What is your structural model not telling you?
Finding hidden modelling errors and inaccuracies in your analysis results.

Dr Ramaseshan Kannan PhD [1], Dr Stephen Hendry PhD [2], Prof Peter Debney [3]

[1] Oasys, Technical Software Group, Arup, 3 Piccadilly Place, Manchester, MN1 3BN, United Kingdom; T +44 20 7755 3601; email: Ramaseshan.Kannan@arup.com

[2] Oasys, Technical Software Group, Arup, 13 Fitzroy Street, London, W1T 4BQ, United Kingdom; T +44 20 7755 3694; email: Stephen.Hendry@arup.com

[3] Oasys, Technical Software Group, Arup, 78 East Street, Leeds, LS9 8EE, United Kingdom; T +44 113 237 8116; email: Peter.Debney@arup.com

ABSTRACT

Finite element analysis involves inherent approximations and numerical errors. In addition to these, the increasing size of structural models and the use of automated workflows for creating them can lead to hidden user errors in the said models. For the engineer to have confidence in analysis results, it is necessary to be aware of how these errors manifest themselves in models, what impact they have on analysis results and, most importantly, how they can be detected. We present novel numerical techniques that the analyst can use to ‘debug’ their models and verify the accuracy their analysis results. These techniques have been implemented in software and have been successfully used by practicing engineers working on live projects.

INTRODUCTION

Since the first use of computers for structural analysis (Felippa, 2001), the computational power available to engineers has increased dramatically. In turn, engineers have been quick to exploit these capabilities and finite element models have grown significantly both in size and in complexity. Other developments, such as generating analysis models from Building Information Models (BIM) and parametric methods, have also contributed this growth. These improvements in the workflow also mean that the engineer is more separated from the resulting model, and thus finds it more difficult to establish the integrity of that model. To gain the benefit from the automated workflow the engineer needs new techniques to understand the complexities of the resulting models. In this paper, we discuss the implementation of new tools based on innovative numerical methods that allow engineers to find hidden errors in their structural analysis models and ensure the robustness of analysis results.

The use of the finite element method for structural analysis involves approximations at various levels. These are:

- Idealization of the structure and its behavior
- Discretization of the governing equations of motion and of equilibrium
- Reducing real numbers down to floating point numbers with finite precision and their subsequent use

Each layer of approximation introduces errors in the computed answer. Techniques for understanding, analyzing, and bounding these errors have developed in parallel with the Finite Element Analysis method. On the other hand, user errors, i.e. errors that are caused by incorrect use of the software or mistakes in the model definition, though common, have received much less attention in academic literature. Examples of such errors include

- Lack of connectivity: adjacent elements that are supposed to share a common node but are connected to different nodes that are coincident, resulting in one of the elements having insufficient restraint.
- Failure to set member end connections correctly, which can occur if a beam is free to rotate about its axis although in the real structure there is a nominal restraint against rotation.
- Modelling beam elements with large sections and/or very small lengths, often the result of importing FE assemblies from CAD models.

Irrespective of whether the error arises from approximation or from erroneous input data, it can lead to inaccuracies in results such as displacements. Such errors can propagate to results such as forces that are subsequently computed using the displacements.

In this article, we first present a short discussion on how inaccuracies arise in structural analysis results while solving problems on computers and demonstrate how modelling errors can amplify these inaccuracies. We then present two novel techniques that engineers can use to detect hidden errors in their FE models and detect potential errors

in force and moment calculations. These tools are being used by engineers to gain confidence in the robustness of their analysis results.

SOLVING PROBLEMS NUMERICALLY ON COMPUTERS

A structural analysis model built and analyzed on an Intel- or AMD-processor based computer (These are also referred to as x86 processor architectures) using a modern-day software package both stores and performs arithmetic on real numbers with finite precision. Such a representation is called a floating-point number and is governed by the IEEE 754 standard. In simple terms, any real number is represented as a floating-point number in the form

$$significand \times base^{exponent}$$

Where the significand is the first n digits of the real number (the precision increases with the size of n), the base (10 if we are using decimal numbers), and the exponent is the value required to return the significand to its original value. For example:

$$42030 = 4.203 \times 10^4 = 4.203e4$$

$$0.0078900 = 7.89 \times 10^{-3} = 7.89e-3$$

$$3.14159265 = 3.142 \times 10^0 = 3.142e0$$

The truncation of the real number reduces its accuracy. If the real number is x then the floating point number FL(x) given by the relationship (Higham, 2002)

$$FL(x) = x(1+\delta)$$

where δ is the round-off error whose magnitude is bounded by a constant that depends on whether the software uses single or double precision. For our discussion in subsequent sections, we shall assume that our computations are done in double precision, i.e. we have about 16 decimal digits of precision. Based on the representation above, an arithmetic operation such as the addition of two floating point numbers will introduce an error in the result as given by

$$FL(a+b) = (a+b)(1+\delta).$$

The arithmetic model above can be easily extended to other operations such as multiplication or exponentiation and, therefore, it is easy to see that a general function f introduces an error of Δy in the result y when it acts on x:

$$y + \Delta y = f(x + \Delta x).$$

The best we can hope for is for Δy to be of the same or smaller order of magnitude as Δx. The error Δy can be affected by both the algorithm used to compute f and the sensitivity of f to changes in x.

To illustrate how errors in x propagate to large errors Δy in y, we present two small examples.

The first example is of a function whose computed solution is outside its analytical range. Consider

$$y = \frac{1 - \cos x}{x^2}$$

With some analysis, it can be seen that y is bounded by 0 and 0.5 for all values of x, i.e., $0 < y < 1/2$.

We wish to compute $y(x = 1.2{\times}10^{-5})$ on a hypothetical computer with 10 digits of precision, i.e., any number with more than 10 decimal digits gets rounded to one with 10 digits or fewer. This is purely for illustration as x86-based computers support up to 16 digits of precision. On such a hypothetical computer,

$$\cos(1.2{\times}10^{-5}) = 0.999\,999\,999\,9 \implies 1 - \cos x = 0.000\,000\,000\,1$$

and

$$(1.2{\times}10^{-5})^2 = 0.000\,000\,000\,1$$

which gives us

$$y(x = 1.2{\times}10^{-5}) = 1$$

Clearly our computed value is outside the range of the function!

The error is caused because of how we chose to compute the function. To fix it, we must change the algorithm used to compute y. Rewriting it as

$$y = \frac{1}{2}\left(\frac{\sin(x/2)}{x/2}\right)^2$$

will yield correct result.

The second example comes from the use of Gaussian elimination, a common method for solving a linear system of equations on a computer. We wish to find the solution of the following simultaneous equations on a computer with 3 digits of precision:

$$0.913x_1 + 0.659x_2 = 0.254$$

and

$$0.780x_1 + 0.563x_2 = 0.217$$

Writing the system as a $2{\times}2$ matrix

$$\begin{pmatrix} 0.913 & 0.659 \\ 0.780 & 0.563 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.254 \\ 0.217 \end{pmatrix}$$

we try to introduce a zero in the first column of the second equation. To do this we multiply first column by the ratio $0.780/0.913$ and subtract the result from the second column to give

$$\begin{pmatrix} 0.913 & 0.659 \\ 0 & 0.001 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.254 \\ 0.001 \end{pmatrix}$$

Doing so leads us to the solution

$$x_1 = -0.443, x_2 = 1$$

But it is easy to verify, by means of solving the equation by hand, that the actual solution of the equations is

$$x_1 = 1, x_2 = -1$$

Therefore, our computed solution not only had the wrong sign, it had a relative error larger than 100%! The reason for the erroneous answer lies in the sensitivity of our matrix to the inversion function. The fix in this case is to simply use more precision. Repeating the same solution with 6 digits of precision gives us the same answer as the exact solution.

NUMERICAL ERRORS AND STRUCTURAL ANALYSIS PROBLEMS

So what is the connection between the above discussion and structural analysis problems?

As mentioned in section 1, there are several sources of approximations leading to errors in structural analysis results. Central to the computation of these results is the solution of a linear system of equations given by $Ku = f$ for a stiffness matrix K, loading f and displacement u. Note that our notation is to use capital letters for matrices and small for vectors and scalars. These linear systems arise in several types of analysis, which include

Static:

- solve $Ku = f$ for displacements u.

Static P-Δ :

- solve $Ku_1 = f_1$ for an initial set of loads f_1, and
- solve $(K + K_g)u_2 = f_1$ for u_2, where the geometric stiffness matrix K_g is formulated using element forces computed from u_1.

Modal dynamic:

- solve $Ku = \lambda Mu$ for natural frequencies $\sqrt{\lambda}$ and vibration mode shapes u, where M is the mass matrix of the finite element assemblage.

Buckling:

- solve $Ku_1 = f_1$ and $Kv = \lambda K_g v$ for buckling modes v and buckling load factors λ, where K_g is the same as in P-Δ analysis.

When solving a linear system of equations involving K, there is an error in the computed displacements u. The error depends not only on errors in K but also on the sensitivity of the solution to small changes. The latter is referred to as the conditioning of the problem. The conditioning of a problem determines the maximum (i.e. worst-case) change in the solution in response to a small change in the input.

ESTIMATING THE CONDITIONING OF THE STIFFNESS MATRIX

To ensure the robustness of results, it is necessary to ensure the stiffness matrix is well conditioned. This conditioning is measured by a numerical property called the condition number of a matrix, which is often denoted by κ. Most textbooks on finite element analysis prescribe computing κ using its definition as the ratio of the largest and the smallest eigenvalues of K. The challenge in doing so is that the extremal eigenvalues are computationally expensive to compute, in fact, it is more expensive to compute κ than to solve $Ku = f$! However, advances in numerical analysis (Tisseur & Higham, 2000) (Kannan, Numerical Linear Algebra Problems in Structural Analysis (Doctoral thesis), 2014) allow κ to be estimated to good accuracy for a fraction of the cost required to compute it. We implemented this algorithm in the structural analysis software package Oasys GSA.

The smallest value of κ is 1. Such a matrix is said to be well-conditioned. The largest theoretical value of κ is infinity and a matrix with this condition number is singular, i.e., it has no unique solution.

The condition number also gives us a rule of thumb for the worst-case accuracy of u. There can be as few as $16 - \log \kappa$ digits of accuracy in the computed solution. A matrix with κ equal to 10^{16} can therefore result in solution with not a single digit accurate (it is singular) and the results from a matrix with $\kappa > 10^{11}$ must be treated with caution.

The presence of user errors in finite element models can lead to a large condition number. Such errors, examples of which we provide in Section 2, can result in zones of excessive high or low stiffness in structural models. In large and complex models, they are not just difficult to spot, it is hard to know if they exist at all. Our approach in GSA was to estimate the condition number of the matrix each time a linear elastic analysis was executed and report it as part of the analysis log. The reporting gives engineers feedback on when their model has potential issues that could affect the accuracy of their results. In some cases, it is simple to identify the cause of the ill-conditioning warning, for instance, a missing global restraint that can set up a rigid body motion for the whole model. In most cases, however, it is tedious and time consuming to find the cause. What the engineer wants is a way to quickly pinpoint where there are potential problems, to simplify the fixing of modelling errors.

A NEW TECHNIQUE TO DETECT THE CAUSES OF ILL-CONDITIONING

Our investigation of ill-conditioning issues led to a new numerical analysis technique for detecting their cause. Our method, which is presented in (Kannan, Hendry, Higham, & Tisseur, 2014), and implemented in Oasys GSA, has been used successfully by engineers to correct their models. We present a summary description of the method in this section and refer the reader to the original paper for further detail.

Errors that cause ill-conditioning belong to one of the following two categories

- Lack of stiffness. This can arise, for instance, when parts of the model are insufficiently restrained or nodes are incorrectly connected.

- Disproportionately large stiffness. These can typically occur when certain elements have large second moments of area but small lengths.

Whilst both categories can generally result from any user error, they are particularly common when models have been imported using automated processes from BIM or CAD packages.

Our method is called Model Stability Analysis and it uses the eigenvectors of the stiffness matrix to formulate 'virtual energies' for elements in the model. It can be shown mathematically that elements with large virtual energies pinpoint parts of the model that either lack stiffness or are disproportionately stiff.

When a model generates an ill-conditioning warning, the engineer can run a Model Stability Analysis and graphically display elements that have large relative virtual energies. An examination of the model (e.g. support conditions, nodal connectivity or cross-section properties) near the said elements would reveal the anomaly that causes ill-conditioning. Once the anomaly is fixed, the engineer re-runs the analysis to ensure the condition number reduces and, if not, runs a Model Stability Analysis again.

We now present an example of the use of Model Stability Analysis on a model that returned ill-conditioning warnings during its analysis. Our example is set in the context of GSA but the underlying numerical analysis is not specific to a software package.

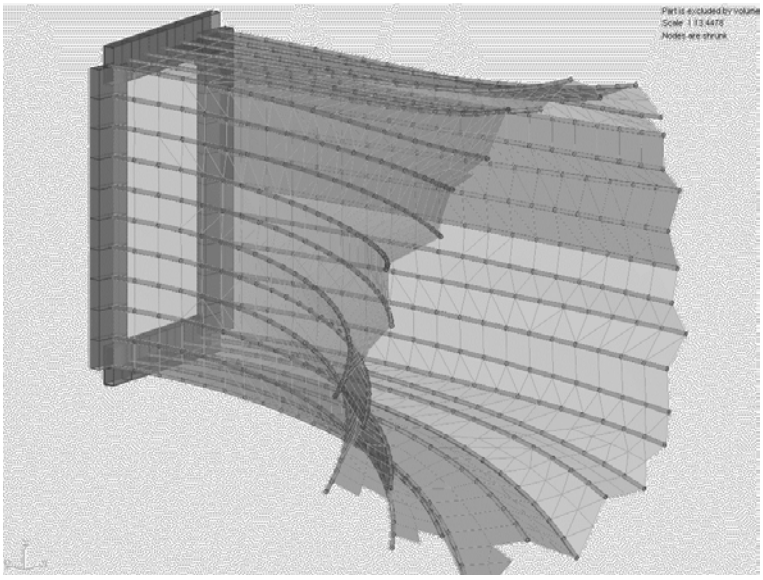


Figure 1 A small portion of the facade model

Figure 1 shows a portion of a larger model of a façade cladding of a structure that consists of 32,000 elements and 21,000 nodes resting on pinned supports. The glass façade panels are modelled using four-noded plane-stress elements. Each panel rests on the grid of beams through springs at each of its four corners, as shown in the zoomed-in view in Figure 2. The element connectivity is illustrated in Figure 4(a).

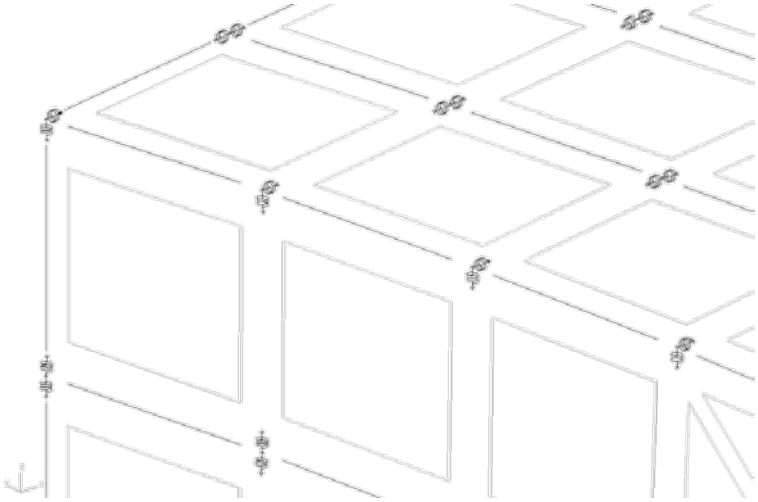


Figure 2 Close-up view of element connectivity. Blue quadrilaterals model facade panels, green lines represent beams and springs are drawn as coils. Gaps between the elements are a graphic view setting and are only for visual clarity.

An early version of the model triggered a 'large condition number' warning with a condition estimate of 10^{12}. Model Stability analysis was run and it identified 2 elements in the model with large virtual energies (Figure 3). On investigation, we found the following error in the nodal connectivity. The corner node of the said plane elements did not share a node with the spring; instead it was connected to a different node that was in the geometric vicinity of the spring element (Figure 4). Thus, the plane elements were unsupported and free to flap about – an error that could lead to potentially incorrect results. Fixing the nodal connectivity error brought the condition number down to 10^{8} and the model analyzed without the ill-conditioning warning.

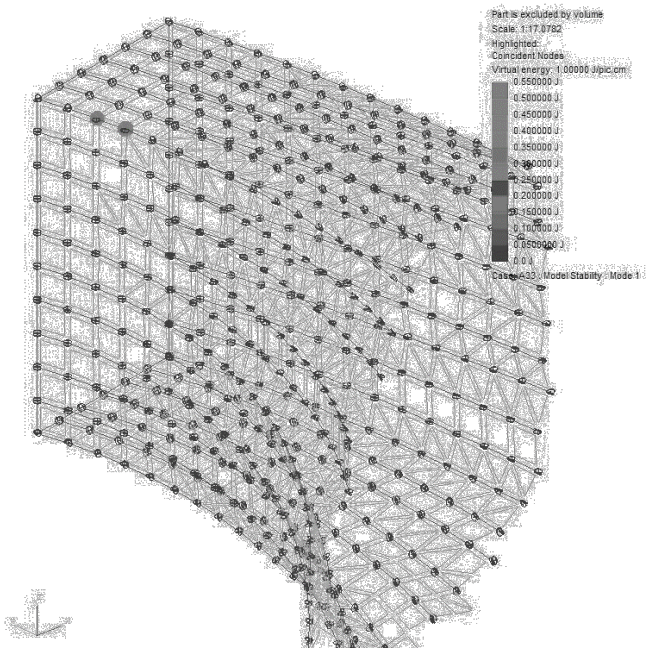
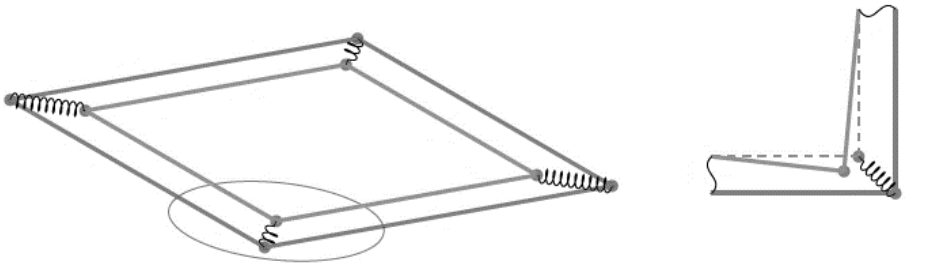


Figure 3 Element virtual energy visualization.



(a) A closer view of element 13828

(b) Corner node connectivity

Figure 4 Nodal connectivity errors for element with large virtual energy

The above is one of the many situations where Model Stability analysis allows the detection of modelling errors that would otherwise be hidden from the modeler – for more examples, we refer the reader to (Kannan, Hendry, Higham, & Tisseur, 2014).

ACCURACY OF ELEMENT FORCE CALCULATION

A second quality analysis included in GSA is that calculating the loss of accuracy due to element size related to the distance of the element from the origin and size of rigid body motion.

In a finite element analysis, element results are based on strains which are a function of relative displacements. Hence, even if the displacement results are accurate enough, there can be inaccuracies in computing strains in elements. By comparing the element displacement with the element distortion, the number of significant figure lost in the calculation of element results can be assessed. Displaying this graphically allows the engineer to identify parts of the model where the results are relatively less accurate.

If we consider a simple bar element (with axial effects only) the force in the bar can be calculated provided we know the strain, from

$$f^{(e)} = AE\epsilon.$$

And assuming constant strain in the element

$$f^{(e)} = AE\frac{(u^{(2)} - u^{(1)})}{l}$$

where l is the length of the bar and the strain depends on the relative displacement $u^{(2)} - u^{(1)}$.

To illustrate this, consider a 1 m steel bar under stress so that corresponding strain is 1x10-3. This means that there is an elongation of 1 mm. If this is part of a large structure, there may well be a rigid body displacement of the bar of say 100 mm meaning that in calculating the force (stress) in this element we are losing two significant figures.

The rigid body displacement u_R is the average displacement of the element

$$u^{(R)} = \frac{(u^{(2)} + u^{(1)})}{2}.$$

Therefore, the displacement causing straining at the nodes is

$$u^{(D,i)} = u^{(i)} - u^{(R)}, \quad i = 1,2.$$

If the rigid body displacement is large compared with the distortional displacement, there can be a loss of significance in computing $u^{(D,i)}$. In the worst case the relative error can be as large as $\|u^{(R)}\|/\|u^{(D)}\|$ and therefore we could lose up to n significant figures given by

$$n = \log_{10} \frac{\|u^{(R)}\|}{\|u^{(D)}\|}.$$

This loss of accuracy can therefore propagate to the force and moment results for the element. Obviously the shorter the element and greater the rigid body displacement the greater the loss of accuracy in the force or stress calculation.

The above can be generalized for an element with more than two nodes each with six (or fewer) degrees of freedom in a straightforward manner (Oasys GSA 8.7 Software Manual, 2015). We shall omit the derivation to favor brevity, but note that in the case of more general case, we have multiple values of 'loss of accuracy' for each element. However as this is essentially a qualitative analysis, rather than assigning the values to nodes it is sufficient to simply record the maximum loss of accuracy value for the element.

This algorithm has been implemented in Oasys GSA. For the simple model shown in Figure 5 below the diameter of the contour blobs indicates the 'loss of accuracy'. It is noticeable that the short (and therefore stiffer) elements have, as expected, a greater loss of accuracy. The closer n (the digits of lost accuracy) gets to 16, the greater the likelihood of inaccuracy in the forces.

CONCLUSIONS

As the analysis power available to engineers continues to increase the size and complexity of models will continue to grow. While an engineer should carry out simple checks to verify the suitability of his model, there is a danger that the complexity means that analysis becomes a black box – especially with automated workflows driven from BIM and CAD. Numerical conditioning and accuracy issues are under-discussed in standard FE material in both academic and industry. This can lead to problems when the engineer is unable to properly assess how well his model represents reality. To have confidence in the modelling it is important that the engineer can detect problems and have confidence in his results.

While mathematical techniques can be used to make some of these assessments it is important that the results of these assessments can be presented in a way that is meaningful to the engineer. Ultimately the best engineering solution depends on the engineer understanding the limitations of his model and knowing what confidence to

place in the analysis. It is the authors' hope that what is presented here is a step in that direction.

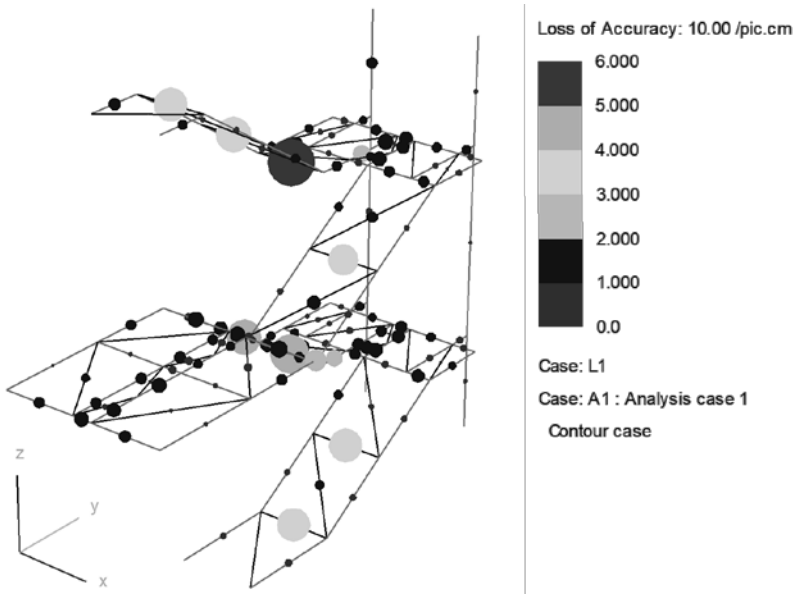


Figure 5 Loss of accuracy in element force/moment results.

BIBLIOGRAPHY

Felippa, C. A. (2001). A historical outline of matrix structural analysis: a play in three acts. *Computers and Structures*, 1313-1324.

GSA Suite. (n.d.). Retrieved from Oasys Software: http://www.oasys-software.com/gsa

Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms.* Philadelphia: Society for Industrial and Applied Mathematics.

Kannan, R. (2014). *Numerical Linear Algebra Problems in Structural Analysis (Doctoral thesis).* Manchester, United Kingdom: School of Mathematics, The University of Manchester. Retrieved from http://eprints.ma.man.ac.uk/2195/

Kannan, R., Hendry, S., Higham, N., & Tisseur, F. (2014). Detecting the causes of ill conditioning in structural finite element models. *Computers & Structures*, 79-89.

Oasys GSA 8.7 Software Manual. (2015, November). *https://www.oasys-software.com/media/Manuals/Latest_Manuals/gsa8.7_manual.pdf.* Retrieved from oasys-software.com/gsa.

Tisseur, F., & Higham, N. (2000). A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. *SIAM Journal of Matrix Analysis and Applications*, 1185-1201.