A Simple Introduction to Finite Element Analysis

Allyson O'Brien

Abstract

While the finite element method is extensively used in theoretical and applied mathematics and in many engineering disciplines, it remains surprisingly unused within the physics community. This paper is intended to introduce FEA in the context of a numerical solver for physics problems, concentrating primarily on solving Schrödinger's equation over complicated boundaries.

1 Introduction

The Finite Element Method (FEM) or Finite Element Analysis (FEA) is a numerical tool that is highly effective at solving partial and nonlinear equations over complicated domains. It is an application of the Ritz method, where the exact PDE is replaced by a discrete approximation which is then solved exactly. FEM approximates the exact PDE as a matrix equation. The size of the matrix is dependent on (i) the size of the the domain over which the PDE exists and (ii) the desired accuracy of the approximation. Solving a large system accurately requires solving a large matrix equation. More detail will be provided in a later section.

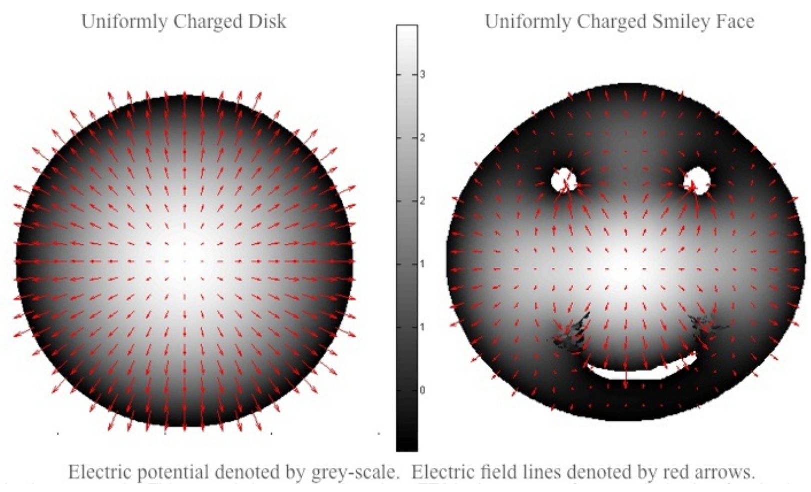


Figure 1: Univformly Charged Disk and Uniformly Charged Smiley Face.

Analytic solutions to physical or mathematical systems are often limited to simple shapes and straightforward boundaries that are rarely the objects of interest in physics[1]. In electrostatics, the values of both the electric potential and electric field lines inside a uniformly-charged disk can be solved using a pencil and paper. Change the domain of the physical system just slightly (say, to that of a uniformly-charged smiley face) and the calculations become much more complicated. In biophysics, FEA can be used to model square-integrable functions over the complicated surface

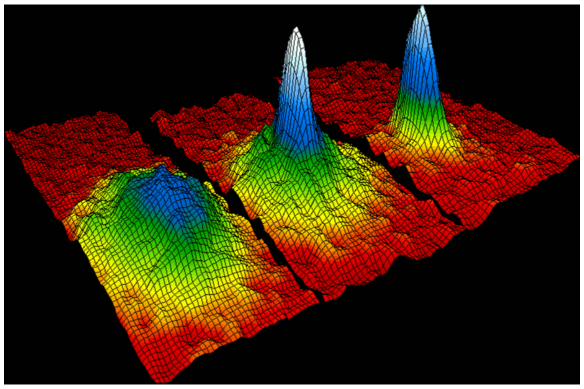


Figure 2: Velocity-distribution data of a gas of rubidium. Pictured left to right: before the appearance of a BoseEinstein condensate, just after the appearance of the condensate, after further evaporation, leaving a sample of nearly pure condensate[3]

.

of a blood cell. As figure 2 shows, FEA can be used in thermal physics to model gases forming a Bose-Einstein condensate. Numerical modeling of Schrödinger's equation is a particularly useful application of FEA in quantum dynamics. Examples of this application will be shown throughout this paper, but first we introduce the basic methods of FEA. These are:

1. Tessellation (discretization of the domain)
2. Matrix computation (computing the approximated PDE)
3. Solving the matrix approximation (eigenvalue problem)
4. Visualization of solutions

2 Tessellation

The finite element method approximates a variational problem as a solvable numerical problem by reducing the degrees of freedom of the system to a finite number. The terms *discretization* and *meshing* are used within FEA literature to describe this step. While FEM can involve tessellation of both space and time dimensions, this paper will only introduce space-dimension tessellation.[5]

In practice, a distribution of discrete points is placed inside and along the boundary of the function. These points, known as vertices or nodes, are where the function will be evaluated. The vertices are then connected to form small, simple and uniform geometries called *finite elements.* The domain of the discretized unknown function is a composite of these tiny geometries.

To better understand the idea of tessellation, an analogy can be made with the more familiar term of pixelation. If the domain of our analytic solution is, say, the shape of a big mushroom from

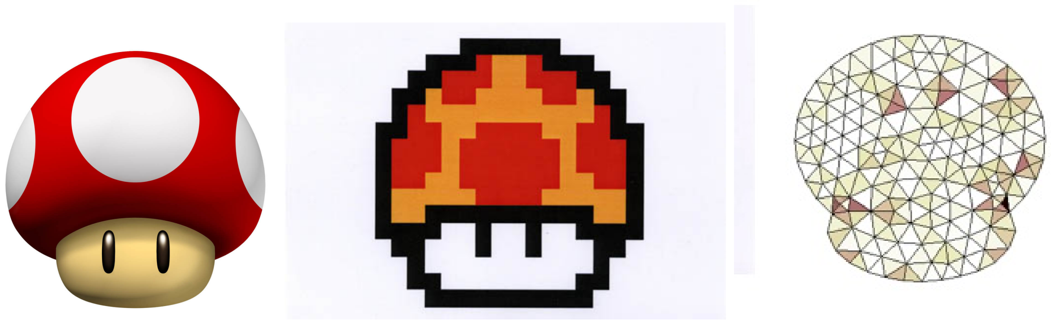

(a) Analytic Mario Function (b) Discretized/pixillated Mario Function (c) Refined Mesh Mario Function

Figure 3: Approximations of Mario's Big Function Mushroom

Super Mario Bros. (Figure 3), the smooth domain of the analytic solution is then "pixelized" into a finite-dimensional function space that can be solved numerically.[1]

The "Mario example" of tessellation provided above shows a 2D object approximated as a composite of tiny triangles. Felippa's book *Introduction to FEM* provides a figure of the types of geometries typically used in one, two and three dimensional finite element methods.[4]

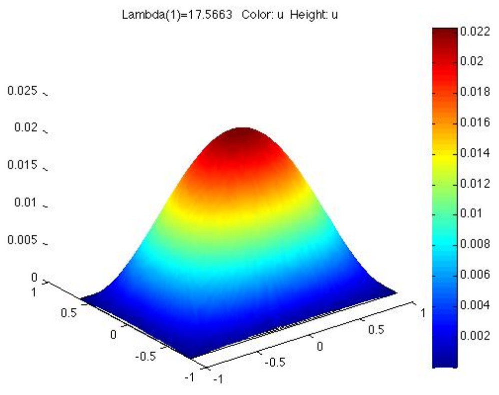
Choosing the appropriate number of vertices used can be tricky. The number of vertices determines the size of the matrix to be diagonalized. Highly accurate approximations produce large, difficult-to-calculate matrices while quicker calculations run the risk of producing shoddy results. Knowing the physics behind a problem can produce more accurate approximations with less computational cost. Given a sense of how the function may vary in space, we can concentrate the distribution of nodes where the system will vary more rapidly and not over-populate portions of the domain with little variation.[6] As an example, consider modeling a melting ice cube over a short time. The physics of heat conduction suggests that we concentrate nodes near the edges of the bound region. The computational cost of a model thus depends on both the physical knowledge of the programmer and the complexity of the problem to be solved.

To illustrate, we "morph" the problem of a particle in a box is morphed into a strange geometry. The first few eigenvectors of the box domain can be calculated using elementary quantum mechanics, but it is more complicated to guess those of the the "not-box" shown in figure 4. The geometry of our not-box is defined and tessellated (figure 5) using MATLAB's `pdetool`, a graphical partial differential equation solver. Refining this mesh (adding more nodes) adds considerable time to calculating the eigenvectors and eigenvalues, while using fewer vertices than shown produces only poor approximations of the ground state solution.

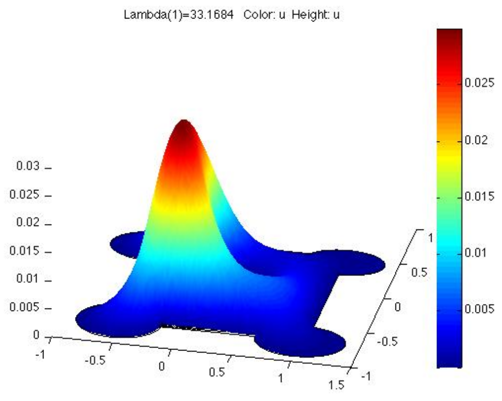
3 Finite element values

Once the bounded region has been tessellated, we solve for the unknown function over the resulting finite elements The overall strategy can be summarized as solving the equation on each of our tiny elements and then "stitching" the results together, though the actual process is somewhat more

[1]Figure 3(b) is a better depiction of meshing in the finite difference method. The finite element method is preserves boundaries quite efficiantly.



(a) GS Eigenvectors of a Particle In A Box



(b) GS Eigenvectors of a Particle In A Not-A-Box

Figure 4: Particle-in-a-box vs particle-in-a-not-box

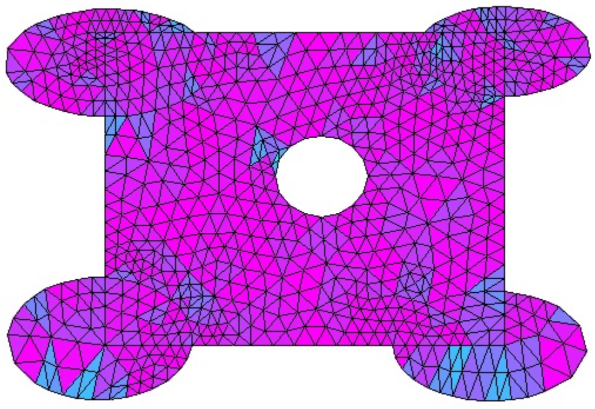


Figure 5: The quality of the geometries is color-coded. Qualities above 0.6 (anything *not* bright blue) are considered acceptable quality by MATLAB [8]

complicated. The goal is to find small solvable matrices that can be substituted into a large $N \times N$ matrix (N is the number of nodes) to form an approximation of the complete system.

Each element is defined by a unique set of identifiable vertices. Within an element, the positions of the vertices are described using natural coordinates consistent with the elements' geometry instead of standard Euclidean coordinates. These coordinates, or basis functions, allow us to compute the solution over each element in terms of the value of the unknown function at the vertices defining the element. In our particle example, the natural coordinates are triangular coordinates. We will label the basis vectors $\zeta_1, \zeta_2, \zeta_3$, and define each such that $\zeta_1 + \zeta_2 + \zeta_3 = 1$. The vertices of our triangular element will be denoted by $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ (see figure 6). Any point on a particular triangle

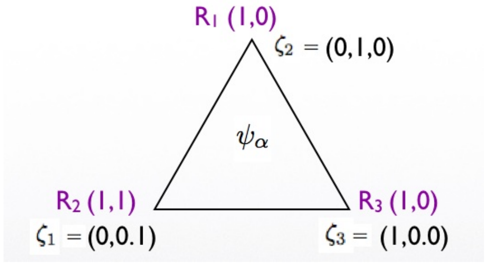


Figure 6: Basis Functions in Natural Coordinates

element T can be described using a weighted sum of the basis vectors:

$$\mathbf{r} = \zeta_1\mathbf{r}_1 + \zeta_2\mathbf{r}_2 + \zeta_3\mathbf{r}_3 \tag{1}$$

We can now write the piece of the unknown function $\psi(\mathbf{r})$ in terms of the vertices as defined in our triangular coordinates:[2]

$$\psi(\mathbf{r}) = \zeta_1\psi(\mathbf{r}_1) + \zeta_2\psi(\mathbf{r}_2) + \zeta_3\psi(\mathbf{r}_3) \tag{2}$$

Using a little geometry[2], the integral of function $\psi(\mathbf{r})$ on T is,

$$\int_T \psi(\mathbf{r})\ dr = 2A\int_0^1\int_0^{1-\zeta_2} \psi(\zeta_1\mathbf{r}_1 + \zeta_2\mathbf{r}_2 + (1-\zeta_1-\zeta_2)\mathbf{r}_3\ d\zeta_1\ d\zeta_2 \tag{3}$$

where A is the area of element T. This is equation shows our function as linear interpolation (curve fitting using linear polynomials). [2] Because the basis functions sum to 1 by definition, our function is already normalized. If our geometry is sufficiently small, we can assume the variation of our function inside the triangle is negligible. We can then approximate $\psi_(\mathbf{r})$ everywhere in the triangle as the solution calculated in the center of our geometry[4].

The integral of the square magnitude of $\psi(\mathbf{r})$ is well defined:[3]

$$\int_T |\psi(\mathbf{r})|^2\ dr = 2A\int_0^1\int_0^{1-\zeta_2} |\psi(\zeta_1\mathbf{r}_1 + \zeta_2\mathbf{r}_2 + (1-\zeta_1-\zeta_2)\mathbf{r}_3)|^2\ d\zeta_1\ d\zeta_2 \tag{4}$$

The above equation shows is a bilinear interpolation (product of two linear functions) of the square of our function.[9] Unfortunately the solution is not as simple as in Cartesian space since our basis functions are not orthogonal. Computing these *overlap integrals* is necessary to solve the "particle-in-a-not-box" system and many other quantum-mechanical problems.

It is important to remember that the natural coordinates for our triangles exist *only* within the triangle.[6] Each triangle T in our domain has a set of basis functions, $\zeta_1, \zeta_2, \zeta_3$ that exist only within the domain of the vertices $\mathbf{r}_{vert}$ describing that triangle. The stationary solutions described in each function $\psi_T$will "add up" to form the overall solution. A brief description is provided in the next section.

[2]described in Fischer [2]

[3]Since our example problem deals only with bound states, the integration shown here uses only the real part of $\psi(\mathbf{r})$. This will not always be the case.

4 Plugging in a function

In the particle-in-a-not-box problem, we defined the boundaries and showed the tessellation of our object in figure 5. We can now solve the Schrödinger equation within each element by defining it in its original variational form and using the variational principle.

$$\delta \int \psi^*(r) \left(-\left(\frac{\hbar^2}{2m} \right) \nabla^2 \Psi(r) + V(r)\psi(r) - E\psi(r) \right) d^3r = 0 \tag{5}$$

The quantum-mechanical variational principle states that if the Hamiltonian is known but the ground state of the Schrödinger equation is unknown, then the ground-state energy is bounded by

$$E_{GS} \leq \langle \psi_T | H | \psi_T \rangle . \tag{6}$$

for any normalized wavefunction $|\psi_T\rangle$.[7] To determine the values of each part of the equation, we consider each term separately and find solutions for potential, kinetic, and total energy in each element in terms of the vertices. Each term will yield a 3×3 matrix for every element. A description of this process can be found in [6].

5 Converting to Cartesian coordinates and solving

The next hurdle is finding a solution in terms of Cartesian (x,y) coordinates. The discrete functions at each element can be expressed in terms of a functional that depends on the vertices of the triangle and the (x,y) coordinates of those vertices.

$$\psi(\mathbf{r}(x,y)) = \zeta_1 \psi(\mathbf{r}_1(x,y)) + \zeta_2 \psi(\mathbf{r}_2) + \zeta_3 \psi(\mathbf{r}_3) \tag{7}$$

This functional can be transformed into an ordinary function $\psi_{x,y}$ by a transformation matrix. To derive the transformation matrix, we start by decomposing our vertices into x and y values. For example, the nodes $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ in the last section will be described by $\mathbf{r}_1 = (x_1, y_1)$, $\mathbf{r}_2 = (x_2, y_2)$ and $\mathbf{r}_3 = (x_3, y_3)$. The interpolation formula still applies:

$$x = \zeta_1 x_1 + \zeta_2 x_2 + \zeta_3 x_3 \tag{8a}$$

$$y = \zeta_1 y_1 + \zeta_2 y_2 + \zeta_3 y_3 \tag{8b}$$

Remembering that $\zeta_1 + \zeta_2 + \zeta_3 = 1$, we can derive a simple matrix to show these relations:

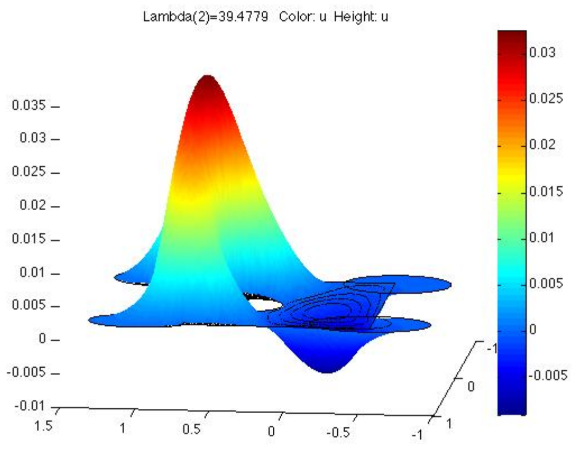
$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} \tag{9}$$

This matrix can be inverted in order to revert to Cartesian coordinates:[5]

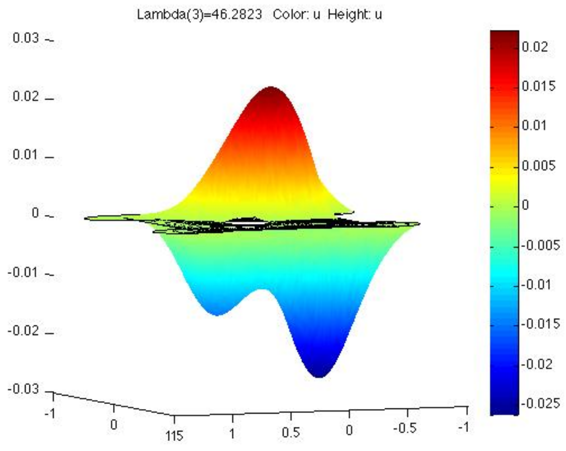
$$\begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} x_1 y_3 - x_3 y_2 & y_2 - y_3 & x_3 - x_2 \\ x_3 y_1 - x_2 y_3 & y_3 - y_1 & x_1 - x_3 \\ x_1 y_2 - x_2 y_1 & y_1 - y_2 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{10}$$

We now have a rubric for transforming back to Cartesian coordinates. The final approximation will be a composite of all overlap integrals (in the form of 3×3 matrices) solved in natural coordinates. This approximation will then be transformed back into Cartesian coordinates by the above transformation matrix. Since all of the points are taken into account, the large matrix will now be have dimensions $N \times N$ where N is the number of nodes. The generalized matrix equation can then be solved.

6 Solutions



(a) First Excited State



(b) Second Excited State

Figure 7: First Two Excited States of our Particle-in-a-not-box.

Solving generalized matrix equations can be computationally demanding, with computing time depending on the software and hardware used. Fast computer algebra systems (software that facilitates symbolic mathematics) such as Mathematica, Maple, and MATLAB ("MATrix LABoratory") are widely available.[8] Most of these programs use the same, or very similar algorithms. Many of these programs come with packages for FEA such as MATLAB's `pdetool`, which we mentioned earlier. The ground state for our particle-in-a-not-box is shown in figure 7(a) and figure 7(b) shows the first few excited states.

7 Conclusion

Though these results are fun to look at, there are many practical problems in quantum mechanics which cannot be easily solved by even the most sophisticated software. Current commercial and academic software for FEM is not often designed with physics research in mind, though it is our hope to change this fact in the not-too-distant future.

References

[1] Ivo Babuska. Generalized finite element methods: Main ideas, results, and perspective. *International Journal of Computational Methods*, 1(1):67103, June 2004.

[2] Christopher J Bradley. *The Algebra of Geometry: Cartesian, Areal and Projective Co-ordinates.* Highperception, Bath, 2007.

[3] Eric A. Cornell, Carl E. Wieman, Eric A. Cornell, and Carl E. Wieman. The Bose-Einstein condensate. *Scientific American*, 278(3):4045, 1998.

[4] C. A Felippa. Introduction to finite element methods, 2001.

[5] C. A Felippa. A compendium of FEM integration formulas for symbolic work. *Engineering Computations*, 21(8):867890, 2004.

[6] Robert Gilmore. From wave mechanics to matrix mechanics. *Not Yet Published*, 2010.

[7] David J. Griffiths. *Introduction to Quantum Mechanics (2nd ed.)*. Prentice Hall, 2004.

[8] MATLAB. *PDE Toolbox: Uer's Guide*. Mathworks Inc, 1995.

[9] Wikipedia. Bilinear interpolation.