Approximating the Capacitance of Square Plates: A Method of Moments Convergence Study

Samuel J. Wyss†
†School of Nuclear Engineering
Purdue University
West Lafayette, Indiana 47907
E-mail: wysss@purdue.edu

***Abstract*—The Method of Moments (MoM) is applied to simulate the capacitance of an arbitrarily sized, square metallic plate. To model this system *in silico*, the Electrostatic Integral Equation (EIE) is used to simulate the charge distribution for a fixed potential on the metallic plate. The total charge is then determined by integrating element-wise over the surface which is then used to solve for plate capacitance. This procedure is used to evaluate various formulations of the underlying integral equations from approximations of square elements, exact forms, to subdomain collocation. Capacitance values from these methods are first compared to data found in the literature as a verification and validation step and then compared to each other in terms of the mesh size required, and corresponding asymptotic number of operations, for these methods to achieve convergence.**

I. Introduction

Contrary to many other methods in Computational Electromagnetics (CEM), the Method of Moments (MoM) is purely based on the integral form of Maxwell's Equations and underlying Green's Functions [1], [2]. In the context of the Electrostatic Integral Equation (EIE), MoM uses the potential contributed by individual point charges on surface elements to solve for charge density as a function of location. This charge distribution can trivially be integrated over surface elements to solve for the net charge on an surface for an arbitrary geometry and applied potential. From this, the capacitance of said structure can easily be obtained.

In matrix form, the EIE relates the point source response of charges in a given surface element to all other surface elements and the applied potential. The use of Green's Functions in the problem formulation allows for these problems to be solved without the need to spatially terminate the simulation region which is a common source of error in other CEM methods. However, the use of Green's Functions results in fully dense matrices, thus requiring the computational complexities and memory footprint that comes with solving them which at first may be seen as a downside of MoM. However, these matrices only require the discretization of surfaces [1], [2] thereby resulting in much smaller system matrices overall which helps to compensate for the typical $O(n^3)$ computation complexity required to solve these dense systems.

The development and results of this work are laid out as follows. Section II contains a short derivation of the EIE from Gauss's Law followed by the setup of the EIE matrix equations for a circular approximation of square surface elements, an exact form for square surface elements, and finally the subdomain collocation. Section III contains a verification of the model with a capacitance range found in the literature, an explanation of the plate charge distribution, and finally, an analysis of the asymptotic number of operations required to determine the plate capacitance for all three methods. Finally, Section IV contains closing remarks regarding the analysis and potential future work.

II. Mathematical Model

To model these systems *in silico*, an appropriate mathematical model must first be derived from Maxwell's Equations. The development of said model is arranged as follows. Section II-1 contains the derivation of the EIE from Maxwell's Equations followed by surface element discretization into a simultaneous set of matrix equations. Section II-2 outlines three formulations of the system matrix $[A]$ starting with square elements approximated as circles, an exact solution for square elements, and finally, subdomain collocation.

1) The Electrostatic Integral Equation (EIE): From basic electrostatics, the integral form of Gauss's Equation over some surface S is as follows [1], [2]

$$\iint_S \epsilon_0^{-1} g(\mathbf{r}, \mathbf{r'}) \rho_s(\mathbf{r'}) dS' = \Phi(r) \tag{1}$$

where ϵ_0 is the permittivity of free space, $\rho_s(\mathbf{r'})$ is the surface charge density, $\Phi(r)$ is the electrostatic potential, and $g(\mathbf{r}, \mathbf{r'})$ is the corresponding Green's function (integration kernel)

$$g(\mathbf{r}, \mathbf{r'}) = \frac{1}{4\pi|\mathbf{r} - \mathbf{r'}|}. \tag{2}$$

Basic electrostatic theory is also used to enforce a boundary condition for the sufficiently conductive metal plate (as assumed in this problem). This boundary condition requires the surface of the plate to be at an equipotential, thus $\Phi(r) = \Phi_0$ where Φ_0 is an arbitrary electric equipotential. Under these stipulations, the only unknown in this problem is the charge density on the plate surface. To solve, we write the charge density in terms of a series of linearly independent basis functions, $v_n(\mathbf{r'})$, and expansion coefficients, c_n, defined over individual elements as follows [1], [2]

$$\rho_s(\mathbf{r'}) = \sum_{n=1}^{N} c_n v_n(\mathbf{r'}). \tag{3}$$

With the surface charge density discretized, it is now necessary to sample the underlying integral equations in a similar fashion using testing functions $w_m(\mathbf{r})$. With these definitions, (1) can now be expressed as the EIE which is defined as follows [1], [2]

$$\sum_{n=1}^{N} c_n \iint_S \iint_S w_m(\mathbf{r})\epsilon_0^{-1} g(\mathbf{r},\mathbf{r'}) v_n(\mathbf{r'}) dS' dS = \iint_S w_m(\mathbf{r})\Phi_0 dS. \quad (4)$$

Equation (4) can be expressed as the following matrix equation

$$[A]\{c\} = \{b\} \quad (5)$$

where $[A]$ is the system matrix defined as

$$[A]_{mn} = \iint_S \iint_S w_m(\mathbf{r})\epsilon_0^{-1} g(\mathbf{r},\mathbf{r'}) v_n(\mathbf{r'}) dS' dS, \quad (6)$$

$\{b\}$ contains the equipotential boundary condition

$$\{b\}_m = \iint_S w_m(\mathbf{r})\Phi_0 dS, \quad (7)$$

and $\{c\}$ is an array of charge density scalars corresponding to their indexed element which is solved for [1], [2]. The following section will go over several methodologies for defining the system matrix and equipotential boundary condition array.

2) System Matrix Formulations: For simplicity only the following zeroth order basis function will be used for all analyses

$$v_n(\mathbf{r'}) = \begin{cases} 1, & r' \in S_n \\ 0, & \text{else} \end{cases} \quad (8)$$

where S_n is defined as the area of the square surface element n.

The first method that will be investigated involves using a Dirac delta function as the testing function

$$w_m(\mathbf{r}) = \delta(\mathbf{r} - \mathbf{r}_m) \quad (9)$$

where $\mathbf{r}_m$ denotes the center of element m. This testing function can be interpreted as testing each element at it's respective center point only [1]. This testing function will be used for the first two methods discussed.

Using (9), (6)-(7) can be re-written as

$$[A]_{mn} = \iint_{S_n} \epsilon_0^{-1} g(\mathbf{r},\mathbf{r'}) dS', \quad (10)$$

and

$$\{b\}_m = \Phi_0. \quad (11)$$

This leads to the first two formulations of the system matrix $[A]$. In the first case, (10) can be approximated using midpoint integration for all $m \neq n$. The case containing $m = n$ contains a numerical singularity. To extract this singularity, the element can be approximated as a circular element possessing the same area as S_n. This results in the following formulation of the system matrix [1], [2],

$$[A]_{mn} = \begin{cases} \frac{S_n}{4\pi\epsilon_0 |\mathbf{r}_m - \mathbf{r}_n|}, & m \neq n \\ \frac{1}{2\epsilon_0}\sqrt{\frac{S_n}{\pi}}, & m = n. \end{cases} \quad (12)$$

For the duration of this paper, this method will be referred to as the circular element approximation.

The next methodology directly integrates (10) for square elements thus eliminating the need to use an approximation. The closed form of (10) for square elements is

$$[A]_{mn} = \frac{1}{4\pi\epsilon_0}\big((x_m - x')\ln((y_m - y') + R) + (y_m - y')\ln((x_m - x') + R)\big) \quad (13)$$

where $R = \sqrt{(x_m - x')^2 + (y_m - y')^2}$ evaluated over $x' = x_n \pm \Delta x/2$, $y' = y_n \pm \Delta y/2$ [2]. This method will be referred to as the exact square element method.

For the third and final method studied here, the pulse function is used as a testing function as opposed to the Dirac delta function, (9), used in (10). This is physically equivalent to averaging the testing function over the entire element as opposed to measuring only at the element center point as in the first two methods [1], [2]. This methodology gives rise to the following system matrix

$$[A]_{mn} = \iint_{S_m} \iint_{S_n} \epsilon_0^{-1} g(\mathbf{r},\mathbf{r'}) dS' dS, \quad (14)$$

and boundary condition array

$$\{b\}_m = S_m \Phi_0. \quad (15)$$

Under this formulation, (14) can be exactly solved resulting in

$$[A]_{mn} = \frac{1}{4\pi\epsilon}\Big(\frac{(x - x')^2(y - y')}{2}\ln((y - y') + R) + \frac{(x - x')(y - y')^2}{2}\ln((x - x') + R) - \frac{(x - x')(y - y')}{4}((x - x') + (y - y')) - \frac{R^3}{6}\Big) \quad (16)$$

where $R = \sqrt{(x - x')^2 + (y - y')^2}$ evaluated over $x' = x_n \pm \Delta x/2$, $y' = y_n \pm \Delta y/2 x = x_m \pm \Delta x/2$, $y = y_m \pm \Delta y/2$ [2]. This method is referred to as subdomain collocation. Unlike the previous set of equations, this method results in symmetric matrices thereby allowing a halving of the total memory footprint (assuming appropriate data structure and solver choice) as only the upper diagonal matrix needs to be stored. Despite this, solving these systems is still asymptotically $O(n^3)$ in the worst case.

One major issue with this formulation is the fact that indeterminate terms ($\ln(0) \times 0$) arise when $x = x'$ or $y = y'$. A trivial means of resolving these indeterminate forms is to add a very small constant c to each $\ln(x + c)$ call such that c is much less than x for any value of x used in the assembly process. Firstly, this prevents indeterminate forms from arising as the natural log of a small number multiplied by zero evaluates to zero. Secondly, provided that $c << x$, the error introduced by including c can be reduced to machine

precision for sufficiently small c. Since the introduced error can be made arbitrarily small, there is no need to conditionally check for problematic arguments to $\ln()$ calls as this c term can always be included. This increases assembly efficiency as the matrix assembler does not need to conditionally check for problematic arguments 16 times for all n^2 elements in the system matrix. For the purposes of this work, $c = 1 \times 10^{-100}$ was chosen as it satisfies $c << x$.

III. Numerical Results

With the mathematical model now fully established, the implementation of this model is documented in Section III-A. From here Sections III-B verifies the model against approximate plate capacitances modeled using a random walk approach. Finally, Section III-C investigates the convergence of all three formulations to a single numerical value as a function of the asymptotic number of operations required to obtain these capacitance values.

A. Implementation

For the purposes of this study, only a square plate of side length $l = 0.01$m is considered. This plate is broken up into an $n \times n$ grid of square surface elements as this is what all formulations of the system matrix are derived for as found in Section II-2. In Secton III-B, the value of n will range from 10-30 for verification and validation against plate capacitance values found in the literature. These values will be expanded in Section III-C to range between 10-100 to model convergence to a single numerical value.

All code for this model was written in Python for its ease of use and plentiful numeric packages such as NumPy and SciPy, both of which were used. The model is constructed in a way to handle system matrix assembly for any number of elements per side and side length for square plates thus making it flexible for this analysis.

B. Verification and Validation

Prior to performing any novel analysis, the model first needs to be validated against theory. In 2004, H. J. Wintle used Monte Carlo methods to place a bound on dimensionless capacitance of square plates [3]. Their models resulted in a dimensionless capacitance $C^* = 0.36 \pm 0.01$ for square plates defined as

$$C^* = \frac{C}{4\pi\epsilon_0 l} \tag{17}$$

where C is the true capacitance value and l is the side length of the plate in meters [3]. Using a side length of $l = 0.01$m, this bound maps to the capacitance range of 0.389-0.411pF. Thus if for some number of elements per side length, n, the model is able to predict values within this range, they are technically valid.

To predict the capacitance of the square plate we, first integrate the charge distribution over all elements which gives the total charge. From basic electrostatics, the capacitance is then determined by dividing the total charge by the arbitrary choice of potential. This mapping and corresponding reduction over all M elements is written succinctly as

$$C = \frac{S}{\Phi_0} \sum_{m=0}^{M-1} \rho_s^{(m)} \tag{18}$$

where S is the surface area of a single square element, and $\rho_s^{(m)}$ is the surface charge density associated with the m^{th} element.

Applying this map-reduce operation to the surface charge distributions obtained from solving (5) results in the following capacitance values as found in Table 1.

Table 1: Predicted Capacitances by Method

Method	10×10	20×20	30×30
Circular Element Approximation	0.397pF	0.403pF	0.405pF
Exact Square Elements	0.394pF	0.401pF	0.403pF
Subdomain Collocation	0.399pF	0.403pF	0.405pF

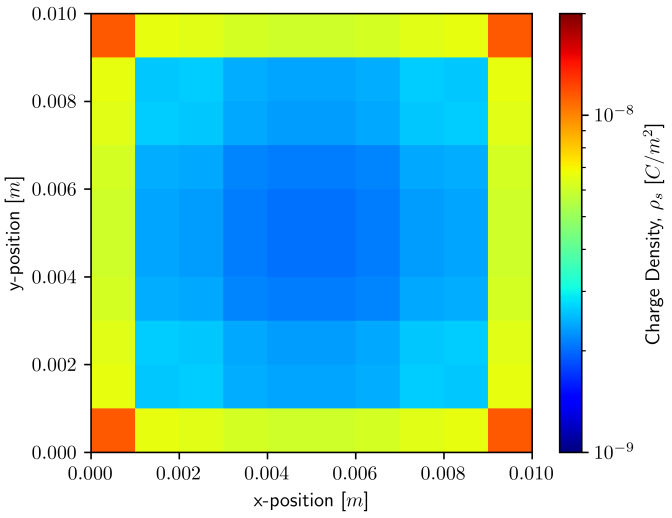
As seen in Table 1, the capacitances of all system matrix formulations for all grid resolutions technically converge to the Monte Carlo solution reported in [3]. In general, the circular element approximation is generally very similar to the subdomain collocation method while the exact square element solution trails slightly behind these methods. Based on this it is assumed that the circular element approximation over estimates the charge density slightly which results in it behaving similarly to subdomain collocation using an entirely different testing function. Further discussion on convergence will be included in Section III-C.

With the capacitance models validated for all mesh resolutions and system matrix formulations, the charge distributions are now analyzed to ensure correctness of the overall distribution and not just total charge. With this, surface charge distributions for 10×10, 20×20, and 30×30 grids using subdomain collocation are found in Fig. 1.

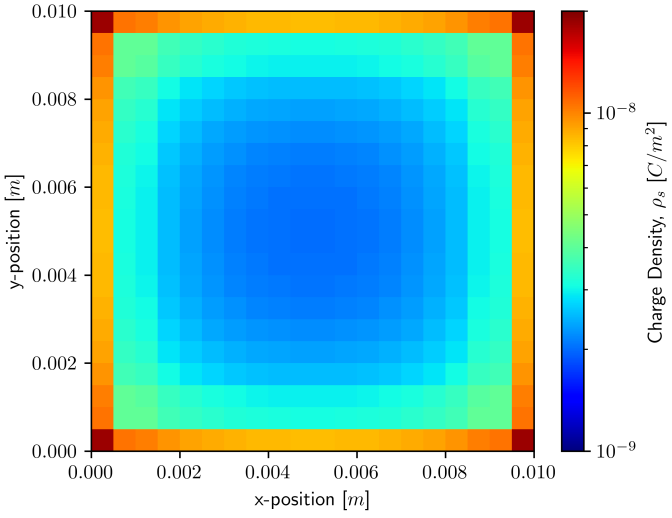
As shown in Fig. 1, the charge distribution for all grid resolutions is focussed on the edges of the plate and is most intense in the corners. This can be understood from basic electrostatic theory as like charges repel. Thus, charges will most likely accumulate in locations with fewer "neighboring charges", namely the corners and edges of the plate. Also seen in Fig. 1, the final "form" of the charge distribution is reached by the 20×20 grid. This metric is rather subjective and is based on a pseudocolor map thus will not be expanded further. In addition to this, the choice of using subdomain collocation is entirely arbitrary. This formulation was chosen as it was deemed the most accurate due to its use of a first order pulse testing function. Distributions for the circular element approximation and exact square element solution looked nearly identical and thus were not included.

C. Convergence Study

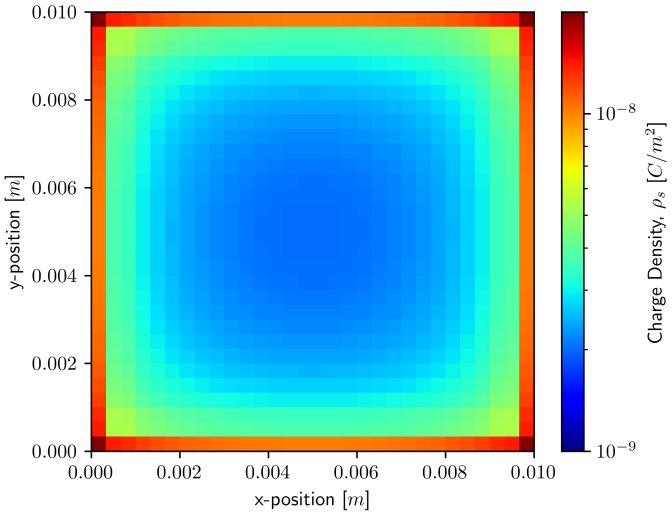
With the model successfully validated, it is now important to compare the convergence rates between these methodologies.



(a)



(b)



(c)

Fig. 1: Surface Charge Density (ρ_s) distributions using Subdomain Collocation for mesh resolutions of (a) 10×10 (b) 20×20 and (c) 30×30.

As discussed in Section III-B, the predicted capacitances of all three formulations and grid resolutions technically converged to capacitance ranges reported in the literature; however, these methods did not necessarily converge to a single value. This begs the following question: what numerical value, if any, do these methods converge to? To this end, a convergence study was performed extending the maximum resolution to a 100×100, sweeping over intermediate resolutions. This maximum resolution was chosen as it was the largest matrix the model, as described, could resolve in a 'reasonable' amount of time. Limitations of the model will be discussed further in Section IV. Results of this sweep can be found in Fig 2 as a function of the asymptotic number of operations required to obtain said solutions which is $O(n^3)$ for dense matrices.

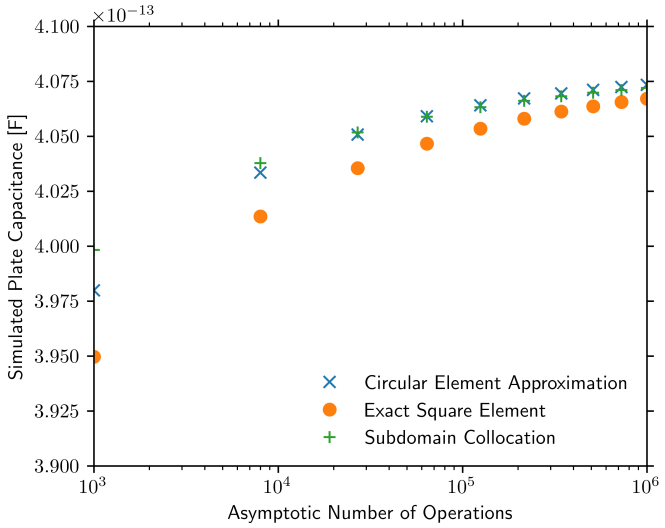


Fig. 2: Convergence Study of All Three System Matrix Formulations Over a Wide Range of Grid Resolutions

As seen in Fig. 2, as the grid resolution, and corresponding number of asymptotic operations increases, all three formulations converge on a capacitance of approximately 0.4075pF. A simple percent difference analysis reveals that these methods produce the same value within 0.1% of each other for $n \geq 30$. This low percent difference results in the circular approximation of square surface elements being the ideal choice for this problem for its overall simplicity and rapid assembly process for matrices on the order of those studied here. This fact would change however if the matrices were far larger. For example, there exist scalable and memory efficient algorithms for solving dense symmetric matrices as those found in subdomain collocation far quicker than dense unsymmetrical matrices found in both of the methodologies using the delta function as a testing function [4]. Thus for solving larger, more complex systems, the subdomain collocation method becomes the optimal choice.

IV. Conclusion

A 2-dimensional method of moments program was developed from Maxwell's Equations allowing for the charge distribution to be calculated on an arbitrarily sized square

metallic plate for an arbitrary number of elements per side. The charge distribution was then integrated to find the total charge collected on the metallic plate. Finally, the total charge was then used to solve for the capacitance of the plate given an arbitrary applied equipotential. These predicted capacitances were next verified against those obtained from Monte Carlo methods found in the literature of which they agreed well. The charge distribution as a function of space was then analyzed and explained with basic electrostatic theory. Finally, a convergence study was performed and showed that all methodologies tested converge to a predicted capacitance of approximately 0.4075pF and all predict values within 0.1% of eachother for $n \geq 30$.

While relatively general in the sense that the model works for any side length, applied potential, and number of elements per side, this model is not performant. Upon software profiling, it was discovered that the code spends the majority of its run time assembling the matrices. This is most notable in the exact square element, and subdomain collocation procedures where each element of the system matrix takes between $4-16$ iterations to fully assemble. When combined with the Python's abysmal loop performance, the matrix assembly dominates the program execution time. To that end, future work should first focus rewriting the model in a language like C/C++/Rust, all of which have optimizing compilers that can take advantage of loop unrolling to dramatically speed up the assembly process. After which, the next logical step would be to expand support to handle unstructured meshes such that the capacitance of arbitrary shaped surfaces could be obtained.

REFERENCES

[1] T. E. Roth, *ECE 61800 Lecture Notes*. Purdue University, 2024.

[2] J.-M. Jin, *Theory and Computation of Electromagnetic Fields*. John Wiley & Sons, 2011.

[3] H. Wintle, "The capacitance of the cube and square plate by random walk methods," *Journal of Electrostatics*, vol. 62, no. 51-62, 2004.

[4] R. A. V. D. G. PAOLO BIENTINESI, INDERJIT S. DHILLON, "A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations," *SIAM J. Sci. Comput.*, vol. 27, no. 1, 2005.