)} \tag{31}$$

A trigonometric identity used later is:

$$\frac{sin\phi}{1+cos\phi} = \frac{2sin\left(\frac{\phi}{2}\right)cos\left(\frac{\phi}{2}\right)}{2cos^2\left(\frac{\phi}{2}\right)} = tan\left(\frac{\phi}{2}\right) \quad \textbf{(32)}$$

2.4 The Jacobi Algorithm

To avoid roundoff error, the iterative updates below are used for equations (14) to (19).

Equation (19):

By definition, the rotation angle ϕ is selected so that equation (19) results in zero $a'_{p,q}$.

$$a'_{p,q} = 0 \Rightarrow a_{p,q}(cos^2\phi - sin^2\phi) + \left(a_{p,p} - a_{q,q}\right)sin\phi cos\phi = 0 \quad \textbf{(33)}$$

$$\Rightarrow a_{q,q}sin^2\phi = \frac{sin\phi\{a_{p,q}(cos^2\phi - sin^2\phi) + a_{p,p}sin\phi cos\phi\}}{cos\phi} \quad \textbf{(34)}$$

Equation (16):

Substituting equation (34) into equation (16) gives:

$$a'_{p,p} = a_{p,p}cos^2\phi + sin\phi\left\{\frac{a_{p,q}(cos^2\phi - sin^2\phi) + a_{p,p}sin\phi cos\phi}{cos\phi}\right\} - 2a_{p,q}sin\phi cos\phi \quad \textbf{(35)}$$

$$= a_{p,p} + \left(\frac{a_{p,q}sin\phi(cos^2\phi - sin^2\phi) - 2a_{p,q}sin\phi cos^2\phi}{cos\phi}\right) \quad \textbf{(36)}$$

$$\Rightarrow a'_{p,p} = a_{p,p} - a_{p,q}sin\phi\left(\frac{sin^2\phi + cos^2\phi}{cos\phi}\right) = a_{p,p} - a_{p,q}\,tan\,\phi \quad \textbf{(37)}$$

Equation (18):

Since the rotation angle ϕ is selected to zero $a'_{p,q}$, equation (18) can be written as:

$$a_{p,q}(cos^2\phi - sin^2\phi) + \left(a_{p,p} - a_{q,q}\right)sin\phi cos\phi = 0 \quad \textbf{(38)}$$

$$\Rightarrow a_{p,p} = a_{q,q} - \left\{\frac{(cos^2\phi - sin^2\phi)a_{p,q}}{sin\phi cos\phi}\right\} \quad \textbf{(39)}$$

Equation (17):

Substituting equation (39) into equation (17) gives:

$$a'_{q,q} = sin^2\phi\left\{a_{q,q} - \left\{\frac{(cos^2\phi - sin^2\phi)a_{p,q}}{sin\phi cos\phi}\right\}\right\} + a_{q,q}cos^2\phi + 2a_{p,q}sin\phi cos\phi \quad \textbf{(40)}$$

$$\Rightarrow a'_{q,q} = a_{q,q} + sin\phi\left(\frac{sin^2\phi + cos^2\phi}{cos\phi}\right)a_{p,q} = a_{q,q} + a_{p,q}\tan\phi \quad \textbf{(41)}$$

Equation (14):

With trivial manipulation, equation (14) can be written as:

$$a'_{r,p} = a_{r,p} - a_{r,p}(1 - cos\phi) - a_{r,q}sin\phi \quad \textbf{(42)}$$

$$= a_{r,p} - sin\phi\left\{a_{r,q} + \frac{(1 - cos\phi)(1 + cos\phi)a_{r,p}}{sin\phi(1 + cos\phi)}\right\} = a_{r,p} - sin\phi\left\{a_{r,q} + \frac{a_{r,p}sin^2\phi}{sin\phi(1 + cos\phi)}\right\} \quad \textbf{(43)}$$

Substituting equation (32) gives:

$$a'_{r,p} = a_{r,p} - sin\phi\left(a_{r,q} + a_{r,p}tan\left(\frac{\phi}{2}\right)\right) \quad \textbf{(44)}$$

Equation (15):

Similarly, with trivial manipulation, equation (15) can be written as:

$$a'_{r,q} = a_{r,q} - a_{r,q}(1 - cos\phi) + a_{r,p}sin\phi \quad \textbf{(45)}$$

$$= a_{r,q} + sin\phi\left\{a_{r,p} - \frac{(1 - cos\phi)(1 + cos\phi)a_{r,q}}{sin\phi(1 + cos\phi)}\right\} = a_{r,q} + sin\phi\left\{a_{r,p} - \frac{a_{r,q}sin^2\phi}{sin\phi(1 + cos\phi)}\right\} \quad \textbf{(46)}$$

Substituting equation (32) gives:

$$a'_{r,q} = a_{r,q} + sin\phi\left(a_{r,p} - a_{r,q}tan\left(\frac{\phi}{2}\right)\right) \quad \textbf{(47)}$$

3. Legal information

3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

3.2 Disclaimers

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein. NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/ or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/salestermsandconditions.

3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

NXP, the NXP logo, Freescale, and the Freescale logo are trademarks of NXP B.V. ARM and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere.

4. Contents

1. Introduction 3
1.1 Summary 3
1.2 Terminology 3
1.3 Software Functions 3
2. Eigenanalysis by Jacobi Algorithm 4
2.1 Introduction 4
2.2 Givens Rotation Matrix 5
2.3 Determining the Givens Rotation Angle 6
2.4 The Jacobi Algorithm 8
3. Legal information 10
3.1 Definitions 10
3.2 Disclaimers 10
3.3 Trademarks 10
4. Contents 11

Date of release: 21 June 2016

Document identifier: AN5020