NAG Toolbox for MATLAB

Chapter Introduction

F04 – Simultaneous Linear Equations

Contents

1   Scope of the Chapter

This chapter is concerned with the solution of the matrix equation $AX = B$, where B may be a single vector or a matrix of multiple right-hand sides. The matrix A may be real, complex, symmetric, Hermitian, positive-definite, positive-definite Toeplitz or banded. It may also be rectangular, in which case a least-squares solution is obtained.

For a general introduction to sparse systems of equations, see the F11 Chapter Introduction, which currently provides functions for large sparse systems. Some functions for sparse problems are also included in this chapter; they are described in Section 3.4.

2   Background to the Problems

A set of linear equations may be written in the form

$$Ax = b$$

where the known matrix A, with real or complex coefficients, is of size m by n (m rows and n columns), the known right-hand vector b has m components (m rows and one column), and the required solution vector x has n components (n rows and one column). There may also be p vectors b_i, for $i = 1, 2, \ldots, p$, on the right-hand side and the equations may then be written as

$$AX = B,$$

the required matrix X having as its p columns the solutions of $Ax_i = b_i$, for $i = 1, 2, \ldots, p$. Some functions deal with the latter case, but for clarity only the case $p = 1$ is discussed here.

The most common problem, the determination of the unique solution of $Ax = b$, occurs when $m = n$ and A is not singular, that is rank$(A) = n$. This is discussed in Section 2.1 below. The next most common problem, discussed in Section 2.2 below, is the determination of the least-squares solution of $Ax \simeq b$ required when $m > n$ and rank$(A) = n$, i.e., the determination of the vector x which minimizes the norm of the residual vector $r = b - Ax$. All other cases are rank deficient, and they are treated in Section 2.3.

2.1   Unique Solution of $Ax = b$

Most functions in this chapter solve this particular problem. The computation starts with the triangular decomposition $A = PLU$, where L and U are respectively lower and upper triangular matrices and P is a permutation matrix, chosen so as to ensure that the decomposition is numerically stable. The solution is then obtained by solving in succession the simpler equations

$$\begin{aligned} Ly &= P^{\mathrm{T}}b \\ Ux &= y \end{aligned}$$

the first by forward-substitution and the second by back-substitution.

If A is real symmetric and positive-definite, $U = L^{\mathrm{T}}$, while if A is complex Hermitian and positive-definite, $U = L^{\mathrm{H}}$; in both these cases P is the identity matrix (i.e., no permutations are necessary). In all other cases either U or L has unit diagonal elements.

Due to rounding errors the computed 'solution' x_0, say, is only an approximation to the true solution x. This approximation will sometimes be satisfactory, agreeing with x to several figures, but if the problem is ill-conditioned then x and x_0 may have few or even no figures in common, and at this stage there is no means of estimating the 'accuracy' of x_0.

There are three possible approaches to estimating the accuracy of a computed solution.

One way to do so, and to 'correct' x_0 when this is meaningful (see next paragraph), involves computing the residual vector $r = b - Ax_0$ in extended precision arithmetic, and obtaining a correction vector d by solving $PLUd = r$. The new approximate solution $x_0 + d$ is usually more accurate and the correcting process is repeated until (a) further corrections are negligible or (b) they show no further decrease.

It must be emphasised that the 'true' solution x may not be meaningful, that is correct to all figures quoted, if the elements of A and b are known with certainty only to say p figures, where p is smaller than the word-length of the computer. The first correction vector d will then give some useful information about

the number of figures in the 'solution' which probably remain unchanged with respect to maximum possible uncertainties in the coefficients.

An alternative approach to assessing the accuracy of the solution is to compute or estimate the **condition number** of A, defined as

$$\kappa(A) = \|A\| . \|A^{-1}\|.$$

Roughly speaking, errors or uncertainties in A or b may be amplified in the solution by a factor $\kappa(A)$. Thus, for example, if the data in A and b are only accurate to 5 digits and $\kappa(A) \approx 10^3$, then the solution cannot be guaranteed to have more than 2 correct digits. If $\kappa(A) \geq 10^5$, the solution may have no meaningful digits.

To be more precise, suppose that

$$Ax = b \qquad \text{and} \qquad (A + \delta A)(x + \delta x) = b + \delta b.$$

Here δA and δb represent perturbations to the matrices A and b which cause a perturbation δx in the solution. We can define measures of the relative sizes of the perturbations in A, b and x as

$$\rho_A = \frac{\|\delta A\|}{\|A\|}, \qquad \rho_b = \frac{\|\delta b\|}{\|b\|} \qquad \text{and} \qquad \rho_x = \frac{\|\delta x\|}{\|x\|} \qquad \text{respectively.}$$

Then

$$\rho_x \leq \frac{\kappa(A)}{1 - \kappa(A)\rho_A}(\rho_A + \rho_b)$$

provided that $\kappa(A)\rho_A < 1$. Often $\kappa(A)\rho_A \ll 1$ and then the bound effectively simplifies to

$$\rho_x \leq \kappa(A)(\rho_A + \rho_b).$$

Hence, if we know $\kappa(A)$, ρ_A and ρ_b, we can compute a bound on the relative errors in the solution. Note that ρ_A, ρ_b and ρ_x are defined in terms of the norms of A, b and x. If A, b or x contains elements of widely differing magnitude, then ρ_A, ρ_b and ρ_x will be dominated by the errors in the larger elements, and ρ_x will give no information about the relative accuracy of smaller elements of x.

A third way to obtain useful information about the accuracy of a solution is to solve two sets of equations, one with the given coefficients, which are assumed to be known with certainty to p figures, and one with the coefficients rounded to $(p - 1)$ figures, and to count the number of figures to which the two solutions agree. In ill-conditioned problems this can be surprisingly small and even zero.

2.2 The Least-squares Solution of $Ax \simeq b$, $m > n$, rank$(A) = n$

The least-squares solution is the vector $\hat{x}$ which minimizes the sum of the squares of the residuals,

$$S = (b - A\hat{x})^{\mathrm{T}}(b - A\hat{x}) = \|b - A\hat{x}\|_2^2.$$

The solution is obtained in two steps.

(i) Householder Transformations are used to reduce A to 'simpler form' via the equation $QA = R$, where R has the appearance

$$\left(\frac{\hat{R}}{0}\right)$$

with $\hat{R}$ a nonsingular upper triangular n by n matrix and 0 a zero matrix of shape $(m - n)$ by n. Similar operations convert b to $Qb = c$, where

$$c = \left(\frac{c_1}{c_2}\right)$$

with c_1 having n rows and c_2 having $(m - n)$ rows.

(ii) The required least-squares solution is obtained by back-substitution in the equation

$$\hat{R}\hat{x} = c_1.$$

Again due to rounding errors the computed $\hat{x}_0$ is only an approximation to the required $\hat{x}$, but as in Section 2.1, this can be improved by 'iterative refinement'. The first correction d is the solution of the least-squares problem

$$Ad = b - A\hat{x}_0 = r$$

and since the matrix A is unchanged, this computation takes less time than that of the original $\hat{x}_0$. The process can be repeated until further corrections are (a) negligible or (b) show no further decrease.

2.3 Rank-deficient Cases

If, in the least-squares problem just discussed, $\text{rank}(A) < n$, then a least-squares solution exists but it is not unique. In this situation it is usual to ask for the least-squares solution 'of minimal length', i.e., the vector x which minimizes $\|x\|_2$, among all those x for which $\|b - Ax\|_2$ is a minimum.

This can be computed from the Singular Value Decomposition (SVD) of A, in which A is factorized as

$$A = QDP^{\mathrm{T}}$$

where Q is an m by n matrix with orthonormal columns, P is an n by n orthogonal matrix and D is an n by n diagonal matrix. The diagonal elements of D are called the 'singular values' of A; they are nonnegative and can be arranged in decreasing order of magnitude:

$$d_1 \geq d_2 \geq \cdots \geq d_n \geq 0.$$

The columns of Q and P are called respectively the left and right singular vectors of A. If the singular values $d_{r+1}, \ldots, d_n$ are zero or negligible, but d_r is not negligible, then the rank of A is taken to be r (see also Section 2.4) and the minimal length least-squares solution of $Ax \simeq b$ is given by

$$\hat{x} = D^{\dagger}Q^{\mathrm{T}}b$$

where $D^{\dagger}$ is the diagonal matrix with diagonal elements $d_1^{-1}, d_2^{-1}, \ldots, d_r^{-1}, 0, \ldots, 0$.

The SVD may also be used to find solutions to the homogeneous system of equations $Ax = 0$, where A is m by n. Such solutions exist if and only if $\text{rank}(A) < n$, and are given by

$$x = \sum_{i=r+1}^{n} \alpha_i p_i$$

where the α_i are arbitrary numbers and the p_i are the columns of P which correspond to negligible elements of D.

The general solution to the rank-deficient least-squares problem is given by $\hat{x} + x$, where $\hat{x}$ is the minimal length least-squares solution and x is any solution of the homogeneous system of equations $Ax = 0$.

2.4 The Rank of a Matrix

In theory the rank is r if $n - r$ elements of the diagonal matrix D of the singular value decomposition are exactly zero. In practice, due to rounding and/or experimental errors, some of these elements have very small values which usually can and should be treated as zero.

For example, the following 5 by 8 matrix has rank 3 in exact arithmetic:

$$\begin{pmatrix} 22 & 14 & -1 & -3 & 9 & 9 & 2 & 4 \\ 10 & 7 & 13 & -2 & 8 & 1 & -6 & 5 \\ 2 & 10 & -1 & 13 & 1 & -7 & 6 & 0 \\ 3 & 0 & -11 & -2 & -2 & 5 & 5 & -2 \\ 7 & 8 & 3 & 4 & 4 & -1 & 1 & 2 \end{pmatrix}.$$

On a computer with 7 decimal digits of precision the computed singular values were

$$3.5 \times 10^1, \quad 2.0 \times 10^1, \quad 2.0 \times 10^1, \quad 1.3 \times 10^{-6}, \quad 5.5 \times 10^{-7}$$

and the rank would be correctly taken to be 3.

It is not, however, always certain that small computed singular values are really zero. With the 7 by 7 Hilbert matrix, for example, where $a_{ij} = 1/(i+j-1)$, the singular values are

$$1.7, \quad 2.7 \times 10^{-1}, \quad 2.1 \times 10^{-2}, \quad 1.0 \times 10^{-3}, \quad 2.9 \times 10^{-5}, \quad 4.9 \times 10^{-7}, \quad 3.5 \times 10^{-9}.$$

Here there is no clear cut-off between small (i.e., negligible) singular values and larger ones. In fact, in exact arithmetic, the matrix is known to have full rank and none of its singular values is zero. On a computer with 7 decimal digits of precision, the matrix is effectively singular, but should its rank be taken to be 6, or 5, or 4?

It is therefore impossible to give an infallible rule, but generally the rank can be taken to be the number of singular values which are neither zero nor very small compared with other singular values. For example, if there is a sharp decrease in singular values from numbers of order unity to numbers of order 10^{-7}, then the latter will almost certainly be zero in a machine in which 7 significant decimal figures is the maximum accuracy. Similarly for a least-squares problem in which the data is known to about four significant figures and the largest singular value is of order unity then a singular value of order 10^{-4} or less should almost certainly be regarded as zero.

It should be emphasised that rank determination and least-squares solutions can be sensitive to the scaling of the matrix. If at all possible the units of measurement should be chosen so that the elements of the matrix have data errors of approximately equal magnitude.

2.5 Generalized Linear Least-squares Problems

The simple type of linear least-squares problem described in Section 2.2 can be generalized in various ways.

1. Linear least-squares problems with **equality constraints**:

$$\text{find } x \text{ to minimize } S = \|c - Ax\|_2^2 \qquad \text{subject to} \qquad Bx = d,$$

   where A is m by n and B is p by n, with $p \leq n \leq m + p$. The equations $Bx = d$ may be regarded as a set of equality constraints on the problem of minimizing S. Alternatively the problem may be regarded as solving an overdetermined system of equations

$$\begin{pmatrix} A \\ B \end{pmatrix} x = \begin{pmatrix} c \\ d \end{pmatrix},$$

   where some of the equations (those involving B) are to be solved exactly, and the others (those involving A) are to be solved in a least-squares sense. The problem has a unique solution on the assumptions that B has full row rank p and the matrix $\begin{pmatrix} A \\ B \end{pmatrix}$ has full column rank n. (For linear least-squares problems with **inequality constraints**, refer to Chapter E04.)

2. **General Gauss–Markov linear model problems**:

$$\text{minimize } \|y\|_2 \qquad \text{subject to} \qquad d = Ax + By,$$

   where A is m by n and B is m by p, with $n \leq m \leq n + p$. When $B = I$, the problem reduces to an ordinary linear least-squares problem. When B is square and nonsingular, it is equivalent to a **weighted linear least-squares problem**:

$$\text{find } x \text{ to minimize } \left\| B^{-1}(d - Ax) \right\|_2.$$

   The problem has a unique solution on the assumptions that A has full column rank n, and the matrix (A, B) has full row rank m.

2.6 Calculating the Inverse of a Matrix

The functions in this chapter can also be used to calculate the inverse of a square matrix A by solving the equation

$$AX = I$$

where I is the identity matrix. However, solving the equations $AX = B$ by calculation of the inverse of the coefficient matrix A, i.e., by $X = A^{-1}B$, is **definitely not recommended**.

Similar remarks apply to the calculation of the pseudo-inverse of a singular or rectangular matrix.

3   Recommendations on Choice and Use of Available Functions

3.1   Black Box and General Purpose Functions

Most of the functions in this chapter are categorised as Black Box functions or general purpose functions.

Black Box functions solve the equations $Ax_i = b_i$, for $i = 1, 2, \ldots, p$, in a single call with the matrix A and the right-hand sides, b_i, being supplied as data. These are the simplest functions to use and are suitable when all the right-hand sides are known in advance and do not occupy too much storage.

General purpose functions, in general, require a previous call to a function in Chapters F01, F03 or F07 to factorize the matrix A. This factorization can then be used repeatedly to solve the equations for one or more right-hand sides which may be generated in the course of the computation. The Black Box functions simply call a factorization function and then a general purpose function to solve the equations.

The function f04qa which uses an iterative method for sparse systems of equations does not fit easily into this categorisation, but is classified as a general purpose function in the decision trees and indexes.

3.2   Systems of Linear Equations

Most of the functions in this chapter solve linear equations $Ax = b$ when A is n by n and a unique solution is expected (case 2.1). If this turns out to be untrue the functions go to a failure exit. The matrix A may be 'general' real or complex, or may have special structure or properties, e.g., it may be banded, tridiagonal, almost block-diagonal, sparse, symmetric, Hermitian, positive-definite (or various combinations of these).

It must be emphasised that it is a waste of computer time and space to use an inappropriate function, for example one for the complex case when the equations are real. It is also unsatisfactory to use the special functions for a positive-definite matrix if this property is not known in advance.

Functions are given for calculating the **approximate solution**, that is solving the linear equations just once, and also for obtaining the **accurate solution** by successive iterative corrections of this first approximation using additional precision arithmetic, as described in Section 2.1. The latter, of course, are more costly in terms of time and storage, since each correction involves the solution of n sets of linear equations and since the original A and its LU decomposition must be stored together with the first and successively corrected approximations to the solution. In practice the storage requirements for the 'corrected' functions are about double those of the 'approximate' functions, though the extra computer time may not be prohibitive since the same matrix and the same LU decomposition is used in every linear equation solution.

A number of the Black Box functions in this chapter return estimates of the condition number and the forward error, along with the solution of the equations. But for those functions that do no return a condition estimate two functions are provided – f04yc for real matrices, f04zc for complex matrices – which can return a cheap but reliable estimate of $\left\| A^{-1} \right\|$, and hence an estimate of the **condition number** $\kappa(A)$ (see Section 2.1). These functions can also be used in conjunction with most of the linear equation solving functions in Chapter F11: further advice is given in the function documents.

Other functions for solving linear equation systems, computing inverse matrices, and estimating condition numbers can be found in Chapter F07, which contains LAPACK software.

3.3   Linear Least-squares Problems

The majority of the functions for solving linear least-squares problems are to be found in Chapter F08.

For case 2.2, when $m \geq n$ and a unique least-squares solution is expected, there are two functions for a general real A, one of which (f04jg) computes a first approximation and the other (f04am) computes iterative corrections. If it transpires that $\mathrm{rank}(A) < n$, so that the least-squares solution is not unique, then f04am takes a failure exit, but f04jg proceeds to compute the **minimal length** solution by using the SVD (see below).

If A is expected to be of less than full rank then one of the functions for calculating the minimal length solution may be used.

For $m \gg n$ the use of the SVD is not significantly more expensive than the use of functions based upon the QR factorization.

Problems with linear **equality constraints** can be solved by f08zb (for real data) or by f08zp (for complex data), provided that the problems are of full rank. Problems with linear **inequality constraints** can be solved by e04nc in Chapter E04.

General Gauss–Markov linear model problems, as formulated in Section 2.5, can be solved by f08za (for real data) or by f08zn (for complex data).

3.4 Sparse Matrix Functions

Functions specifically for sparse matrices are appropriate only when the number of nonzero elements is very small, less than, say, 10% of the n^2 elements of A, and the matrix does not have a relatively small band width.

Chapter F11 contains functions for both the direct and iterative solution of real sparse linear systems. There are two functions in Chapter f04 for solving sparse linear equations (f04ax and f04qa). f04ax utilizes a factorization of the matrix A obtained from f01br or f01bs, while f04qa uses an iterative technique and requires a user-supplied function to compute matrix-vector products Ac and $A^{\mathrm{T}}c$ for any given vector c.

f04qa solves sparse least-squares problems by an iterative technique, and also allows the solution of damped (regularised) least-squares problems (see the function document for details).

4 Decision Trees

The name of the function (if any) that should be used to factorize the matrix A is given in brackets after the name of the function for solving the equations.

Tree 1: Black Box functions for unique solution of $Ax = b$ (Real matrix)

| Is A symmetric? | →yes | Is A positive-definite? | →yes | Is A a band matrix? | →yes | Is A tridiagonal? | →yes | f04bg (see Note 1) or f07ja and f07jb (see Note 2) |

no → f04bf (see Note 1) or f07ha or f07hb (see Note 2)

no → Do you require an accurate solution using iterative refinement? →yes→ f04ab or f04as (see Note 3)

no → Is one triangle of A stored as a linear array? →yes→ f04be (see Note 1) or f07ga or f07gb (see Note 2)

no → f04bd (see Note 1) or f07fa or f07fb (see Note 2)

no → Is one triangle of A stored as a linear array? →yes→ f04bj (see Note 1) or f07pa or f07pb (see Note 2)

no → f04bh (see Note 1) or f07ma or f07mb (see Note 2)

no → Is A a band matrix? →yes→ Is A tridiagonal? →yes→ f04bc (see Note 1) or f07ca or f07cb (see Note 2)

no → f04bb (see Note 1) or f07ba or f07bb (see Note 2)

no → Do you require an accurate solution using iterative refinement? →yes→ f04ae or f04at (see Note 3)

no → f04ba (see Note 1) or f07aa or f07ab (see Note 2)

Tree 2: Black Box functions for unique solution of $Ax = b$ (Complex matrix)

| Is A Hermitian? | →yes | Is A positive-definite? | →yes | Is A a band matrix? | →yes | Is A bidiagonal? | →yes | f04cg (see Note 1) or f07jn or f07jp (see Note 2) |

no → f04cf (see Note 1) or f07hn or f07hp (see Note 2)

no → Is one triangle of A stored as a linear array? →yes→ f04ce (see Note 1) or f07gn or f07gp (see Note 2)

no →

```
                                          ┌─────────────────────┐
                                          │ f04cd (see Note 1) or│
                                          │ f07fn or f07fp (see  │
                                          │ Note 2)              │
                                          └─────────────────────┘
                              no
                    ┌──────────────────┐    ┌─────────────────────┐
                    │ Is one triangle  │    │ f04cj (see Note 1) or│
                    │ of A stored as a │────│ f07pn or f07pp (see  │
                    │ linear array?    │ yes│ Note 2)              │
                    └──────────────────┘    └─────────────────────┘
                              no
                    ┌──────────────────┐
                    │ f04ch (see Note 1)│
                    │ or f07mn or f07mp │
                    │ (see Note 2)      │
                    └──────────────────┘
      no
┌──────────────┐    ┌──────────────────┐    ┌─────────────────────┐
│ Is A         │    │ Is one triangle  │    │ f04dj (see Note 1) or│
│ symmetric?   │────│ of A stored as a │────│ f07qn or f07qp (see  │
└──────────────┘ yes│ linear array?    │ yes│ Note 2)              │
                    └──────────────────┘    └─────────────────────┘
                              no
                    ┌──────────────────┐
                    │ f04dh (see Note 1)│
                    │ or f07nn or f07np │
                    │ (see Note 2)      │
                    └──────────────────┘
      no
┌──────────────┐    ┌──────────────────┐    ┌─────────────────────┐
│ Is A a band  │    │ Is A             │    │ f04cc (see Note 1) or│
│ matrix?      │────│ tridiagonal?     │────│ f07cn or f07cp (see  │
└──────────────┘ yes└──────────────────┘ yes│ Note 2)              │
                              no             └─────────────────────┘
                    ┌──────────────────┐
                    │ f04cb (see Note 1)│
                    │ or f07bn or f07bp │
                    │ (see Note 2)      │
                    └──────────────────┘
      no
┌──────────────┐
│ f04ca (see   │
│ Note 1) or   │
│ f07an or f07ap│
│ (see Note 2) │
└──────────────┘
```

Tree 3: General purpose functions for unique solution of $Ax = b$ (Real matrix)

Is A a sparse matrix and not banded?	Chapter F11 or f04ax (f01br or f01bs) or f04qa

yes →

no ↓

Is A symmetric? —yes→ Is A positive-definite? —yes→ Is A band matrix? —yes→ Is A tridiagonal? —yes→ f07je (f07jd)

no ↓ (from tridiagonal)

Is A variable band width? —yes→ f04mc (f01mc)

no ↓

f07he (f07hd)

no ↓ (from band matrix)

Is A a Toeplitz matrix? —yes→ Are the equations the Yule–Walker equations? —yes→ f04me

no ↓

f04mf

no ↓ (from Toeplitz)

Is one triangle of A stored as a linear array? —yes→ f07ge (f07gd)

no ↓

f07fe (f07fd)

no ↓ (from positive-definite)

Is one triangle of A stored as a linear array? —yes→ f07pe (f07pd)

no ↓

f07me (f07md)

no ↓ (from symmetric)

Is A triangular? —yes→ Is A a band matrix? —yes→ f07ve

no ↓

Is A stored as a linear array? —yes→ f07ue

no ↓

f07te

no ↓ (from triangular)

Is A a band matrix? —yes→ Is A tridiagonal? —yes→ f04le (f01le) or f07ce (f07cd)

no ↓

Is A almost block diagonal? —yes→ f04lh (f01lh)

no ↓

f07be (f07bd)

no ↓ (from band matrix)

f07ae (f07ad)

Tree 4: General purpose functions for unique solution of $Ax = b$ (Complex matrix)

Is A a sparse matrix and not banded? — yes → Chapter F11

no

Is A Hermitian? — yes → Is A positive-definite? — yes → Is A a band matrix? — yes → Is A tridiagonal? — yes → f07js (f07jr)

no → f07hs (f07hr)

no

Is one triangle of A stored as a linear array? — yes → f07gs (f07gr)

no → f07fs (f07fr)

no

Is one triangle of A stored as a linear array? — yes → f07ps (f07pr)

no → f07ms (f07mr)

no

Is A symmetric? — yes → Is one triangle of A stored as a linear array? — yes → f07qs (f07qr)

no → f07ns (f07nr)

no

Is A triangular? — yes → Is A a band matrix? — yes → f07vs

no

Is A stored as a linear array? — yes → f07us

no → f07ts

no

Is A a band matrix? — yes → Is A tridiagonal? — yes → f07cs (f07cr)

no → f07bs (f07br)

no → f07as (f07ar)

Tree 5: General purpose functions for least-squares and homogeneous equations (without constraints)

Note: there are also functions in Chapter F08 for solving least-squares problems.

Is the problem $Ax = 0$? — yes → Is $m < n$? — yes → f04jd

no → f04ja

no

Is A sparse? — yes → f04qa

no

Is rank$(A) = n$? — yes → Are storage and time more important than accuracy? — yes → f04jg

no → f04am

```
                            │ no
                            │
┌──────────────────┬────────┐        ┌─────────────────────────────────────┐
│ Is m > n?        │        │──────  │         f04ja or f04jg              │
└──────────────────┴────────┘ yes    └─────────────────────────────────────┘
                            │ no
┌───────────────────────────┐
│          f04jd            │
└───────────────────────────┘
```

Note 1: also returns an estimate of the condition number and the forward error.

Note 2: also returns an estimate of the condition number, the forward error and the backward error. Requires additional workspace.

Note 3: for a single right-hand side only.

5   Index

6   References

Golub G H and Van Loan C F 1996 *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Lawson C L and Hanson R J 1974 *Solving Least-squares Problems* Prentice–Hall

Wilkinson J H and Reinsch C 1971 *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag