Symbolic Finite Element Analysis for Parametric Studies

R.W. Johnstone*, Toshiyasu Shimizu**, and M. Parameswaran*

* Institute for Micromachine and Microfabrication Research, School of Engineering Science
Simon Fraser University, 8888 University Dr., Burnaby, BC, V5A 1S6, Canada
** Department of Mechanical and Environmental Informatics
Tokyo Institute of Technology, 2-12-1 Ookayama, Meguroku, Tokyo, 152-8552, Japan

ABSTRACT

This paper presents the theory behind the development of a computer representation of symbolic expressions suitable for use in finite element analysis. This representation is used to replace standard floating-point calculations so that the result of the analysis, instead of being a numeric result, is an equation relating the design variables to system behaviour. Representations based on rational multivariate polynomials and rational multivariate orthogonal polynomials are discussed. Some preliminary results on the accuracy of our initial implementations are presented.

Keywords: finite element analysis, symbolic computation, parametric analysis

1 INTRODUCTION

When analytical methods fail, scientists and engineers often turn to numerical methods [1], [2]. Although numerical methods can only approximate a solution, the great speeds of modern computers can make the errors due to the approximations very small. In particular, finite element analysis (FEA) can provide predictions that would be far too difficult to obtain in other ways. Still, although numerical results can be valuable, closeform solutions are preferable.

The importance of numerical methods to finite element analysis is so great that FEA is always assumed to be numerical. This, however, is not the case. FEA can also be used to generate symbolic results; these results are mathematical expressions. For example, FEA can provide an excellent approach for deriving the formulas expressing force/displacement relations in structural systems. The mathematical tools provided by finite elements methods and by numerical methods can be used independently.

Despite the focus on numerical methods, symbolic methods do play a role in FEA. Symbolic methods are used to derive stiffness matrices, element matrices, and even entire finite element formulations [3]–[5]. However, once the stiffness matrix for a problem has been constructed, the next step is almost always numerical. For example, in a force-displacement analysis, the stiffness matrix needs to be inverted to obtain the solution. The matrix inversion is much easier when done numerically due to symbolic growth.

Symbolic computation software (SCS) is well developed, and used in many different types of applications. However, it cannot be used directly in solving finite element problems. Exponential symbolic growth limits naïve implementations of symbolic finite element analysis to a very small number[1] of degrees of freedom.

In this paper, we will present a method of performing finite element analysis using a symbolic kernel. The representation of numbers used in analysis is symbolic expressions instead of floating-point numbers. Unlike standard finite element analysis, the end product of symbolic finite element analysis is an equation [6].

We have developed a number of methods of measuring and limiting symbolic growth. We can thus use finite element methods to determine equations capable of predicting system parameters based on the design parameters. This allows us to perform symbolic FEA on much larger systems then otherwise possible.

Although the symbolic FEA does require more computational effort, the results of symbolic FEA are also considerably more useful. With the results in hand, design parameters can be changed and the results of the analysis computed with minimal effort. This situation contrasts sharply with traditional FEA methods, which requires the entire analysis to be repeated in its entirety.

Such equations have many uses. In particular, they could speed multiple evaluations of similar models. This would aid many important types of analysis, such as probabilistic design studies (PDS), design optimization (DO) problems, and reduced order modeling (ROM).

2 KERNEL DEVELOPMENT

The key idea behind symbolic finite element analysis (SFEA) is to replace the field[2] used to perform calculations from real numbers to equations. For the computer implementation, the elements of the chosen field must have some representation in memory, and algorithms for the various mathematical operations. We refer to a par-

[1] The limit is around 25 degrees of freedom.

[2] A field is an algebraic structure in which the operations of addition, subtraction, multiplication, and division (except division by zero) may be performed. Other requirements are also present.

ticular representation of a field, and the necessary algorithms, as a kernel. SFEA thus means replacing the use of a kernel based on floating-point numbers with a kernel based on data structures that represent equations.

As previously mentioned, standard SCS is not a good choice for the kernel. With each operation, the complexity of the symbolic representations grows. The analysis thus quickly bogs down in manipulating huge equations. This places a severe limit on the size of FEA problems that can be handled with this approach.

Polynomials do not form a field[3]. However, they do form a good illustration for how symbolic growth occurs. When a polynomial of a single variable of degree n is multiplied by another polynomial of degree m, the resulting polynomial will have degree $n + m$. Thus, as the analysis progresses, the polynomials require increasing amounts of memory and time to manage.

However, the above growth is not limited to polynomials, but also applies to floating-point numbers. When a floating-point number with n digits is multiplied by another number with m digits, the result has $n + m$ digits. Fortunately, computers silently truncate the results back to a fixed upper bound. Thus, as calculations progress, the size of the floating-point representations is fixed, and runaway growth is avoided. We thus set out to develop analogous techniques applicable to the equations used in FEA.

A key observation in our development of new kernels for SFEA was that the elements of the system matrices in most FEA problems were rational multivariate polynomials. Rational polynomials, unlike simple polynomials, form a mathematical field, and thus form a very good representation for our kernel. If the growth of the polynomials in both the numerator and denominator can be limited to a fixed bound, rational polynomials would then be eligible for use as a kernel. Further, the algorithms for mathematical operations are very simple.

2.1 Non-dimensional Finite-element Formulations

In general, one can not determine which term of a polynomial, numerically, is most significant, and which is least significant. However, if the variables' ranges are fixed to $[0, 1)$, then an ordering is possible. Terms of higher degree are less significant. The terms in polynomials, including multivariate polynomials, can thus be ordered by degree, and terms that exceed a certain threshold truncated.

Some types of finite-element formulations can be recast into a nearly non-dimensional form, where the local stiffness matrix, and therefore the global stiffness matrix, consists of only non-dimensional variables in the range $[0, 1)$. Once this is done, polynomial truncation based on degree is possible.

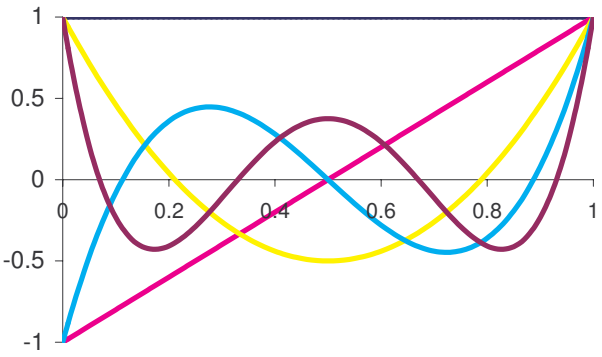

Figure 1: Plot of the first 5 Legendre polynomials over the range zero to one. The polynomials can be identified by the number of extrema they possess.

This approach is still rather simplistic, and is not particularly robust, and can be improved in a number of ways. For example, many variables will have ranges even smaller than $[0, 1)$, and many terms will have large coefficients. Thus, truncation algorithms based on the terms' maximum values are perhaps more accurate.

However, this approach is limited in two fundamental ways:

- The accuracy is very dependent on the actual values that the variables take. Mainly, since 1^n is still one for any value of n, truncation can become questionable.
- Not all FEA problems can be formulated in a non-dimensional form with variables fixed to the range $[0, 1)$.

The above two limitations are addressed by moving to a more generic framework, which is outlined in the following section. However, the kernel described above may still be useful. A kernel based on a monomial representation of polynomials is significantly simpler than a kernel based on orthogonal polynomials, and should be speedier as well.

2.2 Multivariate Orthogonal Polynomials

Multivariate orthogonal polynomials (MOPs) [7] are an extension of standard orthogonal polynomials[4] to the multivariate case.

The main advantage of MOPs as a basis for the kernel, instead of the monomial terms just discussed, is that the orthogonal polynomials are more strictly ordered. Polynomials of higher degree capture finer details of the true function's shape (figure 2.2).

[3]Since, in general, polynomial division does not result in another polynomial.

[4]Familiar cases of orthogonal polynomials are the Lagrange polynomials, Leguerre polynomials, Jacobi, Hermite, etc.

Case	No. of Elements	No. of Nodes	DOF
Case 1	1	4	4
Case 2	3	8	12
Case 3	5	12	20
Case 4	30	62	120

Table 1: Summary of the four structural finite-element problems attempted. Each case run numerically, with Gaussian elimination and rational polynomials, with the Faddeev-Leverrier method and rational polynomials, and with the Faddeev-Leverrier method and rational orthogonal polynomials.

For multivariate polynomials, there will be multiple polynomials of a given degree, and these cannot be ordered. Thus, one can not arbitrarily set the number of polynomials in the basis. This limits the granularity over which the accuracy is controlled. However, the trade-off is a more robust framework for polynomial truncation.

The number of polynomials in the basis can be calculated using the following formuli:

$$C_{n,\nu} = \binom{n+\nu-1}{n} \quad (1)$$

$$S_{n,\nu} = \sum_{i=0}^{n} C_{i,\nu} \quad (2)$$

In equation 1, $C_{n,\nu}$ is the number of multivariate orthogonal polynomials of degree n when there are ν variables[5].

To find the number of terms necessary to complete a basis up to degree n, all the polynomials of degree n or lower must be included. This leads to equation 2, which is the total number of polynomials that will form the basis.

3 RESULTS

We performed a number of small SFEA problems to test polynomial and orthogonal polynomial kernels. Four simple structural finite element problems using two-dimensional linear quad elements was performed (table 3). All four of these cases were run multiple times. First, they were run using numerical techniques to provide a baseline answer. They were then run using different kernels.

The simplest kernel tested was based on rational polynomials over Poisson's ratio, ν, with simple truncation based on degree 30. When using Gaussian elimination to invert the stiffness matrix, this kernel performed very poorly, and had inaccurate results even for Case 1.

[5]Equation 1 also applies to monomials of degree n when there are ν variables.

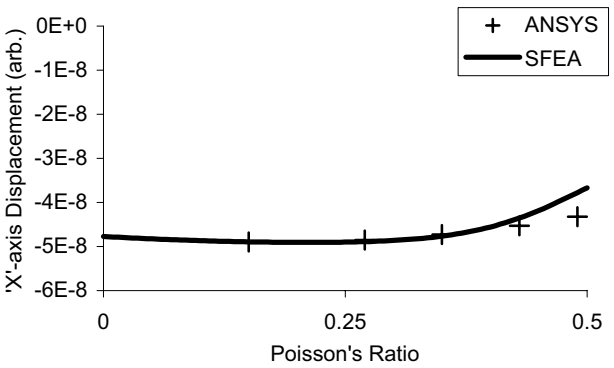

Figure 2: Displacement along 'X'-axis as a function of Poisson's ratio.

A different solution method, Faddeev-Leverrier [8], marginally improved the performance of this kernel.

The rational polynomial kernel was programmed using integer coefficients for the polynomials. Unfortunately, this led to overflow, as the polynomial coefficients become very large. A number of simple solutions were tried, but they were either ineffective at preventing overflow, or led to other numerical problems. We plan to rerun these using floating-point values for the coefficients.

The four FEA problems were also run using rational orthogonal polynomials of degree 5. These performed better, but were still far from acceptable. Case 3 completed, but did agree well with the numeric results (figures 3 and 3). Case 4 was also attempted, but was stopped prematurely because it was very slow compared to the other cases.

The current implementation of the algorithm for multiplying to orthogonal polynomials takes time proportional to $\mathcal{O}(n^3)$, where n is the number of polynomials forming the basis. This is clearly an unacceptable scaling law, especially since we will want to increase the size of the basis. This is not only to increase accuracy, as indicated by Case 3, but also to include additional variables. We believe that an algorithm with time $\mathcal{O}(n^2)$ is possible, but we have not yet completed the implementation.

As it currently stands, the performance penalty of using symbolic methods is many orders of magnitude when compared to numeric methods. This does not completely preclude the usefulness of symbolic methods, but clearly at current speeds numeric methods will still be preferred for PDS and DO. We believe that the performance penalty needs to be reduced to one or two orders of magnitude to be competitive in these applications.

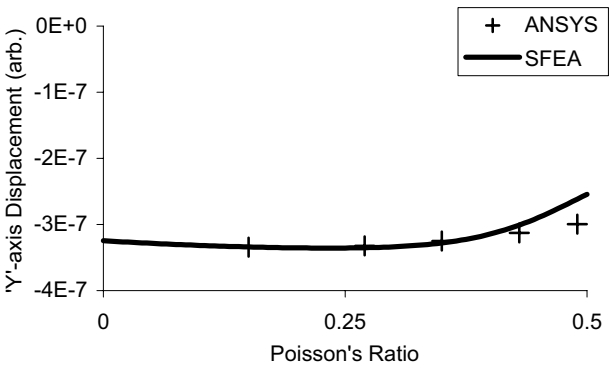

Figure 3: Displacement along 'Y'-axis as a function of Poisson's ratio.

4 CONCLUSION

Symbolic methods could potentially speed up important types of finite element analysis. By expending extra effort up front to derive symbolic expressions, further finite element analysis can be avoided. Finite element analysis need not be entirely numerical.

However, much work remains before SFEA can become widespread. In particular, the speed and scalability of the kernel must be improved.

REFERENCES

[1] J.S. Przemieniecki. *Theory of Matrix Structural Analysis*. Dover Publications, New York, New York, 1985.

[2] S. Moaveni. *Finite Element Analysis: Theory And Application With Ansys*. Prentice-Hall, Upper Saddle River, New Jersey, 1999.

[3] Gustav Amberg, Robert Tonhardt, and Christian Winkler. "Finite element simulations using symbolic computing," *Mathematics and Computers in Simulation*. vol. 49, no. 4-5, pp. 257-74 (1999).

[4] D. Eyheramendy and Th. Zimmermann, "Object-Oriented Symbolic Derivation and Automatic Programming of Finite Elements in Mechanics," *Engineering with Computers*. vol. 15, no. 1, pp. 12-36 (1999).

[5] A. Portela and A. Charafi. *Finite Elements Using Maple: A Symbolic Programming Approach*. Springer, New York, 2002.

[6] Robert W. Johnstone and M. Parameswaran, "Analysis of Flexible Joints for Micromachined Devices," *Proceedings of IMECE'03*. Washington, D.C., 15-21 November 2003.

[7] I. Dumitriu, A. Edelman, and G. Shuman, "MOPS: Multivariate Orthogonal Polynomials (symbolically)," *Preprint*. March 26, 2004.

[8] K.S. Tan, *Symbolic C++: An Introduction to Computer Algebra Using Object-oriented Programming*. Springer, New York, 1998.