

Preface

Finite element methods yield engineering solutions for design analysis. The art of breaking down a formidable boundary value problem into manageable subtasks gave rise to numerical analysis in engineering mechanics. Surprisingly, we have been familiar with such algorithmic constructs, with a variety of notations in different contexts, since our primary school days. We reminisce over our pride of being able to multiply two random digits when we mentally looked up that very useful 9×9 table. Through worded problems, we learned that multiplication was a clever way to perform repeated addition of the same number. And then, long multiplications posed an insurmountable challenge. Our previously mastered skill of addition did not go in vain, however. We merely replaced the 9×9 addition table. We had already used “+,” which was now replaced by “ \times .” These infix notations described binary operations. The technique of carry-over, which is essentially a regrouping concept, was proved by representing a long string of digits as a sum of the list of ones, tens, hundreds, thousands, and so forth. Multiplications did not affect those zeros that were merely place holders. We learned that mathematical proofs were not limited by the number of terms. Then the idea of inverse operations connected addition with subtraction, and multiplication with division via zero and one, which are the respective neutrals. Finally, for binary numbers, *shift-and-add multiplication* substituted the multiplication table with processing lists. To my surprise, I got the same answer whether I used Arabic numerals or transcribed the digits in my native Bengali script. The notion of invariance thus flowered. Then, by repeatedly evaluating a very large number of predefined functions, any arbitrary task should yield a unique list of answers. In mathematical logic, I learned that, ca. 1935, the Church-Turing thesis formalized what could or could not be evaluated within “yet-to-be-built” computers. Fortunately, no aspect of finite elements is *Turing undecidable*! Computers have facilitated stitching together (analogous to carry-over) a very large number of similarly idealized local responses on isolated finite elements. The arbitrary tessellation and the preselected order of local solutions determine the extent of approximations.

Engineering calculations of temperature and stress fields primarily propelled the finite element method to its contemporary popularity. Historically the development of early computers can be attributed to building the jumbo aircraft, especially to performing their finite element analyses.¹ The method became popular in many disciplines that are based on mathematical physics. The intellectual question, then, is how to optimally incorporate computing within mechanics. Turing's work, which even today is far from being widely studied, distinguished *calculating* from *computing*.² By contrast, we have the pedantic formalism of engineering mechanics. In this context, I have been wondering, for the last 40 plus years or so, on how to coalesce these two dissimilar concepts and furnish the minimum basic material for self-teaching.

During approximate evaluation of the temperature scalar and stress tensor in solids, two questions arise. First, what should be the shapes of those broken pieces (that form the body with complicated boundary geometry)? Secondly, to what degree will we strive to capture reasonable *frame-invariant* solutions? Are we satisfied with constant or linear stresses on individual elements, or do we also need quadratic and cubic stress distributions? These specifications dictate the design of an isolated finite element.

Analytical solutions of partial differential equations demonstrate behaviors of physical quantities in space and their evolution in time. But, numerical simulations are essential to establish a new formulation. Approximate techniques bridge the gaps between the elementary and intricate problems. Nevertheless, compression of fundamental steps dictates the structure of our computer programs. Numerics, computation, and programming are intertwined in my everyday research and teaching, and this mindset is reflected in this textbook.

In engineering mathematics courses, the boundaries were amenable to familiar Cartesian and polar coordinate systems. Real-world engineering objects demand various shapes conceived by creative designers. The resulting arbitrary boundary geometry is the reason why analytical solutions cannot be directly employed. Approximate solutions, in spatial modeling, become more tractable from the finite element method than finite differencing.

Series expansions progressively capture the finer details. This observation led Lord Rayleigh to numerically solve many analytically intractable problems.³ Fourier introduced orthogonality that is indicative of independence of functions; essential characteristics of analytical responses appear in those series terms.⁴ Their

¹The late Professor Marvin Minsky pointed this out to me during a private conversation at MIT, ca. 1980.

²Generally speaking, calculation generates an arithmetic-type result to a single problem. Computation is the algebraic task that yields solutions to a class of problems.

³This is the basis of the Rayleigh mode approximation that is elaborated in this textbook.

⁴Courant called these *coordinate functions*—independent components of the solution space. He also used the term *admissible coordinate functions* for boundary value problems.

weights, for elliptic boundary value problems, minimize an energy-type integral in the Ritz variational formulation. This is the backbone of the finite element method that generates approximate numerical solutions.

In 1978, Stephen Wolfram, a teenage PhD student from Caltech, presented a talk to Columbia University's Physics Department. He demonstrated, to the high energy physics research community, his (personal) homegrown C language-based environment for symbolic computation. This inspired me to elucidate engineering mechanics numerics beyond FORTRAN to achieve elegance and clarity via higher-level symbolic constructs. Therein, set theory, matrix algebra, and differential equations can work *in tandem*. This contemplation hibernated in my consciousness for some time, while I concentrated on journal publications of related algebraic formulations.

Mesh generation for plane elements⁵ has been simplified by employing triangulation. Curved-sided elements are used only on the boundary. Many computer algebra programs have built-in functions, e.g., Delaunay triangulations and Voronoi diagrams. The number of element nodes depends on the order of stress profiles.⁶ For incompressible solids, degrees of freedom include the element pressure. The nodal displacements combine the isochoric modes of zero dilatation point-wise. Modal-to-nodal transformations are rectangular matrices. Moore-Penrose⁷ results comply with the physics of the problem.

Mathematica provides a unified programming paradigm to seamlessly carry out numeric and algebraic computations with graphics capability (the appendix includes a brief introduction). Closed-form integration⁸ eliminated the stumbling block to formulating elements in the physical (x, y, z) domain. Curved boundaries and concavity do not pose any difficulty. All derivations and *Mathematica* codes are included in the appendix. Focus on the physics of the problems⁹ may permit a temporary postponement of details in the syntax. In fact, for first-time readers, skipping the *Mathematica* routines¹⁰ will not adversely affect their ability to comprehend physical arguments.

I hope this monograph will not be “yet-another-finite-element-book.” Its distinctive feature is closed-form expressions of vector interpolants that satisfy point-wise equilibrium for all Poisson's ratio. Here, I have generalized (from scalar to vector) the works of Lord Rayleigh, Courant, and Clough. Even though all their publications are in English, it is rather astonishing that very few of us are familiar with their original works¹¹!

⁵Three-dimensional problems are outside the scope of this book.

⁶At least $3 + 2n, 4 + 4n + n^2$ nodes in $\mathbb{R}^2, \mathbb{R}^3$ are needed to pass Irons' *patch tests* in the frame-invariant sense, with n order stress fields from $n + 1$ order displacements.

⁷`PseudoInverse[m]`, for a rectangular matrix m in *Mathematica*.

⁸From the familiar divergence theorem that yielded stress equilibrium and symmetry.

⁹Pertinent continuum mechanics concepts are summarized in the appendix.

¹⁰For industrial applications, C^{++} codes can be generated from those.

¹¹I failed to locate an English version of the Ritz 1908 paper written in German. In his first paragraph, Ritz called his assumed approximation to be a generalization of a truncated Fourier

Within the classical displacement formulation, Clough's interpretation of nodal forces as virtual work quantities resolves, even for incompressible elements, the nonconforming issues altogether.

Truss examples, in the appendix, furnished the template for linear analysis. Even for solid elements, the same *Mathematica* code can satisfy nodal equilibrium and compatibility and then determine stress and displacement unknowns. This is a benefit from algebraic programming where the problem data can specify lengths of different lists.

To conclude, I would like to return to our primary school experiences with arithmetic. Using ± 1 , the four rules of arithmetic are condensed to the operations of addition, multiplication, and exponentiation.¹² Symbolic computation generalizes those basic abstractions within "list operations." They extend the mechanism of addition beyond multiplication and exponentiation for two lists of the same size. Since the functions/operators are symbols themselves, they can be syntactically interchanged, e.g. with differentiation, in deriving the deformation gradient. Also, in conventional inner and outer products, which are contraction and dyadic products for tensors, the multiplication and addition operations can be substituted by any two amenable functions. Table lookup (similar to recalling our memorized addition and multiplication tables up to 9) replaces mathematical handbooks of elementary and transcendental functions of mathematical physics. These basic ingredients transcribe finite element concepts into algebraic functions, where the "rewrite rule" formalism is implemented in the functional programming style. This paradigm furnishes a very powerful engine¹³ for concept development, which constitutes the computational theme of this textbook. Irrespective of intricacies, in tedious substitutions and constructions of long expressions, digital computers perform exactly what we have asked for, faithfully with complete accuracy.

To assure correctness in lengthy algebraic and numerical expressions of shape function vectors and stiffness matrices in the book, I used L^AT_EX to import *Mathematica* results. Since I typeset the book myself, wrote all the *Mathematica* routines, and generated all the figures, I got a thorough training in digital publishing workflow. I became aware of the resemblance between symbolic computation and technical typesetting.

I encourage engineering design analysts of the new millennium to publish more analytically oriented research by harnessing computational thinking.

Columbia University
New York, NY, USA
August 2016

Gautam Dasgupta

series. In his equation (1), the notation ω_n indicated the partial sum of n terms ψ_i , with coefficients a_i , to construct an integral J_n , in (2). Minimization of J_n determined a_i , in (3). He even suggested polynomials for ψ_i . These three equations were employed in solving fourth-order *scalar* partial differential equations for plate problems.

¹²In *Mathematica Plus*, *Times*, and *Power*, respectively.

¹³Based on Church's λ -calculus of ca. 1930.

Finite Element Concepts

A Closed-Form Algebraic Development

Dasgupta, G.

2018, XXXVI, 333 p. 45 illus., Hardcover

ISBN: 978-1-4939-7421-4