A brief overview of the condition number

Aaron C Acton

25 October 2008

Abstract

This article presents an overview of the condition number for a matrix and the potential effect of ill conditioning on the solution of a system of linear equations. The information is intended to be general, although specific information relevant to finite-element analysis is also included. Vector and matrix norms are introduced before defining the condition number, and the choice of matrix norm in the calculation of the condition number is discussed. A method of estimating the condition number is also provided, including a sample implementation in the ANSYS Parametric Design Language (APDL).

Keywords: Ill conditioning, Matrix condition number, Matrix norm, Vector norm
Tested in: ANSYS 11.0

1 Introduction

Many works of scientific and mathematical literature make passing references to the *condition* of a system of equations. In some cases, this term may not be clearly defined, and attempting to research the *condition number* may lead to apparently-inconsistent definitions, which may involve the product of two norms, the quotient of two eigenvalues, or the quotient of two singular values. This article attempts to provide an overview of the condition number and to explain the difference that may be observed between the definitions provided elsewhere.

2 Ill-Conditioned Systems

Consider a simple two-degree-of-freedom (DOF) system, represented by two straight lines that are nearly parallel, as shown in Figure 1 (example borrowed from Meyer [1]). If one of the lines changes slope only slightly, the solution of the system, represented by the intersection of the lines, is drastically altered. Such a system is said to be *ill conditioned* (formal definition presented in Section 4). This two-DOF system can be represented using matrix notation as

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix}$$

or simply

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad ,$$

where a bold-face font indicates a matrix or vector. In practice, the coefficients in $\mathbf{A}$ are often obtained empirically—by measurement of material properties, for example—and may not be known with perfect accuracy. Since small perturbations in the coefficients of such a system can potentially have a large effect on the solution, as demonstrated by Figure 1, it is of interest to detect when a system of equations will exhibit this type of behaviour. One method of doing so is by quantifying the *condition* of the coefficient matrix.

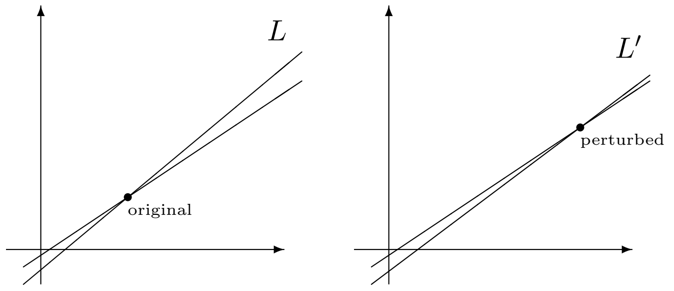

Figure 1: Graphical representation of an ill-conditioned system with two degrees of freedom.

This sensitivity of results to coefficient perturbations is intrinsic to the system itself and is not a result of the solution procedure [1]. Furthermore, even if perfectly-accurate coefficients could be obtained, perturbations can arise from numerical roundoff (resulting from finite-precision arithmetic) and truncation (resulting from finite-precision data representation) [2]. It is not entirely caused by accumulated rounding errors in the solution algorithms, however, since there are relatively few arithmetical operations involved in calculating the solution (at least, when using direct methods), limiting the opportunity for rounding errors to grow [3]. Consequently, no numerical technique can be used with confidence when the system is ill conditioned [1].

Before defining and discussing matrix condition, it is first necessary to briefly review vector and matrix norms.

3 Vector and Matrix Norms

It is elementary to identify a single (*i.e.*, scalar) number as being *larger* or *smaller* than another; however, it may also be necessary to quantify the *size* of vectors or matrices. *Norms* attempt to accomplish this by calculating a single number that represents the magnitude of a matrix or vector, based on some function of all the elements [2].

The p-norms for a vector $\mathbf{x}$ are defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{1}$$

where $p \geq 1$. Some frequently-used, particular cases of p-norms are the 1-norm (*one* norm),

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \quad , \tag{2}$$

the 2-norm (*two* norm, also called the *Euclidean* norm),

$$\|\mathbf{x}\|_2 = \left(\mathbf{x}^T\mathbf{x}\right)^{\frac{1}{2}} = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}} \quad , \tag{3}$$

and the ∞-norm (*infinity* norm),

$$\|\mathbf{x}\|_\infty = \max_i |x_i| \quad . \tag{4}$$

The p-norms for a matrix $\mathbf{A}$, subordinate to the p-norms for a vector, are defined as

$$\|\mathbf{A}\|_p = \sup_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}_p\|}{\|\mathbf{x}_p\|} \quad , \tag{5}$$

where "sup" is the *supremum*, or *least upper bound*. Again, the frequently-used, particular cases are the 1-norm,

$$\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^{n} |a_{i,j}| \quad , \tag{6}$$

the 2-norm (also called the *spectral* norm),

$$\|\mathbf{A}\|_2 = \sqrt{\tilde{\lambda}_{\max}} \quad , \tag{7}$$

where $\tilde{\lambda}_{\max}$ is the maximum eigenvalue of $\mathbf{A}^T\mathbf{A}$, and the ∞-norm,

$$\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{i,j}| \quad , \tag{8}$$

A further discussion on vector and matrix norms can be found in many of the references used for this article.

Although this is a very brief view of norms, it is presented here since the calculation of the condition number depends on norms.

4 Matrix Condition Number

Since small perturbations in the coefficient matrix of an ill-conditioned system can potentially have a large effect on the solution, it is of interest to detect when a particular system may exhibit this behaviour. A quantity known as the *condition number* can be used to indicate the amount of ill conditioning in a system.

The condition number, typically denoted by κ, of a nonsingular (*i.e.*, invertible) matrix $\mathbf{A}$, is defined as

$$\kappa\left(\mathbf{A}\right) = \left\|\mathbf{A}\right\|\left\|\mathbf{A}^{-1}\right\| \quad , \tag{9}$$

where $\mathbf{A}$ is a square matrix, $\mathbf{A}^{-1}$ is the inverse of $\mathbf{A}$, and $\|\cdot\|$ is a matrix norm. A subscript may be introduced into the notation to indicate the choice of norm, such as in

$$\kappa_p\left(\mathbf{A}\right) = \left\|\mathbf{A}\right\|_p\left\|\mathbf{A}^{-1}\right\|_p \quad ,$$

where p may be 1, 2, or ∞, for example, to denote the condition number of $\mathbf{A}$ in the *one*, *two*, or *infinity* norm, respectively. (It should be noted that the condition number calculation is not limited to p-norms; others, such as the *Frobenuis* norm, may be used.) The role of the condition number is described by Kreyszig as follows:

> A linear system of equations $\mathbf{Ax} = \mathbf{b}$ whose condition number is small is well-conditioned. A large condition number indicates ill-conditioning [4].

Kreyszig states that if the system is well conditioned, small inaccuracies in $\mathbf{A}$ can have a only a small effect on the solution of $\mathbf{x}$, but also notes that there is no sharp dividing line between *well-conditioned* and *ill-conditioned*. Stated simply, Bathe explains that "a large condition number means that solution errors are more likely" [2]. Attempts to differentiate between *well* and *ill* will be discussed later.

Zarowski adds to this the case where $\mathbf{b}$ may have inaccuracies, stating that "a matrix $\mathbf{A}$ is *ill-conditioned* if the solution $\mathbf{x}$ (in $\mathbf{Ax} = [\mathbf{b}]$) is very sensitive to perturbations on $\mathbf{A}$ and $[\mathbf{b}]$. Otherwise, the matrix is said to be *well-conditioned*" [3]. In other words, Hahn explains that errors in $\mathbf{b}$ "are likely to be magnified in $\mathbf{x}$ when $\mathbf{A}^{-1}$ has large entries. Ill conditioning should therefore be suspected whenever $\mathbf{A}^{-1}$ has entries that are much larger than 1" [5].

Kiusalaas discusses the situation in which the coefficient matrix is almost singular, and explains that if the condition number "is close to unity, the matrix is well-conditioned. The condition number increases with the degree of ill-conditioning, reaching infinity for a singular matrix" [6]. The following words of caution are also offered, which are worth quoting at length:

> Numerical solutions of ill-conditioned equations are not to be trusted. The reason is that the inevitable roundoff errors during the solution process are equivalent to introducing small changes into the coefficient matrix. This in turn introduces large errors into the solution, the magnitude of which depends on the severity of ill-conditioning. In suspect cases the determinant of the coefficient matrix should be computed so that the degree of ill-conditioning can be estimated [6].

Kiusalaas quantifies this by suggesting that if the determinant is small relative to the elements of $\mathbf{A}$,

$$|\mathbf{A}| \ll \|\mathbf{A}\| \quad ,$$

roundoff error is to be expected [6]. It should not be implied from this, however, that a small determinant is indicative of a large condition number. Even though $|\mathbf{A}| = 0$ indicates a singular matrix, $|\mathbf{A}| \approx 0$ does not necessarily indicate a nearly-singular matrix. Golub reports that there is little correlation between $|\mathbf{A}|$ and $\kappa(\mathbf{A})$ [7].

Meyer also mentions that the coefficients of the system of equations are often obtained empirically within certain tolerances. Consequently, small uncertainty in an ill-conditioned system can result in nonsensical results, even given infinitely-accurate arithmetic [1].

With several choices of norms in addition to the various definitions of ill conditioning, a short discussion on the choice of matrix norm will be presented here.

5 Choice of Matrix Norm

In Section 4, it was seen that there the definition of the condition number depends on the choice of norm, some of which were presented in Section 3. Zarowski provides some insight into this issue, which is also worth quoting at length:

> [T]he specific value of the condition number depends in part on the choice of norm. However, equivalence says that if a matrix is ill-conditioned with respect to one type of norm, then it must be ill-conditioned with respect to any other type of norm. This can simplify analysis in practice because it allows us to compute the condition number using whatever norms are the easiest to work with [3].

It will be seen in the following sections that some norms are easier to obtain than others, especially in the case where singular-value decomposition is used to solve a system of linear equations.

Since the 1-norm from Equation 6 and ∞-norm from Equation 8 are relatively straightforward to calculate, two seemingly-simple cases for the condition number are the respective

$$\kappa_1(\mathbf{A}) = \|\mathbf{A}\|_1 \|\mathbf{A}^{-1}\|_1$$

and

$$\kappa_\infty(\mathbf{A}) = \|\mathbf{A}\|_\infty \|\mathbf{A}^{-1}\|_\infty \quad .$$

While it is true that $\|\mathbf{A}\|_1$ and $\|\mathbf{A}\|_\infty$ can be obtained quickly, $\|\mathbf{A}^{-1}\|_1$ and $\|\mathbf{A}^{-1}\|_\infty$ require calculating $\mathbf{A}^{-1}$, which is expensive in general and may not be easy or possible to calculate, precisely because of ill conditioning [3]. One method of dealing with this issue is to estimate the values of $\|\mathbf{A}^{-1}\|_1$ and $\|\mathbf{A}^{-1}\|_\infty$, a topic to be discussed briefly in Section 8.

The condition number in the 2-norm,

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 \quad ,$$

can be given by

$$\kappa_2(\mathbf{A}) = \sqrt{\frac{\tilde{\lambda}_{\max}}{\tilde{\lambda}_{\min}}} = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \quad , \qquad (10)$$

where $\tilde{\lambda}_{\max}$ and $\tilde{\lambda}_{\min}$ are the largest and smallest eigenvalues of $\mathbf{A}^T\mathbf{A}$, while $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest singular values of $\mathbf{A}$ (discussed later). This says that "**A** is ill-conditioned if the ratio of the biggest to smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$ is large. In other words, a *large eigenvalue spread* is associated with matrix ill conditioning" [3]. Similarly, Bathe states that solution errors can arise not only from a small (near zero) eigenvalue of **A**, but also from a large ratio of the largest to smallest eigenvalues of **A** [2]. The calculation for the condition number given by Bathe is

$$\kappa(\mathbf{A}) = \frac{\lambda_{\max}}{\lambda_{\min}} \quad ,$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of **A** [2]. In fact, this is simply the condition number in the 2-norm for the special case when **A** is symmetric and positive definite [8].

The calculation of the condition number in the 2-norm is trivial during a solution using matrix decomposition. A Gaussian-elimination approach to solving $\mathbf{Ax} = \mathbf{b}$ is to decompose **A** into

$$\mathbf{A} = \mathbf{LU} \quad ,$$

where **L** and **U** are lower and upper triangular matrices, respectively. It may be necessary to introduce an additional matrix (including, but not limited to, the case where **A** is not positive definite), such that

$$\mathbf{A} = \mathbf{L}'\mathbf{D}\mathbf{U}' \quad ,$$

where $\mathbf{L}'$ and $\mathbf{U}'$ are now *unit* triangular matrices (*i.e.*, triangular matrices with all diagonal elements equal to one) and **D** is a diagonal matrix (*i.e.*, all off-diagonal elements equal to zero). This technique is known as *singular value decomposition* (SVD), where the "singular values" of **A** are given by the σ_i values in

$$\mathbf{D} = \operatorname{diag}(\sigma_1, \sigma_1, \ldots, \sigma_n) \quad . \tag{11}$$

It can be seen from Equation 11 that the condition number in the 2-norm can be obtained directly from the entries of **D** using Equation 10 (repeated here for convenience).

$$\kappa_2(\mathbf{A}) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

Thus, a large singular-value spread is associated with an ill-conditioned matrix [1, 3, 8].

As a side note, if **A** is symmetric and positive definite, such that

$$\mathbf{A} = \mathbf{LL}^T \quad ,$$

the Cholesky decomposition can be used to improve on the efficiency of the decomposition algorithm by taking advantage of the symmetry in **A**. SVD may again be used to obtain

$$\mathbf{A} = \mathbf{L}'\mathbf{D}\mathbf{L}'^T \quad ,$$

where again the entries of **D** are the singular values of **A** and can be used to obtain the condition number in the 2-norm.

6 Effect on Accuracy

Zarowski provides the following "rule of thumb" for interpreting condition numbers obtained with p-norms:

> If $\kappa_p(\mathbf{A}) \approx d \times 10^k$, where d is a decimal digit from one to nine, we can expect to lose (at worst) about k digits of accuracy [3].

Bathe also makes reference to the effect on accuracy, showing that an estimate for the number of accurate digits obtained in the solution is

$$s \geq t - \log_{10}[\kappa(\mathbf{A})] \quad ,$$

where s is the number of digits of precision in the solution, and t is the number of digits of precision in the computer [2]. Meyer provides a similar rule:

> If Gaussian elimination with partial pivoting is used to solve a well-scaled nonsingular system $\mathbf{Ax} = \mathbf{b}$ using t-digit floating-point arithmetic, then, assuming no other source of error exists, it can be argued that when κ is of order 10^p, the computed solution is expected to be accurate to at least $t - p$ significant digits, more or less [1].

Consequently, in a computer using double-precision arithmetic, which has 15 digits of accuracy, a value of $k \geq 16$ could indicate that the computed solution is not close to the exact solution.

Finite element codes generally use either single- or double-precision data types to store numerical values, which are stored in 32- or 64-bit floating-point registers, respectively. It is important to note that 32-bit processors generally contain both 32- and 64-bit registers for floating-point numbers, allowing for double-precision arithmetic on 32-bit systems [9].

Irvine explains that floating-point numbers with single-precision are stored in IEEE REAL4 registers and those with double-precision in IEEE REAL8 registers [9], where *IEEE* refers to the standard real-number formats published by the IEEE Computer Society in the "IEEE standard for Binary Floating-Point Arithmetic

(IEEE 754)." The REAL4 registers have $23+1$ significant bits (base-2), giving approximately

$$\lfloor \log_{10}\left(2^{23+1}\right)\rfloor = 7$$

significant digits (base-10). Similarly, REAL8 registers, which have $52+1$ significant bits, have

$$\lfloor \log_{10}\left(2^{52+1}\right)\rfloor = 15$$

significant digits.

7 Estimating the Solution Error

Since small perturbations in the coefficients may cause large errors in the solution, one may attempt to determine the extent of ill conditioning by perturbing the coefficients in $\mathbf{A}$; however, this will only determine the extent of ill conditioning, *not* the amount of error. "If a radical change in the solution is observed for a small perturbation to some set of coefficients, then you have uncovered an ill-conditioned situation. If a given perturbation does not produce a large change in the solution, then nothing can be concluded" [1]. The potential for error in the results deserves a closer examination.

The difference between the computed solution $\hat{\mathbf{x}}$ and the exact solution $\mathbf{x}$ of the system $\mathbf{Ax} = \mathbf{b}$ is simply

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} \quad .$$

If the computed solution $\hat{\mathbf{x}}$ is different than the exact solution $\mathbf{x}$, then $\mathbf{A}\hat{\mathbf{x}} \neq \mathbf{b}$, so the *residual* is then defined as

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}} \quad .$$

Zarowski cautions that "if $\kappa_p(\mathbf{A})$ is large, it is a warning (not a certainty) that small perturbations in $\mathbf{A}$ and [$\mathbf{b}$] may cause $\hat{\mathbf{x}}$ to differ greatly from $\mathbf{x}$. Equivalently, if $\kappa_p(\mathbf{A})$ is large, then a small $\mathbf{r}$ does not imply that $\hat{\mathbf{x}}$ is close to $\mathbf{x}$" [1]. Meyer adds that "[r]esiduals are reliable indicators of accuracy only when $\mathbf{A}$ is well conditioned—if $\mathbf{A}$ is ill conditioned, residuals are nearly meaningless" [1]. This indicates that attempts to check the solution for errors by calculating the residuals does not guarantee that $\hat{\mathbf{x}}$ is close to $\mathbf{x}$ when the system is ill conditioned, even if the residuals are small.

Zarowski defines a single number for the absolute error as

$$\epsilon_a = \|\mathbf{x} - \hat{\mathbf{x}}\|$$

and the relative error as

$$\epsilon_r = \frac{\epsilon_a}{\|\mathbf{x}\|} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \quad .$$

Then, it is shown that an upper bound for the relative error can be obtained by

$$\epsilon_r \leq \kappa \frac{\|\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}\|}{\|\mathbf{b}\|} \quad .$$

Meyer provides an approximate upper bound for the error. In the linear system $\mathbf{Ax} = \mathbf{b}$, if $\mathbf{A}$ is perturbed by some small matrix $\mathbf{B}$, then the relative error between the exact solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ and the calculated solution $\hat{\mathbf{x}} = (\mathbf{A} + \mathbf{B})^{-1}\mathbf{b}$ is given by

$$\epsilon_r \lesssim \kappa \frac{\|\mathbf{B}\|}{\|\mathbf{A}\|} \quad .$$

The technique of perturbing the coefficient matrix to observe the effect on the solution may be useful in determining the extent of ill conditioning in a system; however, this can be a time consuming and computationally expensive [1].

8 Estimating the Condition Number

While calculating the condition number appears to be relatively straightforward, $\mathbf{A}^{-1}$ is generally not known in practice. In addition, when dealing with a large set of equations, as often encountered in finite-element analysis, calculating $\mathbf{A}^{-1}$ can be very computationally expensive. Even if one could afford the time to find $\mathbf{A}^{-1}$, it would be unknown if the resulting matrix is in fact accurate, as conditioning of the matrix is unknown, making for what Zarowski described as a "chicken and egg" problem. It is for these reasons that other methods of estimating the condition number have been developed.

In 1984, Hager published a technique for estimating the condition number of a matrix and compared the results to those from an earlier scheme [10]. It was noted that calculating the condition number of a matrix $\mathbf{A}$ is trivial when the inverse $\mathbf{A}^{-1}$ is known, but it is rarely needed in scientific computations and it is computationally expensive to find. For these reasons, Hager focused on the problem of determining $\|\mathbf{A}^{-1}\|_1$ and $\|\mathbf{A}^{-1}\|_\infty$ from the factors of $\mathbf{A}$, not the inverse. Hager's algorithm is as follows:

Given $\mathbf{A} \in \mathbb{R}^{n \times n}$, this algorithm computes an estimate $\gamma \leq \left\|\mathbf{A}^{-1}\right\|_1$.

```
choose $x$ with $\|x\|_1 = 1$
repeat
    solve $Ay = x$
    $\xi := \mathrm{sign}(y)$
    solve $A^T z = \xi$
    if $\|z\|_\infty \leq z^T x$
        $\gamma = \|y\|_1$
        quit
    $x := e_j$, where $|z_j| = \|z\|_\infty$
end
```

Higham published an analysis of this algorithm in 1987 and found that it performed well in practice, obtaining the estimated maximum in two to three iterations and within an order of magnitude of the true condition number (which is relatively good) [11]. Higham later modified the algorithm to improve the reliability and efficiency [12], and this improved method was later incorporated in MATLAB as the function `condest` [13].

It seems as though no discussion on matrix condition would be complete without at least a brief note about the Hilbert matrix as it is a classical example of an ill-conditioned matrix, and the larger the size of the matrix, the more ill conditioned it becomes. A *Hilbert matrix* of size n, often denoted by $\mathbf{H}_n$, is composed of elements calculated as

$$h_{ij} = \frac{1}{i+j-1} \quad ,$$

for $i, j = 1, 2, \cdots, n$; for example, with $n = 4$,

$$\mathbf{H}_4 = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix} \quad .$$

To demonstrate how the aforementioned Hager algorithm could be implemented in APDL, an input file which calculates the condition number of a Hilbert matrix can be found in Appendix A.1.

9 Conclusions

The condition number is simply a value used to indicate when the solution of a system may be sensitive to coefficient perturbation, which may (or may not) cause large inaccuracies in the results. While there are several methods used to calculate the condition number, all of the methods result in some measure of the condition of a system, and the choice of using one method over another may be made depending on which is easier.

References

[1] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra.* Society for Industrial and Applied Mathematics, 2001.

[2] K.-J. Bathe, *Finite Element Procedures.* Prentice Hall, 2006.

[3] C. J. Zarowski, *An Introduction to Numerical Analysis for Electrical and Computer Engineers.* John Wiley & Sons, 2004.

[4] E. Kreyszig, *Advanced Engineering Mathematics.* John Wiley & Sons, 8th ed., 1999.

[5] B. D. Hahn and K. M. Malan, *Essential Java for Scientists and Engineers.* Butterworth-Heinemann, 2002.

[6] J. Kiusalaas, *Numerical Methods in Engineering with MATLAB®.* Cambridge University Press, 2005.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Johns Hopkins University Press, 3rd ed., 1996.

[8] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics.* Springer, 2000.

[9] K. R. Irvine, *Assembly Language for Intel®-Based Computers.* Pearson Education, 2003.

[10] W. W. Hager, "Condition estimates," *SIAM Journal on Scientific and Statistical Computing*, vol. 5, no. 2, pp. 311–316, 1984.

[11] N. J. Higham, "A survey of condition number estimation for triangular matrices," *SIAM Journal on Scientific and Statistical Computing*, vol. 29, no. 4, pp. 575–596, 1987.

[12] N. J. Higham, "FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation," *ACM Transactions on Mathematical Software*, vol. 14, no. 4, pp. 381–396, 1988.

[13] N. J. Higham and F. Tisseur, "A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra," *SIAM Journal on Scientific and Statistical Computing*, vol. 21, no. 4, pp. 1185–1201, 2000.

A ANSYS Input Files

All input files used for the examples presented in this article are included in the following sections. Header information and page numbering has been removed to simplify copy-and-paste operations.

A.1 Testing the Hager algorithm in ANSYS

```
! calculating the condition number of a Hilbert matrix
! using Hager approximation algorithm

n = 12

! fill the elements according to a Hilbert matrix
*DIM,h,ARRAY,n,n
*DO,ii,1,n
  *DO,jj,1,n
    h(ii,jj) = 1/(ii+jj-1)
  *ENDDO
*ENDDO

! the Hilbert matrix is symmetric, but the transpose
! will be calculated here anyway in case nonsymmetric
! matrices will be analyzed
*MFUN,hT,TRAN,h

! one-norm takes sum of max abs val in row
! (same as infinity-norm on the transpose)
normh_1 = 0
*DO,ii,1,n
  *VABS,,1
  *VSCFUN,max,MAX,hT(1,ii)
  normh_1 = normh_1+max
*ENDDO

! the inverted matrix will be used for comparison
! to the "gamma" obtained with the Hager algorithm
*MOPER,invh,h,INVERT
*MFUN,invhT,TRAN,invh

! one-norm takes sum of max abs val in row
! (same as infinity-norm on the transpose)
norminvh_1 = 0
*DO,ii,1,n
  *VABS,,1
  *VSCFUN,max,MAX,invh(1,ii)
  norminvh_1 = norminvh_1+max
*ENDDO


! the Hager algorithm starts here

*DIM,xi,ARRAY,n
*DIM,e_j,ARRAY,n
*DIM,x,ARRAY,n
*VFILL,x,RAMP,1/n

*DO,jj,1,n
  *MOPER,y,h,SOLV,x
  *DO,ii,1,n
    xi(ii) = SIGN(1,y(ii))
  *ENDDO
  *MOPER,z,hT,SOLV,xi

  ! infinity-norm takes sum of max abs val in col
  *VABS,,1
  *VSCFUN,z_inf,MAX,z

  *MFUN,zT,TRAN,z
  *MOPER,zTx,zT,MULT,x

  *IF,z_inf,LE,zTx(1,1),THEN
```

```
    ! one-norm takes the sum of max abs val in row
    *VABS,,1
    *VSCFUN,gamma,SUM,y
    *EXIT
  *ELSE
    *VFILL,e_j,RAMP,0
    e_j(jj) = 1
    *VFUN,x,COPY,e_j
  *ENDIF
*ENDDO

/COM
/COM,condition number obtained using Hager algorithm
kappa_hager = normh_1*gamma

/COM
/COM,condition number obtained using matrix inversion
kappa_inv = normh_1*norminvh_1
```