2007

Electrical Capacitance Volume Tomography Of High Contrast Dielectrics Using A Cuboid Geometry

Mark Nurge
University of Central Florida

Electrical Capacitance Volume Tomography of High Contrast Dielectrics Using a Cuboid Geometry

by

Mark A. Nurge
B. Electrical Engineering Georgia Institute of Technology, 1985
M.S. Electrical Engineering Georgia Institute of Technology, 1986
M.S. Engineering Management University of Central Florida, 1992
M.S. Physics University of Central Florida, 2003

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Physics
in the College of Sciences
at the University of Central Florida
Orlando, Florida

Spring Term
2007

Major Professor: Patrick Schelling

Abstract

An Electrical Capacitance Volume Tomography system has been created for use with a new image reconstruction algorithm capable of imaging high contrast dielectric distributions. The electrode geometry consists of two 4×4 parallel planes of copper conductors connected through custom built switch electronics to a commercially available capacitance to digital converter.

Typical electrical capacitance tomography (ECT) systems rely solely on mutual capacitance readings to reconstruct images of dielectric distributions. This dissertation presents a method of reconstructing images of high contrast dielectric materials using only the self capacitance measurements. By constraining the unknown dielectric material to one of two values, the inverse problem is no longer ill-determined. Resolution becomes limited only by the accuracy and resolution of the measurement circuitry. Images were reconstructed using this method with both synthetic and real data acquired using an aluminum structure inserted at different positions within the sensing region. Comparisons with standard two dimensional ECT systems highlight the capabilities and limitations of the electronics and reconstruction algorithm.

To my wife, Becky, for her loving support and encouragement to pursue my dream of becoming a physicist

and

To my sons, Ryan and Kyle whose love, joy, and wonder continually serve as a reminder of all that is good and important in this life.

Acknowledgments

This work would not have been possible without the help of the following people:

-Dr. Robert Youngquist for his multitude of contributions, daily mentoring, friendship, and inspiration. He is a role model for all that a NASA scientist should and can become.

-Dr. Robert E. Peale, and Dr. Patrick Schelling for their guidance, encouragement, and insightful comments regarding this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
BACKGROUND

Over the past few years, Kennedy Space Center's Applied Physics Laboratory has developed capacitance based sensing solutions for for a number of aerospace applications. Some of these applications include, measuring mass of cryogenic fluids, level sensing in cryogenic fluids, and moisture detection in composite materials and Shuttle Thermal Protection System (TPS) tiles. [1] [2] [3]. The aim of the present work was to extend skills in capacitive sensing by developing an Electrical Capacitance Volume Tomography (ECVT) system and associated image reconstruction algorithms. A summary of potential target applications is provided in the next few paragraphs.

The Shuttle Program has shown interest in pursuing capacitive imaging for use as a non-destructive evaluation tool. Many non-conductive materials used in spacecraft are sensitive to accumulation of moisture and must be treated to minimize water build up prior to launch. Otherwise, the pressure decrease and temperature increase during vehicle ascent could lead to rapid vaporization of the water, causing damage to the material. An example is the Shuttle TPS tiles. These silica fiber tiles are injected with a substance to make them hydrophobic prior to liftoff. However, the substance breaks down due to the heating on re-entry making them hydrophilic. Any absorbed moisture then must be removed prior to subsequent missions. The need has been identified for a tool to verify that the moisture is properly removed from

the tiles. A capacitance based tool, closely related to a household stud sensor, has been developed to sense the presence of water in the Shuttle's TPS tiles. The sensor can provide the approximate location of the moisture along the surface of the tile, but not the quantity or depth. ECVT could meet this need by providing an image of the three dimensional dielectric distribution for the section of tile under inspection.

The liquid levels of fuel and oxidizer in tanks on orbit are measured by accelerating the spacecraft to get the liquid to all move to one side of the tank. A traditional level sensor is then used to learn how much fluid is left in the tank. This is inefficient since the process consumes precious commodity to get the measurement. An ECVT system could potentially be used to quantify the amount of commodity remaining, without having to fire the spacecraft engines.

Another important on orbit application is moisture sensing in plant growth experiments. In terrestrial based plant growth experiments, many methods exist to determine the soil moisture content. However, on orbit, the micro-gravity environment causes the water in soil to distribute itself in unusual ways. ECVT could be used as part of a nutrient delivery system to minimize waste water and ensure healthy plant growth.

Personnel associated with the University of Manchester Institute of Science and Technology (UMIST) in England have pioneered much of the work leading to industrial use of ECT [4]. ECT has been used since the late 1980s within the petrochemical industry to reconstruct the dielectric profile of material located in cross sections

of non-conductive pipes [5]. One specific use of ECT is to study the dynamics of multi-phase fluid flow through pipes. The sensing elements typically consist of 8-12 electrodes and an overall guard ring surrounding the circumference of the pipe. Mutual capacitance measurements are made on the different pairs of conductors in the system. Image reconstruction is based on $m = n(n-1)/2$ measurements, where n is the number of electrodes. This capacitance data is then inverted through a variety of methods to reproduce a two dimensional image corresponding to the dielectric distribution of the material in the field of view.

Fan [6] provides a recent overview of work that has been done in 3D ECT (also called ECVT) with cylindrical geometries. In some cases, the standard two dimensional sensor is used to capture capacitances at different times as fluid flows through a pipe containing the electrodes. The image is then reconstructed from these 2D snapshots. Others, such as Fan, are using multiple bands of electrodes around the pipe in a true 3D geometry.

A survey of the current state of the art in image processing for ECT is provided by Yang [7]. Many reconstruction techniques rely on using a sensitivity matrix that linearly relates the discrete dielectric distribution to the mutual capacitances. This can be done with low contrast dielectric distributions, but is ill-suited to high-contrast distributions such as water in air, foam, or glass fiber. The reconstruction algorithms are formulated as ill-posed problems due to the desire to construct high resolution images from a limited number of independent mutual capacitance measurements.

The work described here differs from this traditional approach to ECT in several aspects. The described ECVT system uses a cuboid geometry with two parallel planes of 16 conductors arranged in a 4×4 pattern. A new 3D image reconstruction algorithm was created that addresses the aforementioned limitations. It will be shown that it is possible to generate images of high contrast dielectric materials with only the self-capacitance data. An aluminum structure in an air background is used to represent the high contrast dielectric material. The technique is demonstrated with both simulated and real data taken with the instrumentation specially designed and constructed for this work.

CHAPTER 2
IMAGE RECONSTRUCTION

2.1 Review of capacitance

In general, the charges, q_i, on a system of conductors are related to the potentials applied to those conductors, V_j, by the elements of the capacitance matrix, C_{ij}, via the well known equation

$$q_i = \sum_{j=1}^{n} C_{ij} V_j. \tag{2.1}$$

Self capacitance is defined [8] as the total charge on a conductor when it is maintained at a 1 volt potential, while all other conductors are held at ground. These self capacitances are carried in the matrix along the diagonal, corresponding to the elements C_{ii}. The off diagonal elements, $C_{ij}, i \neq j$, are called the coefficients of induction or more commonly, mutual capacitances. Physically, these mutual capacitances correspond to the charge needed to maintain conductor j at ground, while conductor i is maintained at 1 volt potential and all others at ground. Some properties of the matrix are [9]:

- The self capacitances are positive.
- The mutual capacitances are negative.
- The matrix is symmetric with $C_{ij} = C_{ji}$.

- For a closed system containing all field lines, $\sum_{j=1}^{n} C_{ij} = \sum_{i=1}^{n} C_{ij} = 0$ and $C_{ii} = -\sum_{j,j\neq i} C_{ij} = -\sum_{j,j\neq i} C_{ji}$, due to the conservation of charge.

In practice, a closed system is difficult to achieve because some field lines will end on a grounded shield or go to infinity [10]. If these field lines run through the imaging region, they contain information helpful to image reconstruction. Since measurement of the mutual capacitance with respect to the shield is not usually performed, the only source of this additional position information is in the self capacitance readings. They are also larger and and require less circuitry than the mutual capacitances. Therefore, this dissertation will show how this measurement system and reconstruction techniques can be used with self capacitances alone.

2.2 Capacitance relation to the dielectric distribution

To relate the capacitance matrix to the dielectric distribution, start with Gauss's Law, $\nabla \cdot \boldsymbol{D} = \rho_f$, where $\boldsymbol{D}$ is the electric displacement field and ρ_f is the volumetric free charge density. A small Gaussian box can then be used at the surface of a conductor to derive the well known boundary condition $\boldsymbol{D} \cdot \boldsymbol{n} = \sigma$, where $\boldsymbol{n}$ is a unit vector normal to the surface of the conductor and σ is the surface charge density on the conductor. In terms of the electrostatic potential, Φ, and the dielectric distribution, ϵ, this becomes

$$-\epsilon \frac{\partial \Phi}{\partial n} = \sigma. \tag{2.2}$$

Integrating this equation over the entire surface of a conductor gives a form that can be related to capacitance,

$$-\int_S \epsilon \frac{\partial \Phi}{\partial n} dS = q. \tag{2.3}$$

So, for a given conductor, i,

$$C_{ij} = -(\int_S \epsilon \frac{\partial \Phi}{\partial n} dS)_i / V_j, \tag{2.4}$$

where conductor j is raised to some potential V with all others conductors grounded. The capacitance both with and without dielectric material present can be subtracted giving:

$$\Delta C_{ij} = (\epsilon_0 \int_S (\frac{\partial \Phi_{\epsilon_0}}{\partial n} - \epsilon_r \frac{\partial \Phi_{\epsilon_r}}{\partial n}) dS)_i / V_j \tag{2.5}$$

ΔC_{ij} represents the change in capacitance solely due to addition of the dielectric material, and Φ_{ϵ_r} and Φ_{ϵ_0} represent the electrostatic potential with and without the presence of the dielectric material, respectively. As is standard convention, ϵ_0 represents the permittivity of free space and ϵ_r is the relative dielectric of the sample material. In Equation 2.5, ϵ_r is the relative dielectric located in the space immediately along the surface of the conductor, whereas Φ_{ϵ_r} is dependent on the distribution of dielectric material throughout space.

Given a dielectric distribution and the Dirichlet boundary conditions, the differential form of Gauss's Law can be readily solved to find the electrostatic potential everywhere. One set of boundary conditions is needed for each conductor to find

the potentials needed to fully populate the capacitance matrix. Each of the change in capacitances can then be calculated with Equation 2.5. However, the problem at hand is to find the dielectric distribution given the capacitances on the left side of the equation. This is a so called inverse problem. The problem is further complicated since the electrostatic potential is dependent on the distribution of dielectric everywhere in space between the conductors.

An approach for solving this inverse problem involves guessing a dielectric distribution, solving the forward problem, and then comparing it to the measured values. A new guess is formulated based on this comparison and the process is repeated until the difference between computed and measured capacitances reaches some minimum value. To do this numerically, the differential equation must first be stated in a discrete form.

2.3 Discussion of methods explored and their limitations

Before continuing, it makes sense to look at a few paths that were explored that led to implementation with the finite volume method. The following few paragraphs will discuss the logical progression of ideas and their limitations in this application.

2.3.1 Half-space problem

Prior to settling on the geometry of two sets of parallel arrays of conductors, a single plane of conductors was considered with a hope of implementing a system capable of mapping out moisture in Shuttle heat protection tiles. This geometry can be modeled as a half space problem. So, the first attempt at solving the differential equation was to look for a closed form solution for the electrostatic potential, $\Phi(x, y, z)$, for a rectangular conductor raised to some potential, V, while surrounded by an infinite ground plane. With the abundant availability of numeric algorithms and differential equation solvers, it is easy to miss the existence of a closed form solution for a particular problem. Such is the case with this specific problem, as the existence of the closed form solution is not obvious. A generic solution for the half space problem where $\nabla^2 u(x, y, z) = 0$ and $u(x, y, 0) = f(x, y)$ is presented in Sneddon [11] and Riley et al. [12] as

$$u(x_0, y_0, z_0) = \frac{z_0}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{f(x, y)}{[(x - x_0)^2 + (y - y_0)^2 + z_0^2]^{3/2}} dxdy. \tag{2.6}$$

For a rectangular conductor raised to potential, V, centered at the origin in the xy plane, the expression for $\Phi(x, y, z)$ can be written as

$$\Phi(x, y, z) = \frac{z}{2\pi} \int_{-b/2}^{b/2} \int_{-a/2}^{a/2} \frac{V}{[(x' - x)^2 + (y' - y)^2 + z^2]^{3/2}} dx'dy', \tag{2.7}$$

where the conductor has sides of length a and b.

The first integral evaluates to

$$\Phi(x,y,z) = \frac{z}{2\pi}\int_{-b/2}^{b/2}\Big(\frac{V(a/2-x)}{((y'-y)^2+z^2)\sqrt{(a/2-x)^2+(y'-y)^2+z^2}} - \frac{V(-a/2-x)}{((y'-y)^2+z^2)\sqrt{(-a/2-x)^2+(y'-y)^2+z^2}}\Big)dy'. \tag{2.8}$$

The second integral is a little more difficult to solve. One method is to notice that the integral is of form $\int \frac{Ax+B}{(p+R)\sqrt{R}}dx$, where $R = a + bx + cx^2$. The solution to this form is found in equation 2.284 of Gradshteyn and Ryzhik [13]. Alternately, Mathematica version 5 will evaluate the integral symbolically to give

$$\begin{aligned}\Phi(x,y,z) = \frac{V}{2\pi}\Big(&\tan^{-1}\frac{(a-2x)(b-2y)}{2z\sqrt{(a-2x)^2+(b-2y)^2+4z^2}} \\ &+\tan^{-1}\frac{(a+2x)(b-2y)}{2z\sqrt{(a+2x)^2+(b-2y)^2+4z^2}} \\ &+\tan^{-1}\frac{(a-2x)(b+2y)}{2z\sqrt{(a-2x)^2+(b+2y)^2+4z^2}} \\ &+\tan^{-1}\frac{(a+2x)(b+2y)}{2z\sqrt{(a+2x)^2+(b+2y)^2+4z^2}}\Big)\end{aligned} \tag{2.9}$$

Two contour plots are shown in Figure 2.1 for the x=0 plane and the y=0 plane, respectively. The size of the conductor is $a = 0.01$ meter and $b = 0.005$ meter. The units on the axes are meters and $V = 10$ volts.

It is possible to approximate the total charge and capacitance for this electrode by applying Equations 2.3 and 2.4, respectively. However, a gap must be inserted

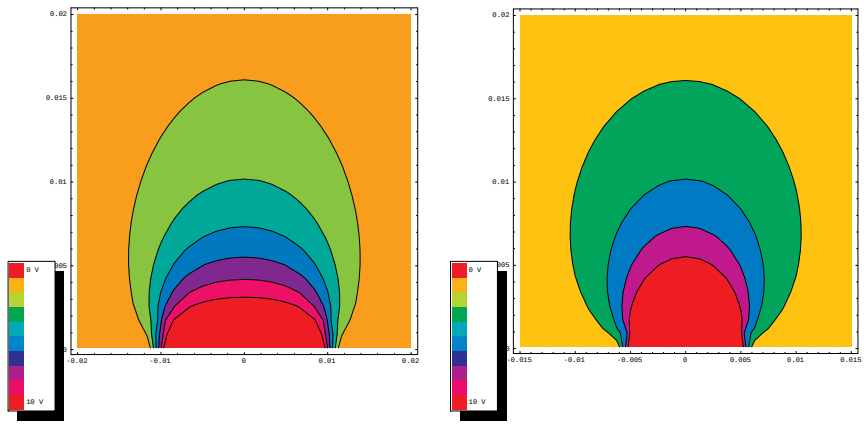


Figure 2.1: Electrostatic potential for a rectangular conductor in the z=0 plane raised to V=10 volts. The plots are for the x=0 and y=0 planes, respectively.

between the rectangular plate and the ground plane to keep the charge density from becoming infinite at the edges. This provides a description of the electrostatic behavior when the dielectric distribution is the same everywhere in space. When the dielectric varies, the differential equation becomes

$$\nabla^2\Phi = -\nabla\Phi \cdot \nabla\epsilon_r/\epsilon_r. \tag{2.10}$$

This equation only appears to have a closed form solution for the case of $\nabla\epsilon_r = 0$. Therefore, the analytical form has limited usefulness in reconstruction of a 3D ECT image. Additionally, it can be seen from Figure 2.1 that the electric field falls off fast since it goes as the gradient of the potential. This means that the capacitances of a single plane of electrodes rapidly become insensitive to changes in dielectric as

the material is positioned further away from the electrode. After noting this, it was decided to use a second parallel array of electrodes surrounded by a ground plane as the basis for the 3D ECT system.

2.3.2 Finite Difference Method (FDM)

A standard way of modeling a differential equation and solving it on a computer is to begin by creating a grid of discrete points to represent the geometry of the physical problem. A series of finite difference equations is then used to relate adjacent points to one another according to the differential equation. For this electrostatic problem, the finite differences are in terms of the electrostatic potentials and the relative dielectric constant of the associated dielectric material. For a cartesian coordinate system with x, y, z directions, Equation 2.10 becomes

$$\frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = -\frac{1}{\epsilon_r}(\frac{\partial\Phi}{\partial x}\frac{\partial\epsilon_r}{\partial x} + \frac{\partial\Phi}{\partial y}\frac{\partial\epsilon_r}{\partial y} + \frac{\partial\Phi}{\partial z}\frac{\partial\epsilon_r}{\partial z}). \tag{2.11}$$

For the region of space between conductors, it's easy to see that Equation 2.11 reduces to Laplace's equation everywhere in space but along the boundary between material with different dielectric constants. The term on the right hand side acts as a source term due to bound charge that appears when dielectric material is polarized. Traditionally, source terms are represented by discontinuous functions. This can

present a problem when attempting a discrete formulation for Equation 2.11, as will be shown next.

The finite difference representation is derived from Taylor series expansions in Kythe [14]. In discrete form,

$$\frac{\partial \Phi}{\partial x}|_{i,j,k} = \frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{2h_x}, \tag{2.12}$$

where, i, j, k are the coordinates of the discrete point in x, y, z directions, respectively and h_x is the spacing between adjacent grid-points in the x-direction. Likewise, the second derivative can be represented by

$$\frac{\partial^2 \Phi}{\partial x^2} = \frac{\Phi_{i+1,j,k} - 2\Phi_{i,j,k} + \Phi_{i-1,j,k}}{h_x^2}. \tag{2.13}$$

The finite difference form for $\partial \epsilon_r / \partial x$ is similar to Equation 2.12, but with ϵ_r replacing Φ. For grid points in space where the dielectric of adjacent grid points only varies in the x-direction, the discrete differential equation becomes

$$\begin{aligned} &\frac{\Phi_{i+1,j,k} - 2\Phi_{i,j,k} + \Phi_{i-1,j,k}}{h_x^2} + \frac{\Phi_{i,j+1,k} - 2\Phi_{i,j,k} + \Phi_{i1,j-1,k}}{h_y^2} \\ &+ \frac{\Phi_{i,j,k+1} - 2\Phi_{i,j,k} + \Phi_{i1,j,k-1}}{h_z^2} = -\frac{\Phi_{i+1,j,k} - \Phi_{i-1,j,k}}{4h_x^2} \frac{\epsilon_{i+1,j,k} - \epsilon_{i-1,j,k}}{\epsilon_{i,j,k}}. \end{aligned} \tag{2.14}$$

The computation of the right hand side is dependent on the dielectric values for three consecutive grid points. So, for a transition from water to air, the fraction containing

ϵ can take on values of ± 79, $\pm 79/80$, or even 0, if the dielectric material is only one voxcell wide. This has the effect of inducing numeric instability when the equation is applied with a relaxation technique over the grid. An example of this is shown in Figure 2.2, which plots the electrostatic potential for both the FDM and theoretical results for an ideal parallel plate capacitor, partially filled with dielectric material ($\epsilon_r = 1000$). The FDM is stable when the contrast between the different dielectric materials is small. The comparison between electostatic potential calculations is shown for $\epsilon_r = 3$ in Figure 2.3.

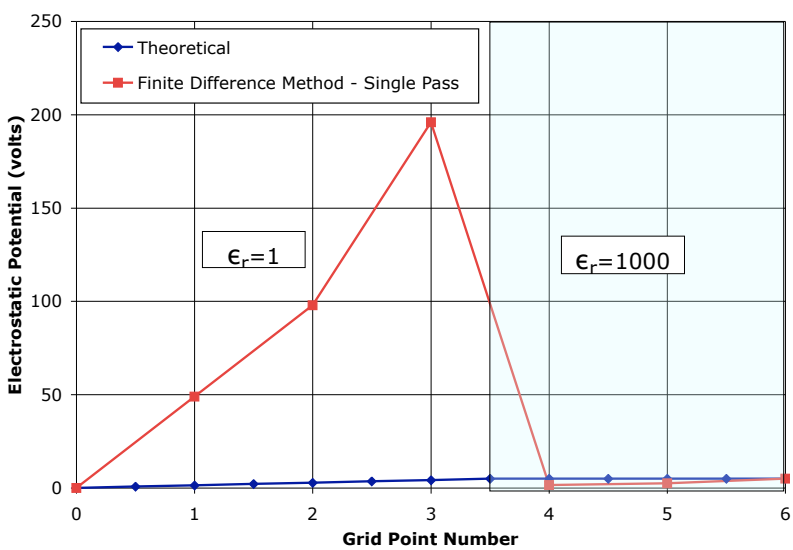


Figure 2.2: Plot of electrostatic potential for an idealized parallel plate capacitor partially filled with dielectric material ($\epsilon_r = 1000$) and 5 volts applied across the conductors. The potential was computed with the FDM and the standard theoretical model for this type of problem. Only a single relaxation pass was performed with FDM due to the numerical instability.

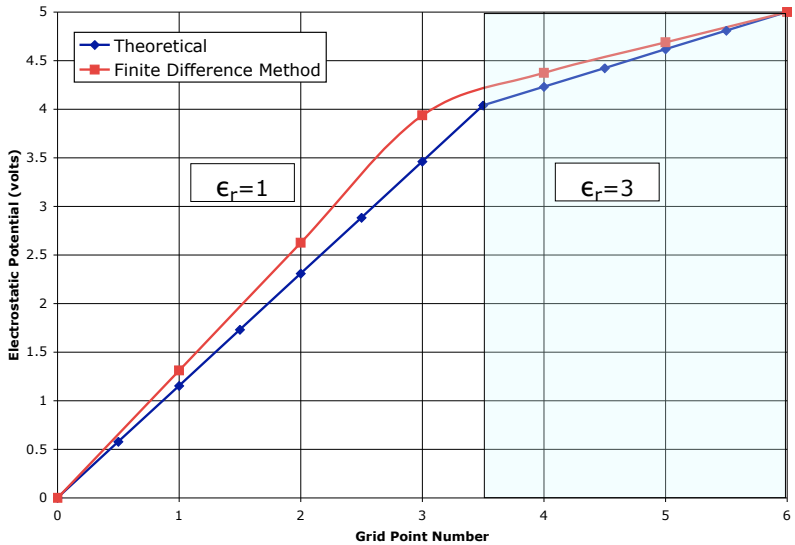


Figure 2.3: Plot of electrostatic potential for an idealized parallel plate capacitor partially filled with dielectric material ($\epsilon_r = 3$) and 5 volts applied across the conductors. The potential was computed with the FDM and the standard theoretical model for this type of problem.

Looking back at Equation 2.12, it is easy to see that the FDM lends itself best to modeling continuous functions. The discontinuity experienced at the edge of the dielectric material is poorly approximated by a ramp across several adjacent points and results in the numeric instability when implemented on a grid by relaxation. After searching for a fix for this problem, the FDM was abandoned for another method.

2.4 Discrete formulation for the forward problem using the Finite Volume Method

2.4.1 Governing math

For the region between electrodes (no free charges), Gauss's Law in terms of the electrostatic potential is equation 2.10. The term on the right hand side of the equation is the source term for the bound charges at the surface of the dielectric material. This creates a discontinuity at the edge of the material that can cause a problem because finite difference equations assume gradual changes in the function between adjacent points. The Finite Volume Method (FVM) [15], used widely in computational fluid dynamics, offers a way to handle this problem. FVM avoids the discontinuities by locating the potentials at the center of the voxels rather than at the vertices, as done by finite difference methods. The following mathematics develop the FVM equations for the voxels in terms of the electrostatic potential and the dielectric distribution.

FVM applies a discrete form of Gauss's Law in integral form to multiple adjoining volumes of space within the region of interest. Since there is no free charge between conductors, this takes the form of a surface integral due to the divergence theorem

$$\int_V \nabla \cdot \boldsymbol{D} dV = \int_S (\boldsymbol{D} \cdot \hat{n}) dS = 0. \tag{2.15}$$

Since the ECVT system has a cuboid geometry, a cartesian coordinate system will be employed. For two adjacent voxels, shown in Figure 2.4, $\boldsymbol{D}$ is continuous across the boundary, meaning $\epsilon_1 \boldsymbol{E}_1 = \epsilon_2 \boldsymbol{E}_2$. Since $\boldsymbol{E} = -\nabla\Phi$, the discrete form of the magnitudes becomes $2\epsilon_1(\Phi_1 - \Phi_2)/h_y = 2\epsilon_2(\Phi_2 - \Phi_3)/h_y$, where h_y is the distance between the centers of the voxels along the y direction. Solving for Φ_2 yields,

$$\Phi_2 = \frac{\epsilon_1\Phi_1 + \epsilon_2\Phi_3}{\epsilon_1 + \epsilon_2}. \tag{2.16}$$

Substituting Equation 2.16 into $D_y = 2\epsilon_1(\Phi_1 - \Phi_2)/h_y$ and simplifying results in

$$D_y = \frac{2\epsilon_1\epsilon_2}{\epsilon_1 + \epsilon_2}\frac{\Phi_1 - \Phi_3}{h_y}. \tag{2.17}$$

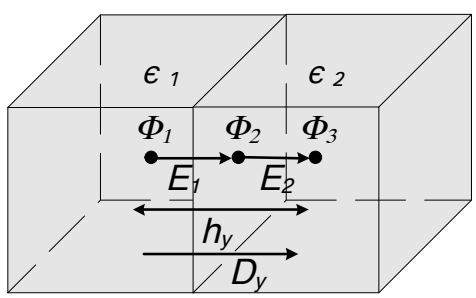


Figure 2.4: Two adjacent voxels showing the electric field lines across one interface. Each voxel is of uniform dielectric material. Electrostatic potentials are shown at the centers of each voxel and at the interface.

Enough information is now available to render equation 2.15 in discrete form. Using figure 2.5 as a reference, it becomes

$$(D_{x_{(i,j,k)}} - D_{x_{(i-1,j,k)}})h_y h_z + (D_{y_{(i,j,k)}} - D_{y_{(i,j-1,k)}})h_x h_z + \\ (D_{z_{(i,j,k)}} - D_{z_{(i,j,k-1)}})h_x h_y = 0, \tag{2.18}$$

where, h_x, h_y, and h_z represent the distance between voxel centers in the x, y, and z directions and i, j, k are the associated coordinates of each voxel center. In terms of the dielectric constant, ϵ, and electrostatic potential, Φ, the y component will have the form

$$D_{y_{(i,j,k)}} = \frac{2\epsilon_{i,j,k}\epsilon_{i,j+1,k}}{\epsilon_{i,j,k} + \epsilon_{i,j+1,k}}(\frac{\Phi_{i,j+1,k} - \Phi_{i,j,k}}{h_y}), \tag{2.19}$$

and the other two components are similar. Equations 2.18 and 2.19 can then be combined for each voxel to achieve a description of the system in terms of the dielectric distribution and the electrostatic potential. The same mathematical relationships can be arrived at by using a circuit analog of two capacitors in series between the two voxel centers, giving some additional insight into the physical behavior. This is the topic of the next subsection.

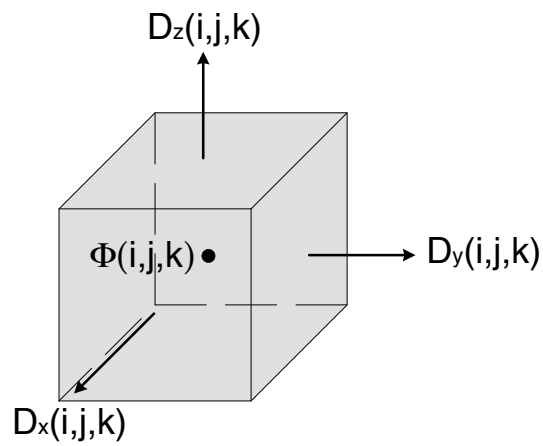


Figure 2.5: Finite Volume Method applied to electrostatics. $\boldsymbol{D}$ components are normal to each cell face, while $\Phi(i, j, k)$ is located in the center of the volume. The entire cell is assumed to be of the same permittivity $\epsilon(i, j, k)$

2.4.2 Circuit analogue

Relaxation techniques have been applied to find numerical solutions to differential equations for almost 70 years. Some of the earliest work was done by Southwell [16]. However, until the advent of the computer, these techniques were tedious to apply due to multiple iterations and large number of grid points needed to accurately model a problem. It has also long been recognized that many differential equations describing different phenomena, have the same form. Since electrical circuits are easy to construct, they have been used as analogues to model different phenomena described by similar differential equations. Several good papers on the subject are presented by Liebmann [17] [18].

One method Liebmann describes is using a lattice of resistors to simulate a parallel plate capacitor. When voltages are applied to the boundary resistors, a voltmeter can be used to measure the potentials at different locations between the 'electrodes', giving an accurate representation of the electrostatic potential, without the need for a computer. In this case, the potential within a parallel plate capacitor is easily calculated, but the problem illustrates the method's ease of use and accuracy. The method need not be limited to resistor networks; other electrical components may be used to better model the physical behavior. For the case of an electrostatics problem, capacitance elements seem like a better physical analogue. So, for the ECT problem, a 3D network of capacitors was used to model the space between conductors. However, instead of constructing a physical model from discrete components, the goal was to take advantage of circuit theory to simplify the problem.

Figure 2.6 shows the electrical circuit analogue to Figure 2.4. Assuming the area of each face is A, then $C_1 = 2\epsilon_1 A/h_y$ and $C_2 = 2\epsilon_2 A/h_y$. The combined series capacitance is

$$C_{1-2} = \frac{A}{h_y}\frac{2\epsilon_1\epsilon_2}{\epsilon_1 + \epsilon_2}. \tag{2.20}$$

When a sinusoidal voltage of frequency ω is applied between V_1 and V_3, the magnitude of the current in the branch between the nodes, I_y, becomes

$$I_y = \omega\frac{A}{h_y}\frac{2\epsilon_1\epsilon_2}{\epsilon_1 + \epsilon_2}(V_1 - V_3). \tag{2.21}$$

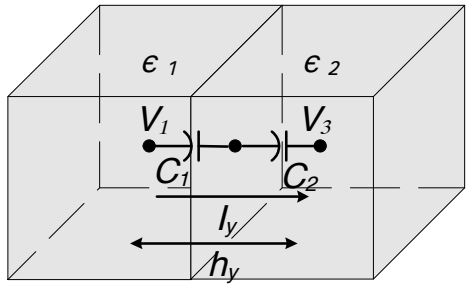


Figure 2.6: Two adjacent voxels showing the equivalent capacitance model across one interface. Each voxel is of uniform dielectric material. Voltages are shown at the centers of each voxel, corresponding to the network nodes.

Equation 2.21 is the circuit analogue of Equation 2.17. When a node equation is written, the sum of the currents into the node must equal zero. Since A has area $h_x h_z$, the equation will have identical form to Equation 2.18. So, the generic node/voxel equation is

$$\begin{aligned}
&\frac{h_y h_z}{h_x}\left(\frac{\epsilon_{i+1,j,k}}{\epsilon_{i,j,k}+\epsilon_{i+1,j,k}}(\Phi_{i+1,j,k}-\Phi_{i,j,k})-\frac{\epsilon_{i-1,j,k}}{\epsilon_{i,j,k}+\epsilon_{i-1,j,k}}(\Phi_{i,j,k}-\Phi_{i-1,j,k})\right)+\\
&\frac{h_x h_z}{h_y}\left(\frac{\epsilon_{i,j+1,k}}{\epsilon_{i,j,k}+\epsilon_{i,j+1,k}}(\Phi_{i,j+1,k}-\Phi_{i,j,k})-\frac{\epsilon_{i,j-1,k}}{\epsilon_{i,j,k}+\epsilon_{i,j-1,k}}(\Phi_{i,j,k}-\Phi_{i,j-1,k})\right)+\\
&\frac{h_x h_y}{h_z}\left(\frac{\epsilon_{i,j,k+1}}{\epsilon_{i,j,k}+\epsilon_{i,j,k+1}}(\Phi_{i,j,k+1}-\Phi_{i,j,k})-\frac{\epsilon_{i,j,k-1}}{\epsilon_{i,j,k}+\epsilon_{i,j,k-1}}(\Phi_{i,j,k}-\Phi_{i,j,k-1})\right)=0, \quad (2.22)
\end{aligned}$$

where Φ is interchangeable with ωV depending on the perspective. This formulation assumes a uniform voxel size throughout space. If a varying grid is used, h_x, h_y, and

h_z will be dependent on position. Changes to Equation 2.22 will be of form:

$$\frac{h_y h_z}{h_x} \frac{\epsilon_{i+1,j,k}}{\epsilon_{i,j,k} + \epsilon_{i+1,j,k}} \rightarrow \frac{h_{y_{i,j,k}} h_{z_{i,j,k}} \epsilon_{i+1,j,k}}{h_{x_{i+1,j,k}} \epsilon_{i,j,k} + h_{x_{i,j,k}} \epsilon_{i+1,j,k}}. \tag{2.23}$$

For small networks, it is possible to derive the input capacitance to a boundary node in terms of the network capacitances only. This allows each of the values in the capacitance matrix to be calculated from the network of capacitors. Further simplifying the problem is that the network capacitors only need to take on one of two possible values. Since capacitance is only a function of the geometry of the conductors and position of dielectric, this has the ability to be a very powerful method since it bypasses the need for also computing the electrostatic potential.

Unfortunately, finding the input capacitance in terms of a large 3D lattice of capacitance elements is not trivial. Since the lattice is repetitive in nature, it stands to reason that perhaps the mathematics might also be repetitive. One way to build this expression is to rewrite Equation 2.22 to solve it for $\Phi_{i,j,k}$. Boundary conditions are applied and the voxels must be given a starting value between the boundary voltages. Equation 2.22 can then be solved at each voxel via a relaxation technique. This must be done enough times over the entire solution space to improve accuracy and allow convergence. Each voxel potential will then be in terms of the dielectric distribution. The capacitance matrix can then be found by applying the discrete form of Equation 2.4. Diophantine mathematic techniques [19] could possibly be applied

to solve the problem, since the dielectric is assumed to be one of two possible values that can be chosen to be integer values.

Mathematica was used to find the 8 unknown potentials on a small $4 \times 4 \times 4$ grid in terms of the 8 unknown dielectrics. The relaxation needed to be run over the grid 5 times in order to achieve numeric answers that had a high degree of accuracy. Each successive relaxation iteration over the grid took approximately an order of magnitude more time than the previous one, with the fifth one taking 147 seconds. Memory demands also grew rapidly with the fifth exceeding 1 GB of RAM. Today's computers and algorithms are optimized for numeric computation and it was much more efficient to guess a dielectric distribution, solve for the potentials numerically, and then calculate the capacitances from the potentials. In the end, this is the method that was used to solve the forward problem for the image reconstruction algorithm described later.

2.4.3 Rationale for using self capacitance

To see how limiting the dielectric distribution to discrete values allows detailed images to be reconstructed from just self capacitance data, it is helpful to examine the mathematics and a few examples. For a system with $i \times j$ voxels parallel to the conductor planes and k voxels between conductor planes, $i \times j \times k$ equations result for each set of boundary conditions. By applying a voltage to each conductor in

turn with the others grounded, n sets of boundary conditions are needed to describe an n conductor system. An additional n^2 equations result from a discrete form of the capacitances equations (equation 2.4). For a closed system, only $n(n-1)/2$ are independent due to conservation of charge and symmetries. So, the total number of equations is $n \times i \times j \times k + n(n-1)/2$. The total number of unknown potentials in the grid is $n \times i \times j \times k$, and the total number of unknown relative dielectrics is $i \times j \times k$. Equating the number of equations and unknown variables reveals that

$$\frac{n(n-1)}{2} = i \times j \times k. \tag{2.24}$$

For the ECVT application in this paper, two parallel 4×4 arrays of electrodes were modeled along with a single conductor surrounding the other sides, resulting in a 33×33 capacitance matrix. $i \times j \times k$ must equal 528 to have a well determined set of equations. This represents the maximum number of unknown voxels of dielectric material that can be uniquely determined from the capacitance measurements, but the underlying assumption of this computation is that both the unknown values of the electrostatic potential and the dielectric constants can take on a continuum of values.

In many real applications, the dielectric constants of the materials to be imaged are well known and can be assumed to have constant values. By restricting the values of the dielectric material, both sides of the equation 2.4 will take on discrete rather than a continuum of values. As an example, for 528 voxels containing dielectric

material or air, 2^{528} possible distributions exist with corresponding sets of discrete capacitances. The ultimate limit to the number of voxels that can be used to model a fixed volume is determined by the ability of the measurement system to resolve differences between adjacent capacitance values.

Since we are concerned with the inverse problem, the question becomes one of uniqueness, does each dielectric profile map to a unique set of capacitances? Start by constructing a simple 3×3 square grid bounded by four conductors. This two dimensional model is shown in figure 2.7, where the inner 3×3 pixels of dielectric material are assumed to be unknown. According to equation 2.24, the four conductors

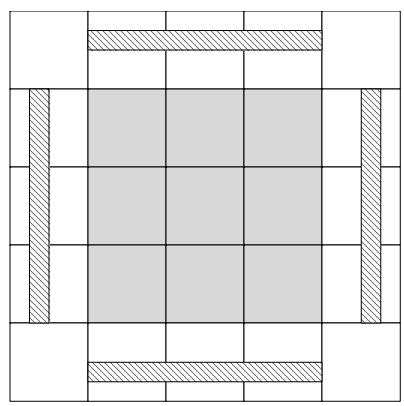


Figure 2.7: Two dimensional cross section of a ECT system with a 3×3 grid of unknown dielectric material. Four electrodes are shown surrounding the region.

will produce 6 independent measurements which are insufficient to yield 9 dielectric values, however, by limiting each pixel to one of two dielectric constants, higher resolutions are achievable. Since the 9 pixels are limited to two values, $2^9 = 512$ distributions are possible. Calculating the capacitance matrices for each of these distributions, it was found that there is no degeneracy when considering the full ca-

pacitance matrix. Looking at only the self capacitances, a single pair is degenerate (figure 2.8). Ignoring this one case, this indicates that making four self capacitance measurements to sufficient resolution will yield a unique solution.

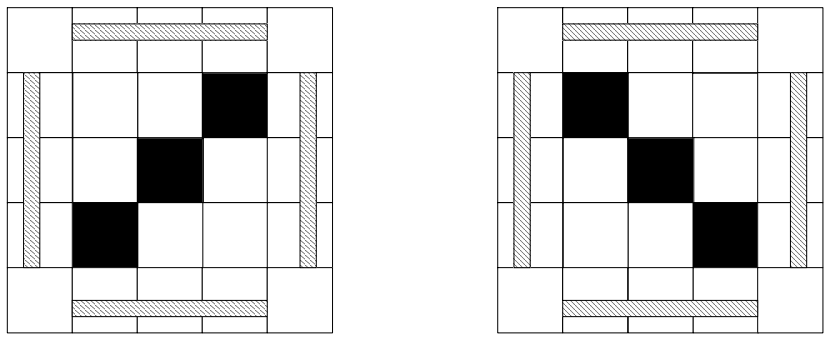


Figure 2.8: Two dielectric distributions with the same four self-capacitances

A second configuration consisted of 4×4 pixels of unknown material surrounded by four conductors, giving $2^{16} = 65,536$ possible dielectric distributions. Again, each dielectric distribution was determined to have a unique capacitance matrix. When considering just the self capacitances, 120 pairs of dielectric distributions where found to be degenerate.

By comparing the number of degenerate states with the total number of states, it can be seen that the 3×3 grid has a 0.391% degeneracy versus 0.366% for the 4×4 grid. As the grid size is expanded, the percentage of degenerate states remains small. These degeneracies are due to the symmetry of the conductors and could be broken by changing the size or position of the conductors. These were 2D examples, but similar properties extend to 3D geometries with three sets of parallel electrodes surrounding a volume in a cubic fashion. Given the small percentage of degenerate

states along with the ease of measuring self capacitances, it was decided to base our ECVT system on self capacitance measurements alone.

2.4.4 Grid design considerations

The initial grid size selected used four voxels in a 2×2 arrangement to model each conductor. The capacitances without the sample present are subtracted from the capacitances with the sample. Therefore, it was assumed that the corresponding change in charge density didn't vary much across the surface of the conductor and could be represented by an average value. Each of these average values are computed from the change in potential normal to the conductor surface as in Equation 2.5. It turned out that the capacitance calculated in this fashion, can vary on the order of $\pm 10\%$ from the actual capacitances measured on an ideal system. This can be illustrated from the following example.

The change in the potentials was calculated for a dielectric sample filling one quarter of the volume between the electrodes. A 100 voxel (10×10) per electrode model was used to highlight the differences with the four voxel model. The model of the sample is shown in Figure 2.9 along with the change in potential in volts for an electrode directly below the sample and also for an electrode adjacent to the distribution. Since the two distributions shown are non-linear, it is easy to see why four voxels per electrode are inadequate to properly calculate the capacitance.

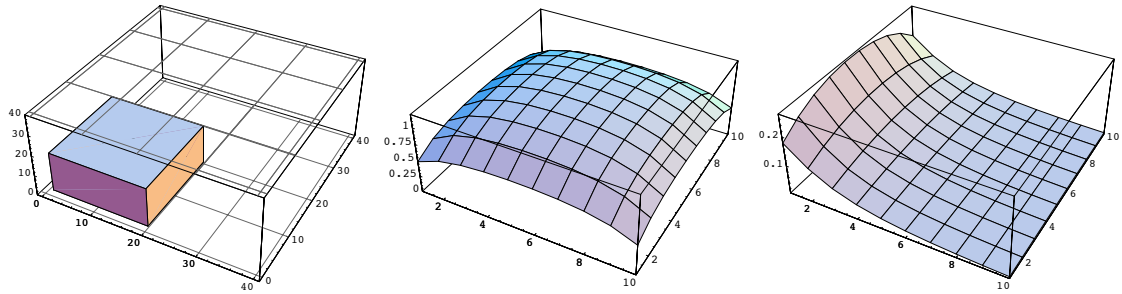


Figure 2.9: When dielectric material is introduced (left), the corresponding change in charge distribution on the conductors can be quite varied. The center figure shows the change in potential in volts when a conductor underneath the dielectric is raised to 2.5 volts. The right figure shows the change for the next conductor over, adjacent to the dielectric.

For the distribution in the middle, a four point model will overstate the change in capacitance. However, it will understate the capacitance for the distribution on the right.

Because grid density greatly affects computation time, it is important to not end up with a more dense grid than necessary. One might ask the question will moving the conductors further apart help reduce non-linear behavior? Do the charges always pile up on the edges? Landau and Lifshitz [9] provide the closed form solution for the charge distribution on an ellipsoid shaped conductor with no other nearby conductors:

$$\sigma(x, y, z) = \frac{q}{4\pi abc}\left(\frac{x^2}{a^4} + \frac{y^2}{b^4} + \frac{z^2}{c^4}\right)^{-\frac{1}{2}}, \tag{2.25}$$

where, a, b, and c are the semi-axis lengths in the x, y, and z directions, respectively and q is the total charge on the ellipsoid. By letting $a = 3$, $b = 2$, $c = 1$, and $q = 1$, a normalized plot of the charge density shows that the charge is densest at the points

furthest from the center of the ellipsoid. This is shown in Figure 2.10. Consequently, the charges will pile up along the edges and corners of the square conductors, even without other nearby conductors.

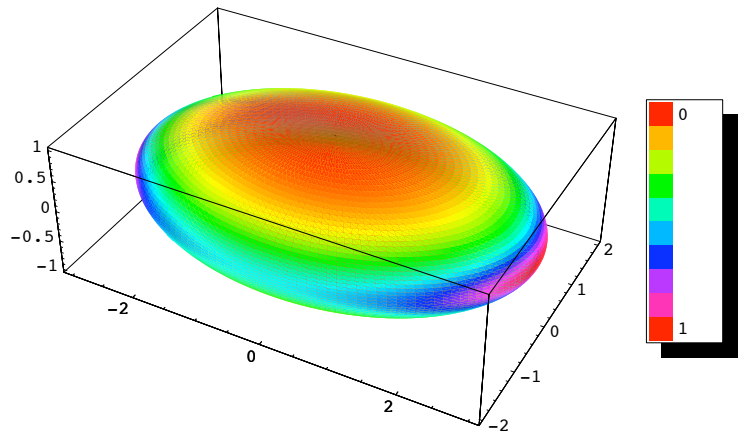


Figure 2.10: Plot of the normalized surface charge density on an conducting ellipsoid in free space.

These physical insights were used to arrive at a model containing 51200 voxels, arranged in a $40 \times 40 \times 32$ grid with the dimensions of h_x, h_y, and h_z nominally equal to 0.2 cm, 0.2 cm, and 0.1 cm, respectively. Values for h_x and h_y were varied slightly to allow voxel centers to line up on the the edges of the conductors, giving a more faithful representation of the capacitance. Two additional planes of voxels were used to model each of the conductor arrays. The first one was used to apply the boundary potentials, the second was spaced a very small distance away to allow accurate computation of the capacitances from the change in potential normal to the surface.

In the previous section on FDM, a set of examples were presented comparing the method to the theoretical computation of the electrostatic potential between two electrodes of an ideal parallel plate capacitor, partially filled with dielectric material. This example is repeated here for the FVM. The results are shown in Figure 2.11 and Figure 2.12. It can be seen that the FVM values closely match the theoretical values. One strength of the FVM is the way it cleanly avoids mathematically handling the discontinuity. However, this same strength could be considered a weakness since it cannot be used to calculate the potential at the discontinuity.

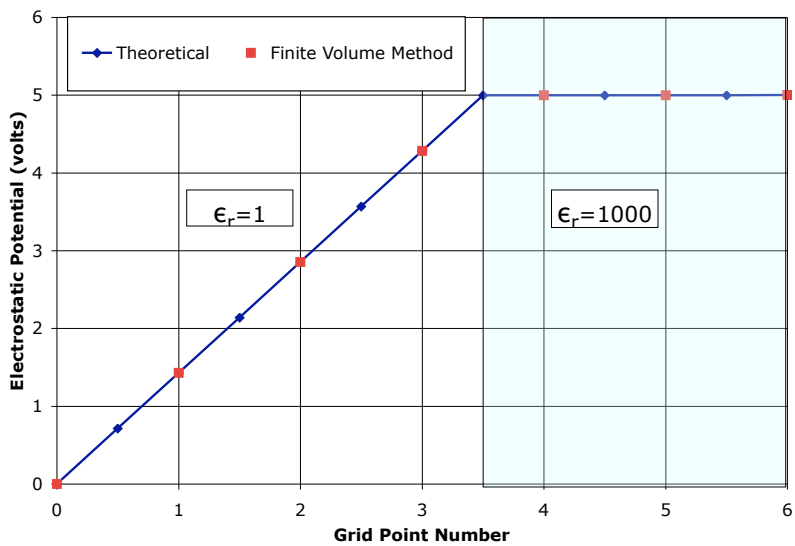


Figure 2.11: Plot of electrostatic potential for an idealized parallel plate capacitor partially filled with dielectric material ($\epsilon_r = 1000$) and 5 volts applied across the conductors. The potential was computed with the FVM and the standard theoretical model for this type of problem.

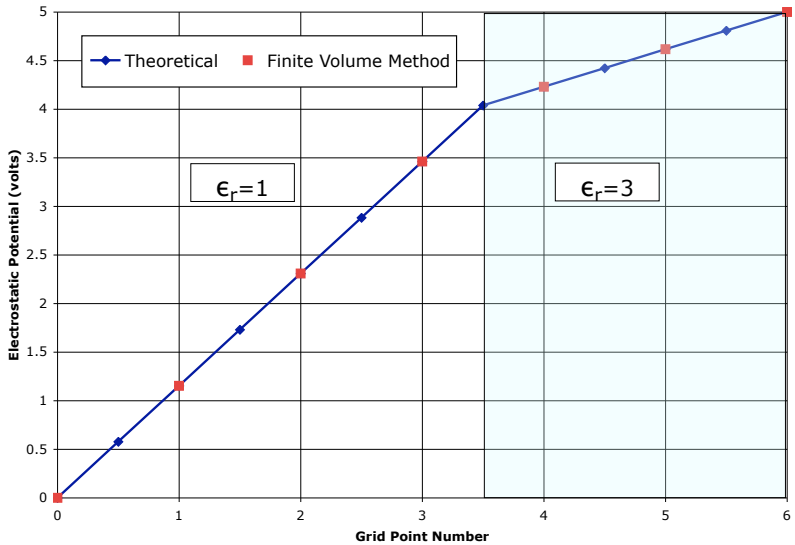


Figure 2.12: Plot of electrostatic potential for an idealized parallel plate capacitor partially filled with dielectric material ($\epsilon_r = 3$) and 5 volts applied across the conductors. The potential was computed with the FVM and the standard theoretical model for this type of problem.

2.5 Algorithm description

Yang and Peng's review paper [7] provides a summary of work on image reconstruction algorithms for ECT, including iterative methods that address high contrast image reconstruction. More recent work on non-linear image reconstruction [20] uses an iterative regularized Gauss-Newton scheme that recomputes the sensitivity matrix with each iteration. In general, these iterative methods assume that the dielectric distribution can take on a continuum of values, making the inverse problem ill-posed when the resolution exceeds the number of independent measurements. They also

are based on incremental changes of the dielectric distribution which is incompatible with the present case where only two dielectric values are possible.

The system of equations required for our algorithm is found by applying FVM across the entire grid of voxels. Equations 2.18 and 2.19 are combined, canceling D, to yield a governing equation for each voxel containing only the electrostatic potential and the dielectric. The resulting set of equations is used with applied boundary voltages to repeatedly compute the electrostatic potentials for a series of guessed dielectric distributions. These equations are expressed as a matrix consisting of coefficients based on dielectric values of the voxels, a vector of the unknown potentials, and a solution vector dependent on the boundary conditions and the dielectric values of the voxels. When a guess is made of the dielectric distribution, the unknown potentials are found for each of the boundary conditions using linear algebra. The matrix is sparse and unsymmetric, so the LinearSolve command in Mathematica finds the solution using an unsymmetric multi-frontal method written by Tim Davis called UMFPACK [21]. Since the potentials are then known everywhere, the capacitances and change in capacitances are calculated via the discrete forms of equations 2.4 and 2.5.

As shown in the previous section, constraining the dielectric permittivity of the constituents to one of two values changes the nature of the math problem, allowing high resolutions to be generated. For this work, the low dielectric material was assumed to be air with an approximate relative dielectric value of 1. The high

dielectric material was assumed to be water with impurities making it act like a conductor, giving it a relative dielectric much higher than the standard table value of 80 associated with de-ionized water. A voxel in an ECVT filled with high permittivity material has a strong influence on capacitance when compared to low dielectric material such as glass or air. However, there is little difference between material with relative dielectrics above 80 due to a saturation effect [20], so that at low frequencies, capacitance measurements will differ little between water and metal. An aluminum structure in an air background is used to represent the high contrast dielectric material since it is electrically similar to water but easier to precisely locate within the ECVT system. A perfect conductor has an infinite permittivity, but can be fairly well approximated by assuming a relative dielectric value greater than 80. (In this case, a value of 99,999 was used.)

The algorithm created for this work is a form of steepest descent method, seeking to minimize a figure of merit (FOM) that is a measure of how close a guessed solution is to the global minimum. Steepest descent methods begin at some point in multi-dimensional space uphill from the global minimum. The figures of merit are calculated for a series of guessed solutions and compared to the FOM for the starting point. The guess that provides the best change in FOM from the starting point is selected and becomes the new starting point. This process continues until the global minimum is found or the algorithm becomes trapped in a local minimum. Steepest descent methods are all prone to getting stuck in local minima, so key parameters are

tuned to provide improved performance according to the physics of the problem to be solved. In our algorithm, the tuning parameters are: the starting point selection, the FOM, guess selection, and the criteria for determining which downhill move to make. The next few paragraphs elaborate on these details and provide the specific steps used in the code.

The image reconstruction algorithm begins by assuming a starting dielectric distribution which totally fills the volume between electrode planes with high dielectric material, with the exception of the layer immediately above each electrode. The algorithm then carves away at this by toggling the dielectric constant of small groups of voxels until a minimum FOM is reached. The FOM is based on an L1 norm and has the following form:

$$FOM = \frac{\sum_i |C_{ii}^M - C_{ii}^G|}{\sum_i |C_{ii}^M|}. \tag{2.26}$$

C_{ii}^M represents the difference between the measured self capacitances with and without the dielectric material present. For a given guess, C_{ii}^G represents the difference between the calculated self capacitances with and without the dielectric material. The variable, i represents the array indices that run from 1 through the number of conductors.

Since groups of voxels in the center between electrodes produce a weaker change in self capacitance than those closer to the electrodes, a scale factor is multiplied by the calculated change in the FOM for the purpose of determining which downhill step to make. This scale factor is equal to one over the change in self capacitance

of the nearest electrode when a group of voxels is filled with high dielectric material while all other voxels contain low dielectric material.

The position of each group of voxels is tracked by a counter associated with the nearest electrode. A set of guessed solutions is obtained by beginning with the starting dielectric profile and toggling in turn the dielectric state of the group of voxels pointed to by each counter. Since there are 32 electrodes and one counter per electrode, this results in 32 guessed solutions per iteration of the algorithm. The guessed solution that results in the largest decrease in the FOM (when multiplied by the scale factor) is assumed to be part of the solution. This is illustrated by the following pseudo-code:

$$BGI = Index[Max[SF(ctr_1)*\Delta FOM_1, SF(ctr_2)*\Delta FOM_2, ..., SF(ctr_n)*\Delta FOM_n]], \tag{2.27}$$

where BGI is the best guess index, $SF(ctr_n)$ is the scale factor associated with the nth conductor's counter value and ΔFOM_n is the difference between the starting FOM and the FOM for the guessed solution associated with the nth conductor. The starting FOM is reduced and the change in the dielectric constant is accepted as the new starting solution:

$$FOM_{l+1} = FOM_l - \Delta FOM_{BGI}, \tag{2.28}$$

where l is the iteration number. If the change in the FOMs are all less than or equal to zero, the original dielectric constant is retained for those voxels associated with the BGI in the pseudo-code below:

$$BGI = Index[Min[SF(ctr_1)*\Delta FOM_1, SF(ctr_2)*\Delta FOM_2, ..., SF(ctr_n)*\Delta FOM_n]], \tag{2.29}$$

The counter associated with the BGI is altered to skip over this group of voxels and the starting FOM remains unchanged. Only one counter is revised per iteration to allow for more gradual convergence.

The counters are initialized to begin toggling the dielectric state of the group of voxels closest to the electrode. As the algorithm progresses, each counter moves toward the opposing electrode. Once per iteration a the best solution is selected, the counter is updated corresponding to the voxels that changed (or didn't change based on negative changes in FOM), and the associated dielectric distribution becomes the new starting point. The process repeats until all opposing counters have crossed each other. Since the algorithm is a type of steepest descent method, it begins with the voxels that have the largest effect on the self capacitance readings and ends with those with the least impact. If the final FOM is not sufficiently close to the global minimum, the counters can be reset and the process repeated.

For this work, the voxels were arranged in a $40 \times 40 \times 34$ inner grid (10×10 over a single electrode with 34 positions between opposing electrodes) with the dimensions

of h_x, h_y, and h_z nominally equal to 0.2 cm, 0.2 cm, and 0.1 cm, respectively. Values for h_x and h_y were varied slightly to allow voxel centers to line up on the the edges of the conductors, giving a more faithful representation of the capacitance. Two additional planes of voxels were used to model each of the conductor arrays. The first one was used to apply the boundary potentials, the second was spaced a very small distance away to allow accurate computation of the capacitances from the change in potential normal to the surface.

With 100 voxels in each plane above each electrode, toggling each voxel individually in turn would create 3200 guessed solutions before updating the starting solution. However, to save on execution time, the entire 10x10 set of voxels above each conductor in each plane was assumed to be either full or empty creating 32 guessed solutions, one associated with each electrode. Because the system is least sensitive in the center between the arrays of electrodes, the middle four planes corresponding to $k = 17$ through $k = 20$ where grouped together, where k is the position of the voxels between the arrays and $k = 1$ and $k = 36$ are the positions of the sensing conductors. Likewise, the pairs $k = 15$, $k = 16$ and $k = 21$, $k = 22$ where grouped with each other. This results in 100, 200, and 400 voxels being grouped together, depending on their position with respect to the center of the array.

Specific steps in the algorithm:

1. Set up the equations to solve the forward problem for the electrostatic potentials according to FVM as described above.

2. Find the scale factors for each of the groups of voxels as described above.

3. Initialize position counters for each electrode to track the location of a guessed variation in the dielectric profile with respect to the closest opposing electrodes.

4. Compute potentials and capacitances (according to the discrete form of equation 2.4) for the system filled with low dielectric material.

5. Compute potentials and capacitances for the system filled with high dielectric material (with exception of a small gap of low dielectric material above the conductors, to keep from shorting conductors together).

6. Compute the change in self capacitances between an empty (low dielectric state) and full system (high dielectric state) to become the starting guess.

7. Compute the starting FOM using the measured self capacitances and the calculated change in capacitances for the starting guess.

8. Make a guess at the relative dielectric distribution by toggling the relative dielectric of the group of voxels pointed to by the counter for the selected electrode, according to the method described in the previous paragraphs.

9. Using the guess, solve the forward problem for the electrostatic potential using the equations in step 1.

10. Calculate the corresponding change in capacitances for the guess and its change in FOM from the starting FOM.

11. Repeat steps 7-9 for each of the remaining electrodes.

12. Multiply the change in FOM by the scale factors for each of the guessed solutions.

13. If the largest scaled change in FOM has a positive value: The counter for the associated electrode is changed to select the next group of voxels, one increment further from the electrode. The guessed solution and it's FOM are accepted as the new starting point for the next iteration.

14. If the largest scaled change in FOM is less than or equal to zero, the electrode associated with the most negative change is selected: The counter for the associated electrode is changed to select the next group of voxels, one increment further from the electrode, but the starting profile and FOM are retained for the next iteration.

15. When counters for opposing electrodes cross, they are both eliminated from steps 7-11.

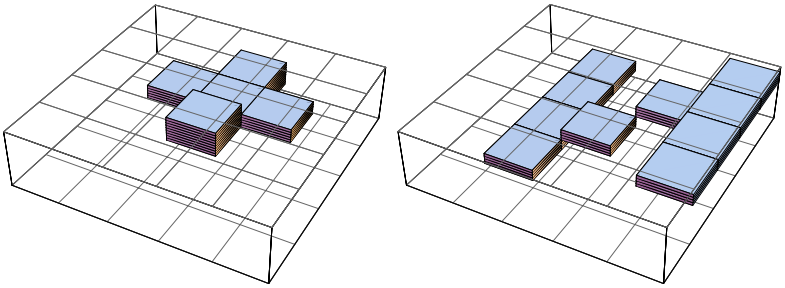
16. Repeat steps 7-12 until all of the counters for opposing electrodes cross each other.

17. If the final FOM is higher than desired, reset the counters and begin again at step 7.

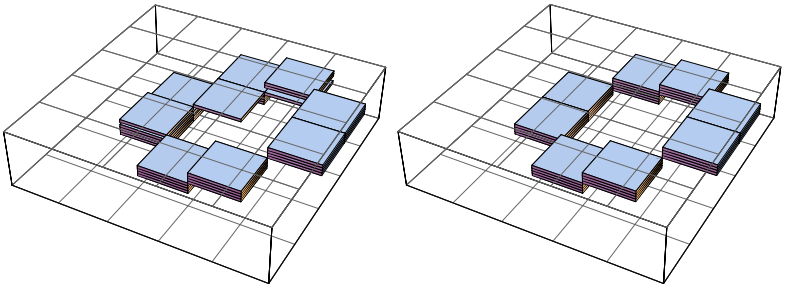
18. Repeat steps 7-14 until either a suitable FOM is reached or no further improvement is achieved.

For testing synthetic data, the resolution was reduced in the i, and j directions to have 1 voxel over the surface of an electrode. This doesn't model the capacitance accurately enough to compare it to actual data, but does allow much faster execution of the code to characterize the algorithm performance. Three different shaped synthetic samples were created in software to test the algorithm, a cross , a z, and a toroid. The algorithm was able to exactly reconstruct the images for the cross and the z, shown in figure 2.13a . The reconstruction of the toroid (figure 2.13b) took four passes of the algorithm, achieving a final FOM of 0.00587. Figure 2.13c shows the convergence of the algorithm for each of the shapes as material is removed from the starting profile. The logarithm of the FOM is plotted to show the rapid change in convergence that occurs when sufficient material is removed to begin exposing the shape.

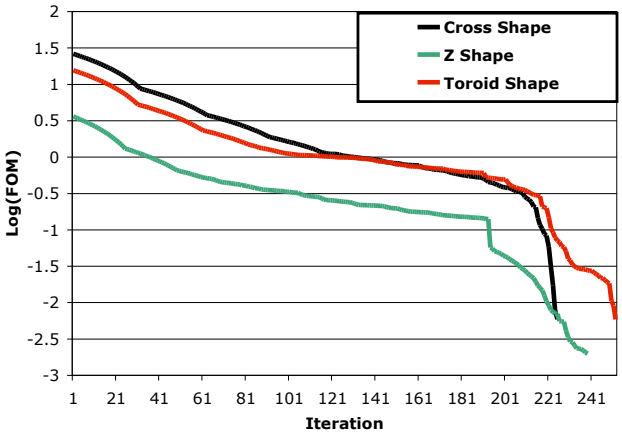
A series of tests were performed to determine the sensitivity of the reconstruction algorithm to noise. Increasing amounts of root mean square (RMS) noise were added to the calculated capacitances associated with the synthetic cross shaped sample. The result of these tests are shown in figure 2.14, revealing that the number of errors in the reconstructed image starts to increase above approximately 1 fF of RMS noise.



(a) Exactly reconstructed images of synthetic cross and z shapes.



(b) The reconstructed toroid image is shown (left) along with the ideal reconstruction (right).



(c) Curves showing logarithm of the figure of merit versus the iteration number as material is removed. The last point on the cross and z shapes has been removed, since the FOM converged to zero.

Figure 2.13: The reconstructed images are given for three different simulated high dielectric objects placed between the two arrays of conductors. The graph shows the rapid convergence that occurs when sufficient material has been removed to begin exposing the shape of the object.

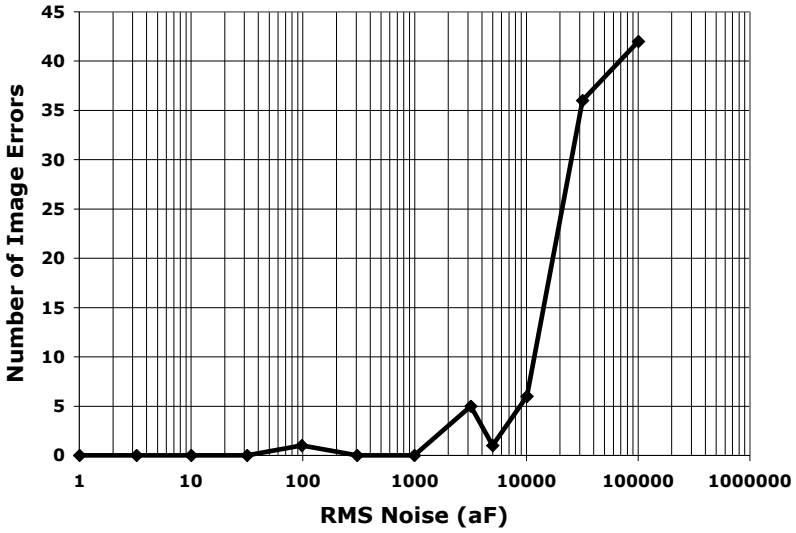


Figure 2.14: Graph showing the approximate number of errors in the reconstructed image versus the amount of RMS noise added to ideal data. Each data point was found by adding a fixed amount of RMS noise to synthetic data generated from the cross shape shown in figure 2.13a. The algorithm was then run repeatedly until no further improvement occurred. Looking at the groups of voxels as either high or low, the error represents the number of "bit flip" it would take to convert the final image to the ideal noise free image.

CHAPTER 3
ELECTRONICS DESIGN

The performance of an ECT system is dependent on many factors such as geometry, number and size of electrodes, the material to be imaged, and the required speed and image detail. No one set of performance requirements will apply equally to all ECT systems. However, it is helpful to start with a set for reference. Huang [22] provides requirements for a typical 12 electrode industrial system used for imaging pipe flow of oil/gas. Salient characteristics mentioned are a resolution of 0.3 femtofarads (fF), a dynamic range of 0.3 fF to 2.0 pF, an rms noise level of less than 0.1 fF, low baseline drift, and an acquisition rate of 10 ms for one frame of data. This section will focus on evaluating the electronics performance for an application where image detail is more important than acquisition rate. Therefore, the emphasis will be on discussing range, resolution, noise, and drift.

Resolution is the key parameter for determining the level of detail that can be reproduced in an image. In general, the better the resolution, the higher the reproduced image detail. However, for a given geometry, a 0.5 fF resolution might produce adequate images for water, but poor images for a lower dielectric material like oil. Ideally, electronics capable of resolving a single charge change would be desirable for the best image reproduction. For a 2.5 volt excitation, this corresponds to a change in capacitance of 0.06408 attofarads (aF). This is well below the detection limit of today's capacitance measurement circuits. However, Analog Devices, Inc.

has recently released a sigma-delta converter capable of resolving down to 4 aF, an order or two of magnitude better than a typical commercial LCR meter. The next few subsections will explore the application of this new chip in an ECVT system.

3.1 ECT measurement system

The EVAL-AD7746 evaluation kit [23] [24] became available from Analog Devices, Inc. in May 2005 and was used as the cornerstone for measuring the elements of the capacitance matrix for the ECVT system. The AD7746 is a capacitance to digital converter chip based on the class of devices referred to as sigma delta converters [25]. The EVAL-AD7746 consists of an AD7746 mounted on a circuit board with a USB interface, connectors for the excitation supplies and one of the input channels, and some flexible space on which to mount auxiliary circuitry.

3.1.1 Sigma delta converter overview

A sigma delta converter is essentially a one bit analog to digital converter formed by running the measurement signal through an integrator and then a comparator. Feedback is routed to a summing node on the input to the integrator to try and balance the input leads to the comparator. The difference (delta) between the reference and input signals is coded by the comparator. The output of the comparator is tied to a

counter to keep track of the number of times the comparator has a logic 1 output. Another counter is used to keep track of the number of clock cycles. This is the sigma or summing part of the name. The output is the ratio of the two counters times the reference voltage. So, for an input voltage that is one half of the reference, the comparator will output a logic 1 to the counters half of the time. Extensive digital signal processing is done on the output to remove noise and extend the resolution. The benefit of this technology is the high resolutions that are achievable, albeit at the price of slower sample rates. A more detailed description can be found in the paper by Aziz [25].

When configured to receive a voltage input, a sigma delta converter uses a reference capacitor in series with the reference voltage and another in series with the input signal. It was noted in a paper by Brytcha [23] that the design could be altered to measure capacitance by applying a known input voltage and allowing the input capacitor to vary instead of the reverse. The resulting simplified circuit is shown in Figure 3.1 . Further information on the AD7746 and EVAL-AD7746 used in this design and sigma delta converters in general is available through the Analog Devices website [24].

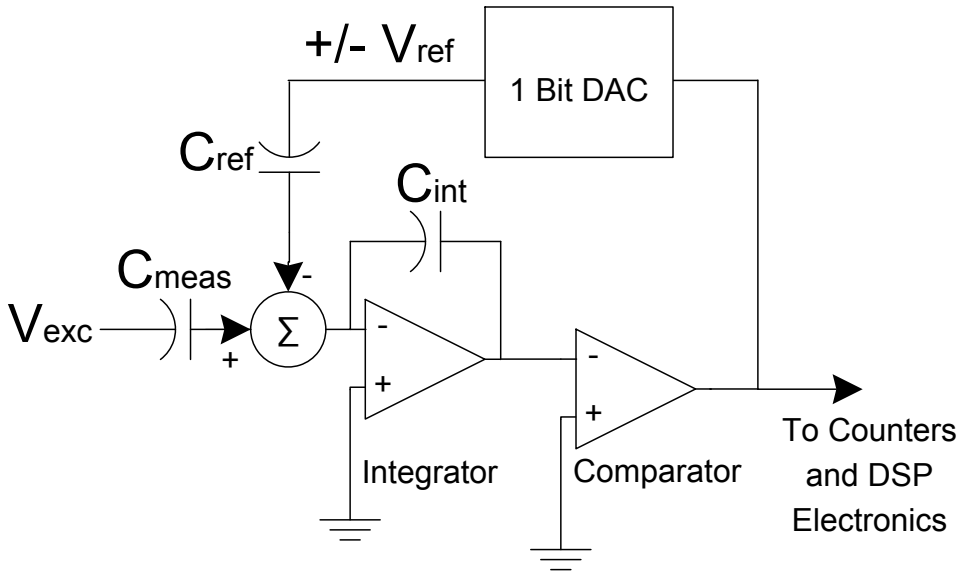


Figure 3.1: Simplified diagram of sigma-delta converter for capacitance measurement

3.1.2 Electrode design

Two 4×4 arrays of electrodes were milled from square sections of copper clad circuit board material and mounted on two 0.81 cm (0.25 in) thick pieces of glass-filled plastic backing. The glass-filled plastic backing was cut to approximately square shapes, 10 cm on a side. Each electrode is 1.9 cm on a side, placed on 2.0 cm centers. The parallel arrays were mounted with the electrode arrays 3.3 cm apart. The open ends were surrounded by a metal guard to reduce the sensitivity of the electrodes to outside interference and to help maintain the spacing between the arrays. Figure 3.2 shows an exploded view of this configuration. Ribbon cable was used to connect each of the electrodes through a penetration through the back of the plastic to a

connector on its edge. A ribbon cable was also used to then connect each array to the switch network.

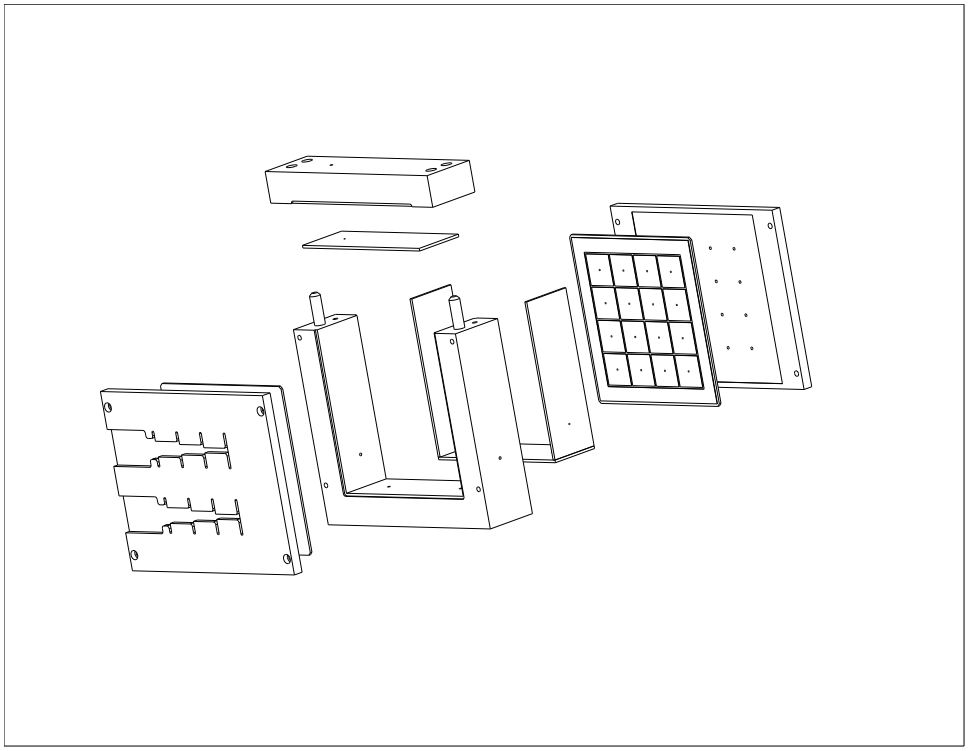


Figure 3.2: An exploded view of the ECVT sensor showing the position of the two 4×4 arrays of conductors.

3.1.3 Switch network design

A switch network was designed and fabricated to switch each conductor to either the excitation side of the AD7746 or the input side. Figure 3.3 shows a simplified version of the electronics for switching 3 conductors. This pattern is then repeated for the other conductors. The 74HCT4053 analog multiplexer was selected to perform the

switching due to the small parasitic capacitances, 5 pF in the off state and 8 pF in the on state. However, with all 32 electrodes connected, only around 68 pF of offset was seen in practice. To keep noise to a minimum, the switch electronics were powered from batteries. The AD7746 was configured to measure the single ended signal coming from the switch network.

3.1.4 Range extension

The dynamic range of AD7746 is ±4 pF, which is well suited to ECT measurement. It can also compensate nominally 17 pF of common mode capacitance, but this is too small for the 3D ECT system due to the added capacitances from switches and cabling. However, both ranges can be extended by adding a circuit to scale the excitation. This technique is explained in a preliminary technical note available through the manufacturer [26]. An ultralow noise op amp, Maxim MAX412MJA, and the two 1% resistors shown in Figure 3.3 were selected to expand each of the ranges.

Each AD7746 contains two capacitance measurement circuits and identical but separate excitation sources (Excitation A and B). This range extension technique requires both excitation sources on the AD7746, disabling one of the capacitance measurement circuits. Excitation A is inverted by writing to the appropriate register and run through one leg of a voltage divider. Excitation B is run through the other

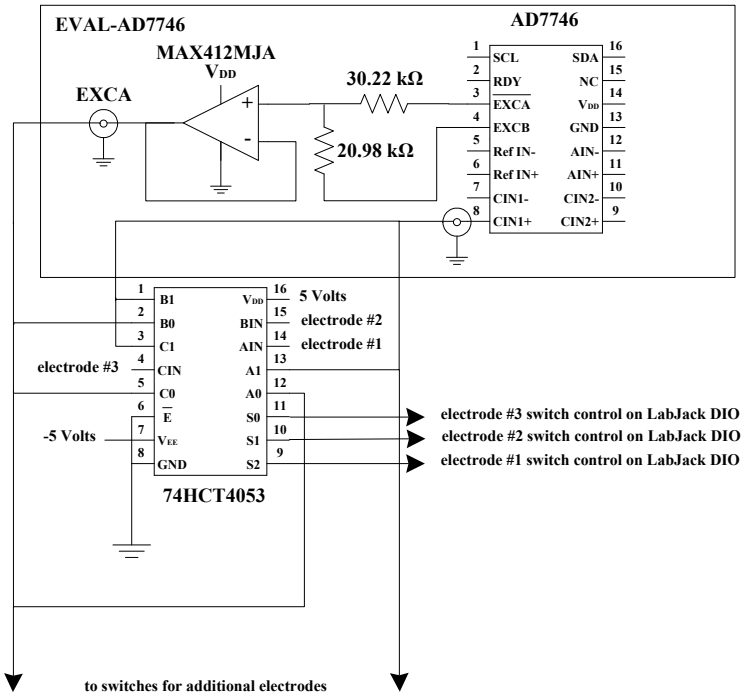


Figure 3.3: This simplified circuit diagram shows the basic element of the switch network for the electrodes and the modifications needed on the EVAL-AD7746 to expand the range of the AD7746 Capacitance to Digital Converter. The amplifier and resistor components shown expand the range and offset compensation by a factor of 5.54.

leg, providing the summation of the two signals at the input to the buffer amplifier. This has the effect of reducing the amplitude of the excitation signal by a factor of 5.54. The down side of reducing the excitation voltage is that the noise is increased by the same factor. The capacitance digital to analog converter used for compensating common mode capacitance had an initial range of 18.917 pF on the specific board tested. Therefore, the offset compensation is changed to 104.8 pF and the dynamic range to ±22.69 pF as a result of the excitation alteration. The resolution is changed from 4 aF to 22.16 aF.

3.1.5 Shielding Design, Noise Reduction, and Design Evolution

The design of the ECVT sensor and associated electronics evolved through testing. An early version is shown in Figure 3.4. This version was not well shielded and allowed the cables to be easily disturbed. Small disturbances of cabling or boards positioning resulted in fempto-Farad sized changes in offset capacitances making it difficult to get consistent readings.

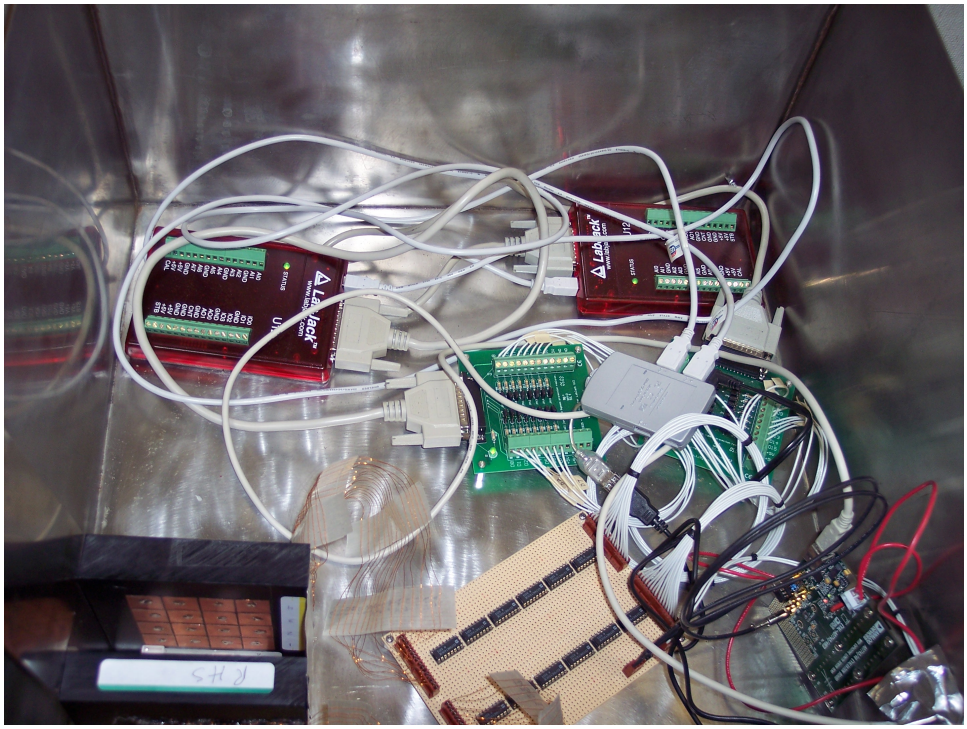


Figure 3.4: Photo of the first generation ECVT with copper tape electrodes, switch circuitry, and the EVAL-AD7746.

The second generation used ribbon cables to connect to the switch circuitry. However, they were still too long and caused high capacitance offsets resulting in the need for a range extension scaling factor of 10.57. The switch electronics were rigidly

mounted in a box with the EVAL-AD7746 to provide additional shielding and a more constant offset. This version is shown in Figure 3.5. Because the electrodes were constructed by hand with copper tape, the spacing and alignment of electrodes was less uniform than machine cut electrodes on copper clad circuit boards. Additionally, the tape had a tendency to separate at the edges from the plastic backing over time. This combined to introduce inconsistencies between the actual array and the one modeled in software for image reconstruction.

Figure 3.5: Photo of the second generation ECVT shielded with mu metal (cover removed). The switch circuitry and the EVAL-AD7746 where rigidly mounted in the gray box to prevent the signal cables from moving.

The third generation array was machined to address the aforementioned problems. The cover for the sample chamber was designed to sit on pins to reproduce the

same configuration both empty or when a sample was inserted. Another problem of earlier versions was lack of a sample holder to provide consistent alignment. It was difficult to achieve a known alignment when testing the reconstruction software. Small bumps to the worktable would often cause the sample to shift in the sample chamber causing wasted data runs. So, a sample holder and new test object were fabricated to eliminate these problems as well. A photograph of this final assembly is shown in Figure 3.6.

3.1.6 Control and measurement description

Two LabJack U12 modules were used to provide the digital control signals for the switches. Both the LabJack U12 and EVAL-AD7746 interface to the computer via a universal serial bus link. To help with communications and coding, the manufacturers each provided a data link library and code examples in National Instruments LabVIEW software. Overall control was therefore accomplished with virtual instruments created with LabVIEW software.

Self capacitance values are obtained by switching each electrode in turn to the CIN+ side of the EVAL-AD7746 while the others (including the guard) are connected to the excitation. Because the guard was not connected through a switch, its self capacitance could not be measured directly. So, image reconstruction is based on the self capacitances of the 32 sensing conductors. Although not used for this work,

mutual capacitances can be obtained by switching two electrodes at a time to the CIN+ side of the EVAL-AD7746 while the others (including the guard) are connected to the excitation. The mutual capacitances can then calculated by subtracting the self capacitances for each electrode and dividing by 2 as shown by

$$C_{i,j} = (C_{measured} - C_{i,i} - C_{j,j})/2. \tag{3.1}$$

Because each mutual capacitance is calculated from two larger self capacitance measurements, the noise and drift are more difficult to mathematically remove than for the self capacitances alone.

The actual self capacitance of each electrode with no sample present is approximately 2 pF on top of an offset of roughly 68 pF. For image reconstruction, only the changes in capacitance caused by the dielectric material are important. These changes are found by taking a set of readings void of dielectric material and subtracting it from a set measured with it present. As a result, the absolute accuracy of the AD7746 becomes less important.

3.2 Performance characterization of ECT electronics

To understand how suitable the electronics are for ECT, one needs to look at effects the environment has on the system. This was accomplished by configuring the

system to measure the self capacitance of one of the electrodes overnight to characterize measurement noise and drift. To keep noise to a minimum, the array assembly, LabJack modules, switch network, batteries, and EVAL-AD7746 were placed in a grounded stainless steel box. The EVAL-AD7746 was configured to sample at 4.56 Hz, the slowest acquisition rate available, but the one with the greatest noise suppression. LabVIEW software acquired the data from the EVAL-AD7746 and wrote the result to a data file on a PC laptop. Figure 3.7 shows the result of a 100 point moving average applied to the data to isolate drift effects and noise levels. Subtracting the moving average from the raw data to remove the drift component, reveals the noise to have a root mean square value of 1.5 fF. This is much higher than the manufacturer's specification for the AD7746 of 4 aF. Attempting to isolate the noise sources, a fixed capacitor hooked up directly to the EVAL-AD7746 revealed that approximately 40 aF of noise was due to the other electronics and connectors supporting the AD7746. When combined with the range extension circuitry, an overall noise level of 0.22 fF can be expected due to the gain of 5.54. Therefore, the majority of the noise is likely due to the added external switch circuitry and cabling. An improved circuit board design and packaging effort could be used to reduce this noise source.

Figure 3.8 gives a plot of the capacitance versus the temperature from a sensor on the AD7746 chip. This reveals that temperature is the dominant cause of drift in capacitance. Shifts in the data on the plot are likely due to changes in moisture in

the air. Earlier tests done on a 2nd generation sensor array and a Vaisala HMP45A humidity and temperature probe confirmed the dependence on both temperature and absolute humidity.

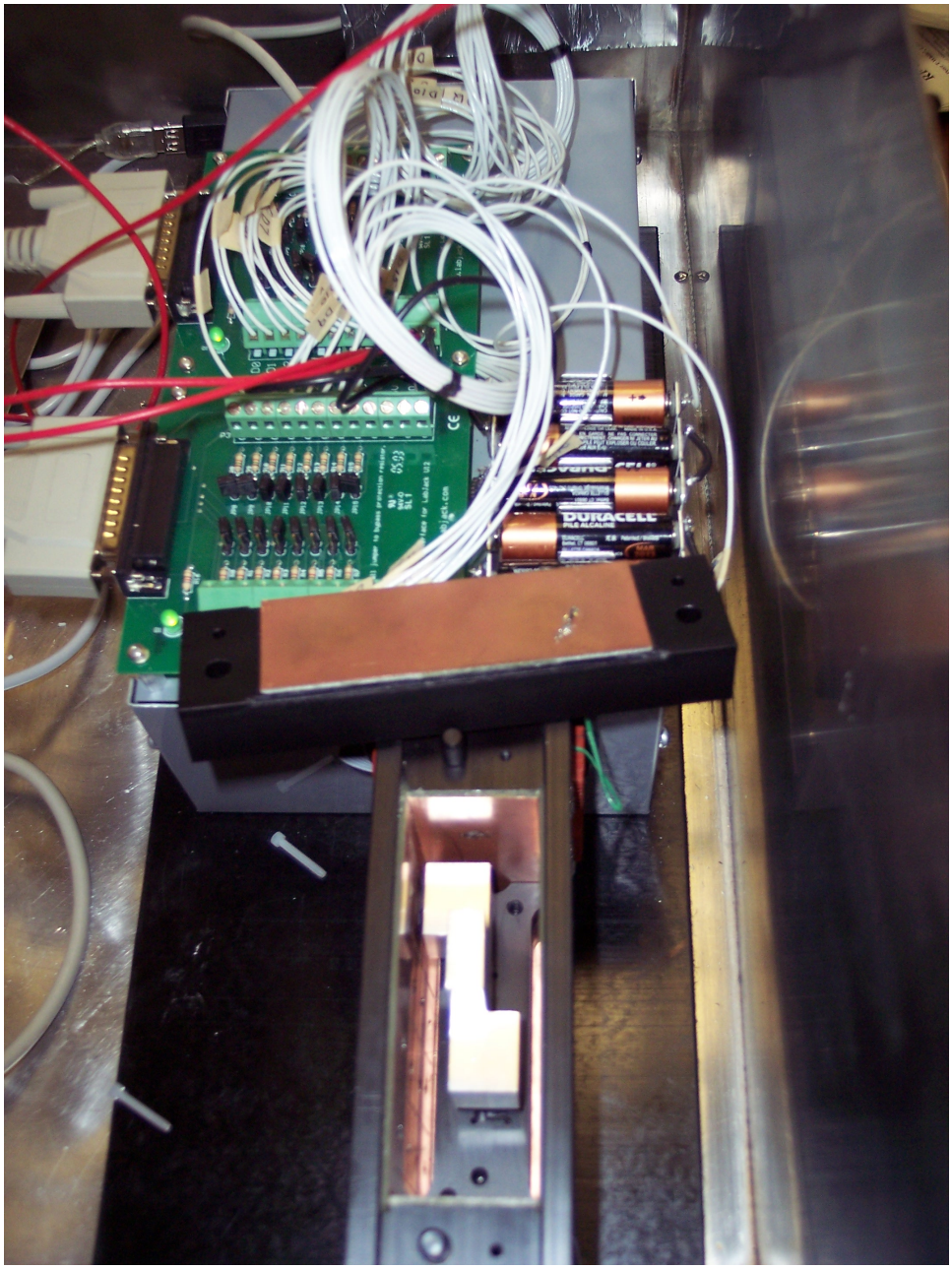


Figure 3.6: Photo of the third and final generation ECVT with the cover removed. The new sample holder and test object can be seen between the conductor arrays.

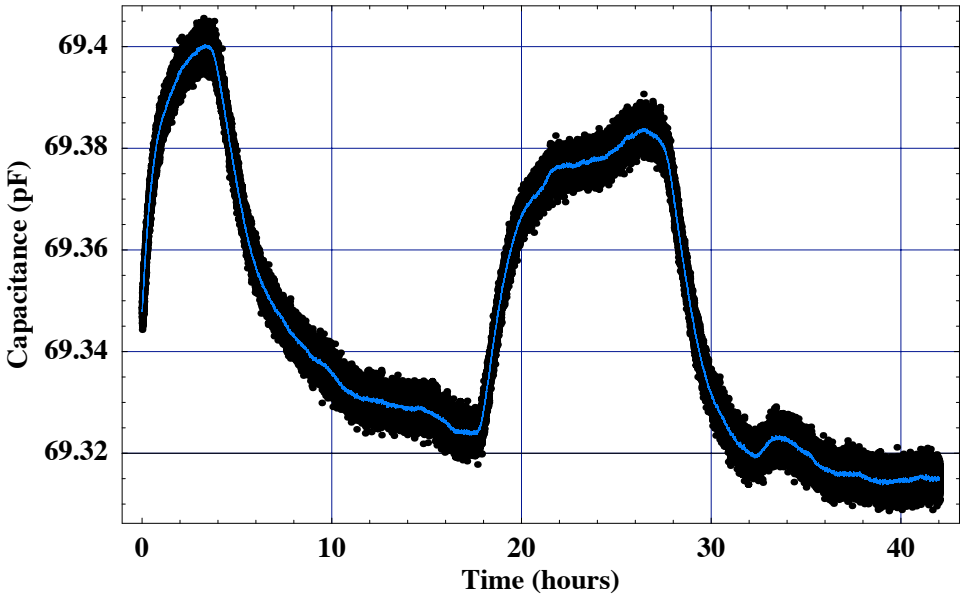


Figure 3.7: Plot of the self-capacitance readings from a single electrode in the 3D tomography system along with the 100 point moving average. The plot spans approximately 40+ hour time period.

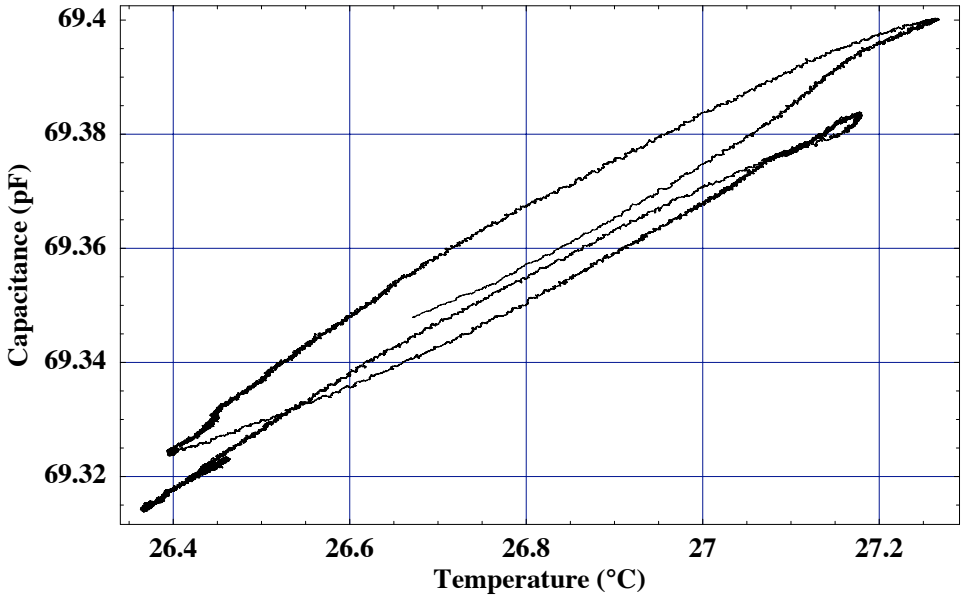


Figure 3.8: Plot of the capacitance versus temperature on the EVAL-AD7746. The data shown is the result of a 100 point moving average applied to the raw data. Temperature change is revealed to be the major cause of the drift in the self capacitance shown in Figure 3.7. Shifts in the curve are due to changes in absolute humidity.

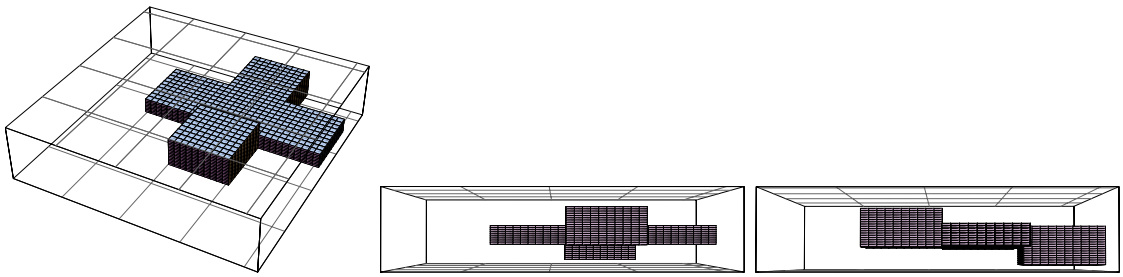
CHAPTER 4
3D IMAGING RESULTS

Several sets of data were collected with and without an aluminum structure that was shaped like the one used in the previously discussed synthetic data test. Each data run consisted of 100-300 capacitance and corresponding on chip temperature measurements for each of the 32 electrodes. If comparisons between the capacitance and temperature for a given electrode revealed drift, a linear fit was done to allow the data to be referenced to a common temperature. Otherwise, the points were averaged to provide a single value each for the capacitance and the temperature. The change in capacitances were calculated with these averages to provide input to the reconstruction algorithm coded in Mathematica.

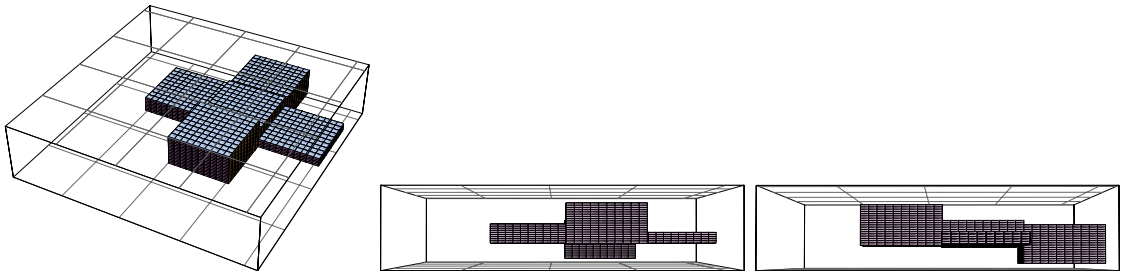
A sensitivity analysis was run on the model to determine the variation in capacitance that each voxel of dielectric material would cause with all other voxels void of material. The least sensitive voxels, located in the center between the conductor arrays, can cause approximately a 4 fF change in the self capacitance of its nearest conductor. If the other voxels are not void of dielectric material, the sensitivity can go higher or lower depending on the total dielectric distribution.

The root mean square difference (RMS) between the capacitances produced by the idealized model and the best set of data was 2.5 fF. By comparing this to the sensitivity in the previous paragraph and the results in figure 2.14, it is expected that the algorithm would reconstruct the image with a few possible differences in

low sensitivity areas. The reconstruction algorithm converged to a FOM of 0.0334 after the first pass, versus 0.0268 for the idealized profile. Figures 4.1a and 4.1b show a comparison between the images of the idealized structure and the one generated from actual data, with only two differences in the image details.



(a) Reference images of the aluminum structure.



(b) Reproduced images of the best set of data.

Figure 4.1: The top row shows the top and two side images of idealized aluminum structure to be compared against the second row which shows the corresponding reconstructed images from actual data. The images differ by two groups of voxels near the center between electrodes. Both differences are visible in the first reproduced image, appearing as voids in the top and right hand sides of the cross shape. The second and third reproduced images each show one of the two differences

Even though 1.5 fF of RMS noise was seen in experimental data, a 2.5 fF RMS difference was seen when comparing the measured capacitances to those calculated with an idealized representation of the object. The likely cause is that there are still some differences between the physical ECVT system and the model of it created in software. One suspect is the modeling of the electrodes. A fairly uniform grid

of voxels was used in a 10×10 arrangement per electrode. Since charge tends to concentrate near the edges of the conductors, a denser grid of voxels near the edges may produce a better correlation to the physical system.

CHAPTER 5
CONCLUSIONS

The change in self capacitances for 32 sensing conductors was used with a new algorithm to successfully reconstruct 3D images of conductive material on an inner grid containing 54,400 voxels, grouped into 464 blocks. By constraining the value of the unknown dielectrics, higher resolutions were achieved than would have been possible through conventional linear algebra. This new method was applied to high contrast dielectrics, extending the usefulness of ECT.

The algorithm took approximately one week to run in Mathematica on a 1.8 MHz PowerPC G5 computer. Higher resolutions are possible, but would take substantially longer to reproduce. The LinearSolve routine in Mathematica took approximately 40 seconds to solve the sparse matrix for a single guess. Since 32 guesses are made per iteration, this is largest consumer of the computer resources. It is likely that shorter processing time would be achievable with a compiled code and newer computing hardware.

Some of the presented ideas could benefit from further exploration. Two simplified examples illustrated that it is possible to obtain higher resolution images than the number of independent measurements would normally dictate. Further work needs to be done to better identify the limits for different electrode geometries using more mathematical rigor. Limiting the dielectric constants of the discrete elements to something other than a continuum of values facilitated the increased resolution.

However, a rather traditional approach of guessing a solution and solving the forward problem repeatedly formed the basis for the reconstruction algorithm. By limiting the dielectric constant to discrete values, diophantine mathematic techniques could possibly be applied to solving the system equations more directly.

The EVAL-AD7746 showed itself to be an attractive low cost option for use in ECT imaging. A large amount of offset capacitance was produced by the external switch network needed to access the electrodes. The EVAL-AD7746 was able to compensate for this by adding circuitry to divide the excitation signal by a factor of 5.54. This produced electronics capable of removing 104.8 pF of offset capacitance. The calculated dynamic range of the assembly is ± 22.69 pF with a resolution of 22.16 aF but the noise test revealed the rms noise to be 1.5 fF. This is much higher than that specified by the manufacturer for the AD7746 alone and is likely due to the added circuitry for the switch network and the range change. However, an acceptable level is achievable through averaging. Drift was minimized by collecting data over a period of a couple of hours and using an on chip temperature measurement to compensate the data. Newer chips are now available that provide automatic compensation for background temperature and humidity variations. They also contain the ability to switch in more than one sensor, reducing the need for external switch circuitry and making ECT electronics even more accessible.

APPENDIX: IMAGE RECONSTRUCTION CODE

A.1 Initialization

This Mathematica program computes the 3D dielectric profile as derived from the capacitance readings for a set of parallel 2 dimensional electrodes. This version contains 100 voxels per electrode, and tests solutions that fill 100, 200, and 400 discrete elements at a time in planes parallel to the electrodes. The code below initializes the majority of the variables for the program and sets up the voxel grid for use with the finite volume method.

```
<< Graphics`Polyhedra`;
<< LinearAlgebra`MatrixManipulation`;
<< Graphics`Graphics3D`;
(*Number of conductors in the i and j directions*)
nci = 4;
ncj = 4;
numconductors = nci * ncj * 2;

(*Number of voxels in i, j, and k directions*)
imax = 40;
jmax = 40;
kmax = 34;

(* vin is the voltage applied to the drive conductor *)
vin = 2.5;

(* hx, hy, hz - length in meters between gridpoints in the i,j,k directions. Distances
vary due to guard and capacitance computations. hz is also used to tune the proper
position of the sample *)
hx = Table[.00225, {i, 1, imax + 2}];
hx[[1]] = .01;
hx[[2]] = .001;
hx[[11]] = .001;
hx[[12]] = .001;
hx[[21]] = .001;
hx[[22]] = .001;
hx[[31]] = .001;
hx[[32]] = .001;
hx[[imax + 1]] = .001;
hx[[imax + 2]] = .01;

hy = Table[.00225, {j, 1, jmax + 2}];
hy[[1]] = .01;
hy[[2]] = .001;
```

```
hy[[11]] = .001;
hy[[12]] = .001;
hy[[21]] = .001;
hy[[22]] = .001;
hy[[31]] = .001;
hy[[32]] = .001;
hy[[imax + 1]] = .001;
hy[[imax + 2]] = .01;

hz = Table[.032/(kmax − 2), {k, 1, kmax + 2}];
hz[[1]] = .000000001;
hz[[2]] = .000000001;
hz[[3]] = .00122;
hz[[kmax]] = .00216;
hz[[kmax + 1]] = .000000001;
hz[[kmax + 2]] = .000000001;

(* Set up the dielectric constant, conductor area, permittivity of free space *)
diecon = 99999.;
conArea = .0190^2;
ϵ = 8.85;  (* pF/m *)

(*Initializes a table for the potentials of the voxels. neq is the number non-zero
elements in the sparse matrix. *)
ϕ = Table[0., {i, 1, imax + 2}, {j, 1, jmax + 2}, {k, 1, kmax + 2}, {l, 1, numconductors}];
neq = kmax ∗ jmax ∗ imax + 2 ∗ kmax ∗ jmax ∗ (imax − 1) + 2 ∗ kmax ∗ (jmax − 1) ∗ imax
+2 ∗ (kmax − 1) ∗ jmax ∗ imax;

(*Table of the relative dielectric values for each of the voxels*)
ϵ0 = Table[1., {i, 1, imax + 2}, {j, 1, jmax + 2}, {k, 1, kmax + 2}];
coordtab = Partition[Flatten[
Table[{i, j, k}, {i, 2, imax + 1}, {j, 2, jmax + 1}, {k, 2, kmax + 1}]], 3];

(*Initializes the table of the starting dielectric guess - full of material*)
ϵ = ReplacePart[ϵ0, diecon, coordtab];

(*The code below sets up a table to weight the charge contribution correctly when
summing the change in potential across the surface of the conductor. The edges should
contribute 0.5 and the corners 0.25. However, there seems to be a problem with missing
charge due to the way it bunches up on the edges. As a result, I chose to compensate
by weighting them differently*)
voxelweighttab = Table[1., {j, 1, jmax/ncj}, {i, 1, imax/nci}];
Table[voxelweighttab[[1, j]] = 0.85, {j, 1, jmax/ncj}];
```

```
Table[voxelweighttab[[imax/nci, j]] = 0.85, {j, 1, jmax/ncj}];
Table[voxelweighttab[[i, 1]] = 0.85, {i, 1, imax/nci}];
Table[voxelweighttab[[i, jmax/ncj]] = 0.85, {i, 1, imax/nci}];
voxelweighttab[[1, 1]] = 0.6;
voxelweighttab[[1, jmax/ncj]] = 0.6;
voxelweighttab[[imax/nci, 1]] = 0.6;
voxelweighttab[[imax/nci, jmax/ncj]] = 0.6;

(*Set up the counter and read in the scaling matrix*)
conCounter = Partition[Join[Table[4, {i, 1, 16}], Table[kmax − 6, {i, 1, 16}]], 4];
sensMatrix = Get["/Volumes/MNT1/sensMatrixHiRes9-20.mx"];
scaleMatrix = sensMatrix[[1]];

(*This code inserts vin on the proper electrodes*)
For[jc = 1, jc ≤ ncj, jc++,
For[ic = 1, ic ≤ nci, ic++,
For[j = 1, j ≤ jmax/ncj, j++,
For[i = 1, i ≤ imax/nci, i++,
{ϕ[[((imax/nci) ∗ (ic − 1) + i + 1), ((jmax/ncj) ∗ (jc − 1) + j + 1), 1,
ncj(jc − 1) + ic]] = vin;
ϕ[[((imax/nci) ∗ (ic − 1) + i + 1), ((jmax/ncj) ∗ (jc − 1) + j + 1), kmax + 2,
nci ∗ ncj + ncj(jc − 1) + ic]] = vin; }]]]]

Phitable = Table[0., {l, 1, numconductors}, {i, 1, 2 ∗ imax}, {j, 1, jmax}];
For[l = 1, l ≤ numconductors, l++,
Phitable[[l]] = Join[Table[ϕ[[i, j, 1, l]], {i, 2, imax + 1}, {j, 2, jmax + 1}],
Table[ϕ[[i, j, kmax + 2, l]], {i, 2, imax + 1}, {j, 2, jmax + 1}]]]
```

A.2 Finite Volume Method equations

This section of code sets up the discrete equations for the grid based on the finite volume method. This is just a discrete form of the integral representation of Gauss's Law. The result is a large sparse matrix (sparemat) based on the dielectric profile, ϵ, and a solution vector (bvec) based on the dielectric profile and the boundary voltages applied to each of the conductors. This section is called repeatedly with different guesses for the dielectric profile. This is the slowest section of my code.

```
sparsematcoords = Compile[{{imax, _Integer}, {jmax, _Integer}, {kmax, _Integer}},
Module[{i, j, k}, {Partition[Join[

Flatten[Table[{imax * jmax * (k - 2) + imax * (j - 2) + i - 1, imax * jmax * (k - 2)
+imax * (j - 2) + i - 1}, {i, 2, imax + 1}, {j, 2, jmax + 1}, {k, 2, kmax + 1}]],

Flatten[Table[{imax * jmax * (k - 2) + imax * (j - 2) + i - 1, imax * jmax * (k - 2)
+imax * (j - 2) + i - 2}, {i, 3, imax + 1}, {j, 2, jmax + 1}, {k, 2, kmax + 1}]],

Flatten[Table[{imax * jmax * (k - 2) + imax * (j - 2) + i - 1, imax * jmax * (k - 2)
+imax * (j - 2) + i}, {i, 2, imax}, {j, 2, jmax + 1}, {k, 2, kmax + 1}]],

Flatten[Table[{imax * jmax * (k - 2) + imax * (j - 2) + i - 1, imax * jmax * (k - 2)
+imax * (j - 3) + i - 1}, {i, 2, imax + 1}, {j, 3, jmax + 1}, {k, 2, kmax + 1}]],

Flatten[Table[{imax * jmax * (k - 2) + imax * (j - 2) + i - 1, imax * jmax * (k - 2)
+imax * (j - 1) + i - 1}, {i, 2, imax + 1}, {j, 2, jmax}, {k, 2, kmax + 1}]],

Flatten[Table[{imax * jmax * (k - 2) + imax * (j - 2) + i - 1, imax * jmax * (k - 3)
+imax * (j - 2) + i - 1}, {i, 2, imax + 1}, {j, 2, jmax + 1}, {k, 3, kmax + 1}]],

Flatten[Table[{imax * jmax * (k - 2) + imax * (j - 2) + i - 1, imax * jmax * (k - 1)
+imax * (j - 2) + i - 1}, {i, 2, imax + 1}, {j, 2, jmax + 1}, {k, 2, kmax}]]], 2]}]];

sparsematvals = Compile[{{ϵ, _Real, 3}, {hx, _Real, 1}, {hy, _Real, 1}, {hz, _Real, 1},
{imax, _Integer}, {jmax, _Integer}, {kmax, _Integer}},
Module[{i, j, k}, {Join[Flatten[Table
[- (hy[[j]]hz[[k]]ϵ[[i-1,j,k]] / (hx[[i]]ϵ[[i-1,j,k]]+hx[[i-1]]ϵ[[i,j,k]]) + hy[[j]]hz[[k]]ϵ[[i+1,j,k]] / (hx[[i]]ϵ[[i+1,j,k]]+hx[[i+1]]ϵ[[i,j,k]])
+ hx[[i]]hz[[k]]ϵ[[i,j-1,k]] / (hy[[j]]ϵ[[i,j-1,k]]+hy[[j-1]]ϵ[[i,j,k]]) + hx[[i]]hz[[k]]ϵ[[i,j+1,k]] / (hy[[j]]ϵ[[i,j+1,k]]+hy[[j+1]]ϵ[[i,j,k]])
+ hx[[i]]hy[[j]]ϵ[[i,j,k-1]] / (hz[[k]]ϵ[[i,j,k-1]]+hz[[k-1]]ϵ[[i,j,k]]) + hx[[i]]hy[[j]]ϵ[[i,j,k]] / (hz[[k]]ϵ[[i,j,k+1]]+hz[[k+1]]ϵ[[i,j,k]]) ),
{i, 2, imax + 1}, {j, 2, jmax + 1}, {k, 2, kmax + 1}]],

Flatten [Table [hy[[j]]hz[[k]]ϵ[[i-1,j,k]] / (hx[[i]]ϵ[[i-1,j,k]]+hx[[i-1]]ϵ[[i,j,k]]),
```

```
{i, 3, imax + 1}, {j, 2, jmax + 1}, {k, 2, kmax + 1}]],
Flatten [Table [hy[[j]]hz[[k]]ϵ[[i+1,j,k]] / (hx[[i]]ϵ[[i+1,j,k]]+hx[[i+1]]ϵ[[i,j,k]]),
{i, 2, imax}, {j, 2, jmax + 1}, {k, 2, kmax + 1}]],
Flatten [Table [hx[[i]]hz[[k]]ϵ[[i,j−1,k]] / (hy[[j]]ϵ[[i,j−1,k]]+hy[[j−1]]ϵ[[i,j,k]]),
{i, 2, imax + 1}, {j, 3, jmax + 1}, {k, 2, kmax + 1}]],
Flatten [Table [hx[[i]]hz[[k]]ϵ[[i,j+1,k]] / (hy[[j]]ϵ[[i,j+1,k]]+hy[[j+1]]ϵ[[i,j,k]]),
{i, 2, imax + 1}, {j, 2, jmax}, {k, 2, kmax + 1}]],
Flatten [Table [hx[[i]]hy[[j]]ϵ[[i,j,k−1]] / (hz[[k]]ϵ[[i,j,k−1]]+hz[[k−1]]ϵ[[i,j,k]]),
{i, 2, imax + 1}, {j, 2, jmax + 1}, {k, 3, kmax + 1}]],
Flatten [Table [hx[[i]]hy[[j]]ϵ[[i,j,k+1]] / (hz[[k]]ϵ[[i,j,k+1]]+hz[[k+1]]ϵ[[i,j,k]]),
{i, 2, imax + 1}, {j, 2, jmax + 1}, {k, 2, kmax}]]]}]];
bveccoords = Compile[{{imax, _Integer}, {jmax, _Integer}, {kmax, _Integer}},
Module[{i, j, k}, {Flatten[
Table[{{imax * (j − 2) + i − 1}, {imax * jmax * (kmax − 1) + imax * (j − 2) + i − 1}},
{i, 2, imax + 1}, {j, 2, jmax + 1}]]}]];

bvecvals = Compile[{{ϵ, _Real, 3}, {ϕ, _Real, 4}, {hx, _Real, 1}, {hy, _Real, 1},
{hz, _Real, 1}, {imax, _Integer}, {jmax, _Integer}, {kmax, _Integer},
{numconductors, _Integer}},
Module[{i, j, k, l}, {Partition[Flatten [Table [{− hx[[i]]hy[[j]]ϕ[[i,j,1,l]]ϵ[[i,j,1]] / (hz[[2]]ϵ[[i,j,1]]+hz[[1]]ϵ[[i,j,2]]),
− hx[[i]]hy[[j]]ϕ[[i,j,kmax+2,l]]ϵ[[i,j,kmax+2]] / (hz[[kmax+1]]ϵ[[i,j,kmax+2]]+hz[[kmax+2]]ϵ[[i,j,kmax+1]])},
{l, 1, numconductors}, {i, 2, imax + 1}, {j, 2, jmax + 1}]], 2 * imax * jmax]}]];
```

A.3 Empty solution

The code below calls the subroutine FindMatrix to generate the matrix and solution vectors based on a dielectric distribution ϵ (in this case it is filled with 1's). LinearSolve is then used to solve for the potential for each of the boundary voltage conditions.

```
Timing[spcoord = sparsematcoords[imax, jmax, kmax];]
Timing[spmat = sparsematvals[ϵ0, hx, hy, hz, imax, jmax, kmax];]
Timing[sparsemat = SparseArray[spcoord[[1]] → spmat[[1]],
{imax * jmax * kmax, imax * jmax * kmax}];]

Timing[bvcoord = bveccoords[imax, jmax, kmax];]
Timing[bvecv = bvecvals[ϵ0, ϕ, hx, hy, hz, imax, jmax, kmax, numconductors];]
Timing[bvec = Table[SparseArray[bvcoord[[1]] → bvecv[[1, l]], imax * jmax * kmax],
{l, 1, numconductors}];]

Timing[sol = LinearSolve[sparsemat];]
EmptySol = Table[sol[Normal[bvec[[l]]]], {l, 1, numconductors}];

Timing[EmptySolCap = Table[
Join[Transpose[Partition[Partition[EmptySol[[l]], imax * jmax][[1]], imax]],
Transpose[Partition[Partition[EmptySol[[l]], imax * jmax][[kmax]], imax]]],
{l, 1, numconductors}];]

deltaEmptySol = EmptySolCap − Phitable;

EmptyCapTable = Table[Table[
Sum[deltaEmptySol[[l, jmax/ncj(jc − 1) + j, imax/nci(ic − 1) + i]]*
voxelweighttab[[i, j]], {j, 1, jmax/ncj}, {i, 1, imax/nci}]/
(jmax/ncj * imax/nci), {jc, 1, 2 * ncj}, {ic, 1, nci}], {l, 1, numconductors}];
```

A.4 Read in real data

The code below is used for reading in actual test data when not using synthetic data.

```
SelfCapDiff = Import["/Volumes/Macintosh HD/scapdiff10-04-06.xls"];
(*SelfCapDiff =
Import["/Volumes/MNT1/My Documents/capacitive array/data new array/
data 9-20-06/scapdiff9-20-06.xls"]; *)
EnergyRef = Total[Flatten[Abs[SelfCapDiff]]]
SelfCapDiff//MatrixForm
scap = SelfCapDiff;
```

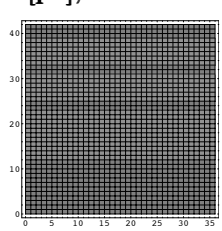
2.05597

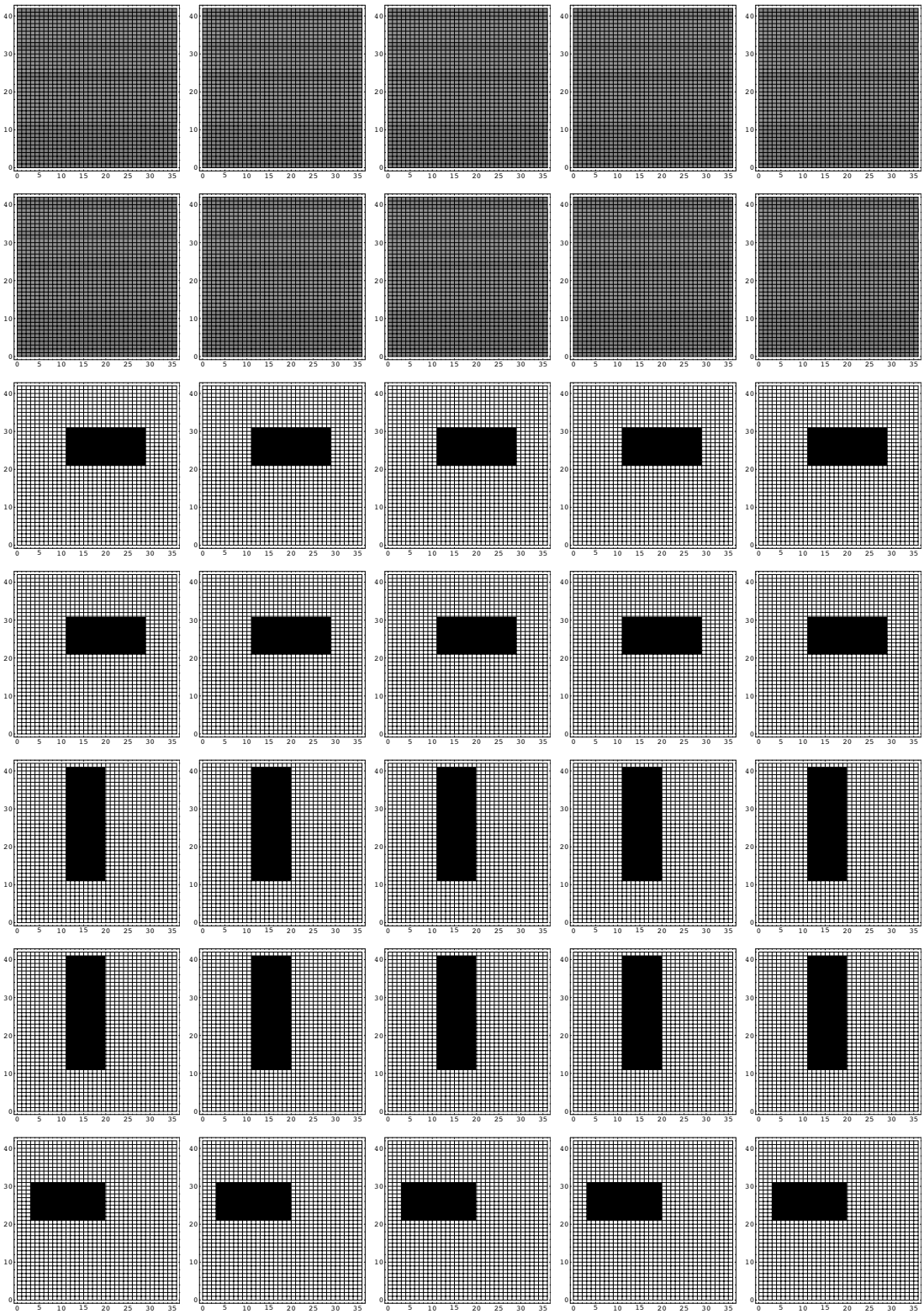
$$\begin{pmatrix} 0.000286939 & 0.00319531 & 0.0134019 & 0.00276934 \\ 0.00320271 & 0.0233191 & 0.0644303 & 0.0796496 \\ 0.0143976 & 0.0655804 & 0.114217 & 1.18195 \\ 0.00264755 & 0.0198995 & 0.0583882 & 0.0808104 \\ 0. & 0.0000734302 & 0.00383096 & 0.0000230271 \\ 0.00268501 & 0.0238399 & 0.0216018 & 0.00744121 \\ 0.022843 & 0.143914 & 0.0353009 & 0.0180004 \\ 0.00311149 & 0.023935 & 0.0159271 & 0.00529916 \end{pmatrix}$$

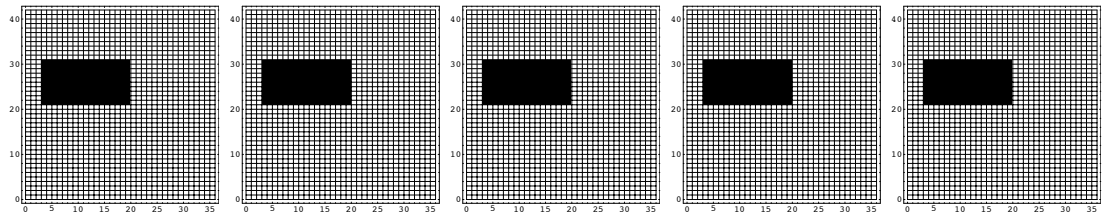
A.5 Create synthetic data

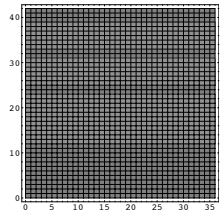
The code below is used to generate synthetic data. The coordinates must be input for the desired sample shape. The result is then plotted in two different ways. The first plot has the electrodes on both the left and right sides of the image. Each frame represents a slice of voxels in the i–k plane for each of the voxels in the j direction. The last image is an attempt at a 3D representation of the dielectric material contructed from graphics primitives.

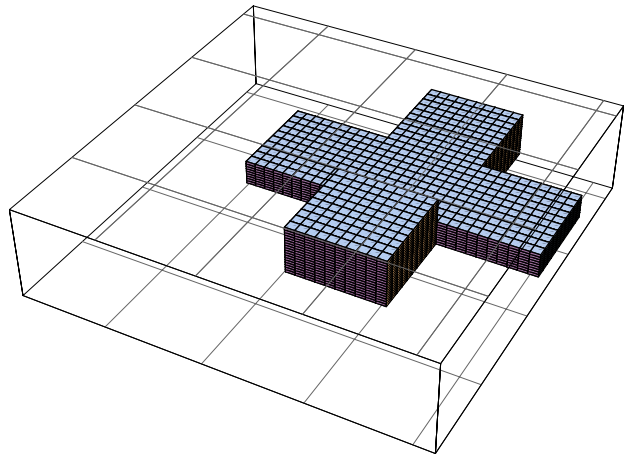
```
newcoordlist1 =
Partition[Flatten[Table[{i, j, k}, {i, 22, 31}, {j, 22, 31}, {k, 7, 24}]], 3];
newcoordlist2 =
Partition[Flatten[Table[{i, j, k}, {i, 32, 41}, {j, 12, 21}, {k, 15, 24}]], 3];
newcoordlist2a =
Partition[Flatten[Table[{i, j, k}, {i, 22, 31}, {j, 12, 21}, {k, 15, 24}]], 3];
newcoordlist2b =
Partition[Flatten[Table[{i, j, k}, {i, 12, 21}, {j, 12, 21}, {k, 15, 24}]], 3];
newcoordlist3 =
Partition[Flatten[Table[{i, j, k}, {i, 22, 31}, {j, 2, 11}, {k, 15, 32}]], 3];
newcoordlist = Join[newcoordlist1, newcoordlist2, newcoordlist2a,
newcoordlist2b, newcoordlist3];
eprofile = ReplacePart[ϵ0, diecon, newcoordlist];
electrodeDielectric =
Join[
Table[2 * eprofile[[i, j, 1]]*
eprofile[[i, j, 2]]/(eprofile[[i, j, 1]] + eprofile[[i, j, 2]]),
{i, 2, imax + 1}, {j, 2, jmax + 1}],
Table[2 * eprofile[[i, j, kmax + 1]]*
eprofile[[i, j, kmax + 2]]/
(eprofile[[i, j, kmax + 1]] + eprofile[[i, j, kmax + 2]]), {i, 2, imax + 1},
{j, 2, jmax + 1}]];

For[j = 1, j ≤ jmax + 2, j++,
ListDensityPlot[Transpose[−eprofile][[j]]]];
ecoord = Position[eprofile, diecon];
p1 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{4.098, 0.018, 0.831}];
Show[p1];
```






0 10 20 30 40
0 5 10 15 20 25 30 35


0 10 20 30 40
0 5 10 15 20 25 30 35



A.6 Calculate electrostatic potential for synthetic data

The code below is used with the synthetic data only. It calls the subroutine to generate the matrix and solution vectors based on the synthetic sample dielectric distribution, eprofile. LinearSolve is then used to solve for the potential for each of the boundary voltage conditions. RMS noise can be inserted here to test the sensitivity of the algorithm to less than perfect data.

```
Clear[sparsemat, bvec];
spmat = sparsematvals[eprofile, hx, hy, hz, imax, jmax, kmax];
sparsemat = SparseArray[spcoord[[1]] → spmat[[1]], {imax * jmax * kmax, imax * jmax
*kmax}];
bvecv = bvecvals[eprofile, ϕ, hx, hy, hz, imax, jmax, kmax, numconductors];
bvec = Table[SparseArray[bvcoord[[1]] → bvecv[[1, l]], imax * jmax * kmax],
{l, 1, numconductors}];
Timing[sol = LinearSolve[sparsemat]; ]
SampleSol = Table[sol[Normal[bvec[[l]]]], {l, 1, numconductors}];
<< Statistics`ContinuousDistributions`
noise = RandomArray[NormalDistribution[0, Sqrt[1000. * 1000.] * 10^(−6)],
{32, 8, 4}];
```

{39.6966Second, Null}

```
Mean[Flatten[noise]]
StandardDeviation[Flatten[noise]]
```

0.0000372629

0.000973555

A.7 Capacitance computations for synthetic data

The code below is used to calculate the capacitance matrix for the synthetic dielectric structure. It accomplishes this by stripping out only the boundary potentials just above the surface of the conductors and averaging them each electrode to get a number that is proportional to the change in capacitance for each electrode. DiffCapTable represents the change in capacitance (with and without dielectric). EmptyCapTable and RefCapTable are the potentials just above the conductors for both cases. To get the self capacitances, the equation has the form $(vin - \Phi)(conArea * \epsilon)/(2 * hz * vin)$. To get mutual capacitances, $(0 - \Phi)(conArea * \epsilon)/(2 * hz * vin)$.

```
Timing[RefSolCap = Table[Join[
Transpose[Partition[Partition[SampleSol[[l]], imax * jmax][[1]], imax]],
Transpose[Partition[Partition[SampleSol[[l]], imax * jmax][[kmax]], imax]]],
{l, 1, numconductors}]; ]

deltaRefSolCap = Table[(RefSolCap[[l]] − Phitable[[l]]) * electrodeDielectric,
{l, 1, numconductors}];

RefCapTable = Table[Table[
Sum[deltaRefSolCap[[l, jmax/ncj(jc − 1) + j, imax/nci(ic − 1) + i]]*
voxelweighttab[[i, j]], {j, 1, jmax/ncj}, {i, 1, imax/nci}]/
(jmax/ncj * imax/nci), {jc, 1, 2 * ncj}, {ic, 1, nci}], {l, 1, numconductors}];

DiffCapTable = −2(conArea * ϵ)/((hz[[1]] + hz[[2]]) * vin)
*(RefCapTable − EmptyCapTable);

MCapTable = ReplacePart[DiffCapTable, 0., Partition[
Flatten[Table[{{nci * (j − 1) + i, i, j}, {nci * (j − 1) + i + ncj * nci, i + nci, j}},
{i, 1, nci}, {j, 1, ncj}]], 3]];

SelfCapDiff = Join[Table[DiffCapTable[[nci * (j − 1) + i, i, j]], {i, 1, nci}, {j, 1, ncj}],
Table[DiffCapTable[[nci * (j − 1) + i + ncj * nci, i + nci, j]], {i, 1, nci}, {j, 1, ncj}]];

EnergyRef = Total[Flatten[SelfCapDiff]]

dcap = Join[Table[DiffCapTable[[k, i, j]], {j, 1, ncj}, {i, 1, nci}, {k, 1, 32}],
Table[DiffCapTable[[k, i + nci, j]], {j, 1, ncj}, {i, 1, nci}, {k, 1, 32}]];

SCposition = Position[MCapTable, 0.];
```

1.65079

A.8 Comparison with an idealized software representation

Because the code takes a week to run, it is important to evaluate the quality of the data set prior to running the algorithm. This section of code provides the means to compare actual capacitance data with the calculated capacitances from an idealized representation of the aluminum structure used for testing. Representations are shown from one of the data sets. The idealized self capacitance difference matrix is shown first, followed by the one from the actual data and the difference between the two. The last three values are the FOM, the average difference, and the rms difference for the two matrices.

```
SelfCapDiff//MatrixForm
scap//MatrixForm
scap − SelfCapDiff//MatrixForm
Total[Flatten[Abs[scap − SelfCapDiff]]]/Total[Flatten[Abs[scap]]]
Total[Flatten[Abs[scap − SelfCapDiff]]]/32
Sqrt[Total[Flatten[scap − SelfCapDiff]^2]/32]
```

$$\begin{pmatrix} 0.0000800705 & 0.00280372 & 0.0124336 & 0.00261315 \\ 0.00286209 & 0.0211619 & 0.0619209 & 0.0825696 \\ 0.0124486 & 0.0602841 & 0.112028 & 1.18925 \\ 0.00266295 & 0.0190707 & 0.0572207 & 0.0812507 \\ 0.0000760102 & 0.00133827 & 0.00434364 & 0.00117868 \\ 0.00404767 & 0.0243445 & 0.0176811 & 0.00624346 \\ 0.0221197 & 0.136193 & 0.0303785 & 0.0131022 \\ 0.00383986 & 0.0229636 & 0.0145635 & 0.00518604 \end{pmatrix}$$

$$\begin{pmatrix} 0.000286939 & 0.00319531 & 0.0134019 & 0.00276934 \\ 0.00320271 & 0.0233191 & 0.0644303 & 0.0796496 \\ 0.0143976 & 0.0655804 & 0.114217 & 1.18195 \\ 0.00264755 & 0.0198995 & 0.0583882 & 0.0808104 \\ 0. & 0.0000734302 & 0.00383096 & 0.0000230271 \\ 0.00268501 & 0.0238399 & 0.0216018 & 0.00744121 \\ 0.022843 & 0.143914 & 0.0353009 & 0.0180004 \\ 0.00311149 & 0.023935 & 0.0159271 & 0.00529916 \end{pmatrix}$$

$$\begin{pmatrix} 0.000206868 & 0.000391594 & 0.00096823 & 0.000156189 \\ 0.000340621 & 0.00215719 & 0.00250944 & -0.00292 \\ 0.00194902 & 0.0052963 & 0.00218909 & -0.00729399 \\ -0.0000154033 & 0.000828821 & 0.00116752 & -0.000440305 \\ -0.0000760102 & -0.00126484 & -0.00051268 & -0.00115565 \\ -0.00136266 & -0.00050459 & 0.00392069 & 0.00119776 \\ 0.000723311 & 0.00772112 & 0.00492238 & 0.00489818 \\ -0.000728363 & 0.000971417 & 0.0013636 & 0.000113117 \end{pmatrix}$$

0.0293131 0.00188334 0.0027764

A.9 Main subroutine

This subroutine toggles each element of space in the chosen plane above the conductors to a dielectric of ϵ_r or 1. It then calculates the difference in self-capacitance of the electrode closest to the element of space that is toggled. It also calculated the associated change in energy and a metric associated with the goodness of fit to the measured or synthetic capacitance values. This is the key routine in reconstructing the dielectric profile. The LinearSolve routine relies on the MultiFrontal method for sparse matrices.

```
SelfCapDiffComp[conCounter_, spcoord_, bvcoord_, Phitable_, systemEnergy_, SelfCapDiff_,
DiffCapTable_, MCapTable_, SCposition_, EmptyCapTable_, EnergyRef_, newerules_, nci_,
ncj_, imax_, jmax_, ϕ_, hx_, hy_, hz_, kmax_, numconductors_, EnergyCompare_]:=

Module[{testsum, i, j, ib, jb, testerule, testelectrodeDielectric, ic, jc, spmat, sparsemat,
bvecv, bvec, isol, intersol, RefSolCap1, deltaRefSolCap1, RefCapTable1, captable,
FullCapDiff, SelfCapDiff1, MCapTable1},
{
EnergyCompare = Table[systemEnergy, {i, 1, 2 * nci}, {j, 1, ncj}];
For[j = 1, j ≤ ncj, j = j + 1,
For[i = 1, i ≤ nci, i = i + 1, {
testerule = newerules;
If[conCounter[[i, j]] − conCounter[[i + nci, j]] > 0, Continue[]];
If[conCounter[[i, j]] < 15,
{
If[testerule[[imax/nci * (i − 1) + 2, jmax/ncj * (j − 1) + 2,
conCounter[[i, j]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj,
conCounter[[i, j]]]] = diecon;
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj,
conCounter[[i, j]]]] = 1.; ]]];
}];

If[conCounter[[i, j]] == 15,
{If[testerule[[imax/nci * (i − 1) + 2, jmax/ncj * (j − 1) + 2, 15]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 15]] = diecon;
```

```
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 16]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 15]] = 1.;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 16]] = 1.; }]]];
}];

If[conCounter[[i, j]] == 16,
{If[testerule[[imax/nci * (i - 1) + 2, jmax/ncj * (j - 1) + 2, 17]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 17]] = diecon;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 18]] = diecon;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 19]] = diecon;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 20]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 17]] = 1.;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 18]] = 1.;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 19]] = 1.;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 20]] = 1.; }]]];
}];

If[conCounter[[i, j]] == 17,
{If[testerule[[imax/nci * (i - 1) + 2, jmax/ncj * (j - 1) + 2, 21]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 21]] = diecon;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 22]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 21]] = 1.;
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj, 22]] = 1.; }]]];
}];

If[conCounter[[i, j]] ≥ 18,
{If[testerule[[imax/nci * (i - 1) + 2, jmax/ncj * (j - 1) + 2,
5 + conCounter[[i, j]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
```

```
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj,
5 + conCounter[[i, j]]]] = diecon;
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj,
5 + conCounter[[i, j]]]] = 1.; ]]];
}];

testelectrodeDielectric = Join[Table[2 * testerule[[i, j, 1]] * testerule[[i, j, 2]]/
(testerule[[i, j, 1]] + testerule[[i, j, 2]]), {i, 2, imax + 1}, {j, 2, jmax + 1}],
Table[2 * testerule[[i, j, kmax + 1]] * testerule[[i, j, kmax + 2]]/
(testerule[[i, j, kmax + 1]] + testerule[[i, j, kmax + 2]]), {i, 2, imax + 1}, {j, 2, jmax + 1}]];

spmat = sparsematvals[testerule, hx, hy, hz, imax, jmax, kmax];
sparsemat = SparseArray[spcoord[[1]] → spmat[[1]],
{imax * jmax * kmax, imax * jmax * kmax}];
bvecv = bvecvals[testerule, ϕ, hx, hy, hz, imax, jmax, kmax, numconductors];
bvec = Table[SparseArray[bvcoord[[1]] → bvecv[[1, l]], imax * jmax * kmax],
{l, 1, numconductors}];
(*The matrix is the same for all boundary conditions for a given dielectric profile. So,
it's more efficient to solve the matrix onetime and apply that function to the bvec that
corresponds to each boundary condition *)
isol = LinearSolve[sparsemat];

(*intersol contains the solution vectors (electrostatic pot.) for a given dielectric profile*)
intersol = Table[isol[Normal[bvec[[l]]]], {l, 1, numconductors}];

(*RefSolCap1 extracts the boundary potentials that correspond to the capacitances*)
RefSolCap1 = Table[Join[Transpose[Partition[Partition[intersol[[l]], imax * jmax][[1]],
imax]],
Transpose[Partition[Partition[intersol[[l]], imax * jmax][[kmax]], imax]]],
{l, 1, numconductors}];
deltaRefSolCap1 = Table[(RefSolCap1[[l]] - Phitable[[l]]) * testelectrodeDielectric,
{l, 1, numconductors}];

(*RefCapTable averages the potentials above each conductor in order to calculate the
change in capacitances*)
RefCapTable1 = Table[Table[Sum[
deltaRefSolCap1[[l, jmax/ncj(jc - 1) + jb, imax/nci(ic - 1) + ib]] * voxelweighttab[[ib, jb]],
{jb, 1, jmax/ncj}, {ib, 1, imax/nci}]/jmax/ncj * imax/nci), {jc, 1, 2 * ncj}, {ic, 1, nci}],
{l, 1, numconductors}];
```

```
(*FullCapDiff contains the change in capacitance between the guessed solution and
empty space. Each represents a different dielectric distribution*)
FullCapDiff = −2(conArea ∗ ϵ)/((hz[[1]] + hz[[2]]) ∗ vin) ∗ (RefCapTable1
−EmptyCapTable);
(*SelfCapDiff1 pulls out all the changes in self-capacitance for a configuration v*)
SelfCapDiff1 = Join[Table[FullCapDiff[[nci ∗ (jc − 1) + ic, ic, jc]], {ic, 1, nci}, {jc, 1, ncj}],
Table[FullCapDiff[[nci ∗ (jc − 1) + ic + ncj ∗ nci, ic + nci, jc]], {ic, 1, nci}, {jc, 1, ncj}]];

(*The best representation of the energy in the system are the self-capacitances because
 we don't know the charge induced by the dielectric on the ground plane. So,
EnergyCompare is a normalized energy comparison based on the self-capacitances.
EnergyRef is the total of thechange in self-capacitances from the measured system. *)

EnergyCompare[[i, j]] = Total[Flatten[Abs[(SelfCapDiff − SelfCapDiff1)]]]
/Total[Flatten[Abs[(SelfCapDiff)]]];
}
]];

For[j = 1, j ≤ ncj, j = j + 1,
For[i = 1, i ≤ nci, i = i + 1, {
testerule = newerules;
If[conCounter[[i, j]] − conCounter[[i + nci, j]] > 0, Continue[]];
If[conCounter[[i + nci, j]] < 15,
{If[testerule[[imax/nci ∗ (i − 1) + 2, jmax/ncj ∗ (j − 1) + 2,
conCounter[[i + nci, j]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
testerule[[imax/nci ∗ (i − 1) + 1 + ci, jmax/ncj ∗ (j − 1) + 1 + cj,
conCounter[[i + nci, j]]]] = diecon;
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
testerule[[imax/nci ∗ (i − 1) + 1 + ci, jmax/ncj ∗ (j − 1) + 1 + cj,
conCounter[[i + nci, j]]]] = 1.; ]]];
}];

If[conCounter[[i + nci, j]] == 15,
{If[testerule[[imax/nci ∗ (i − 1) + 2, jmax/ncj ∗ (j − 1) + 2, 15]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci ∗ (i − 1) + 1 + ci, jmax/ncj ∗ (j − 1) + 1 + cj, 15]] = diecon;
testerule[[imax/nci ∗ (i − 1) + 1 + ci, jmax/ncj ∗ (j − 1) + 1 + cj, 16]] = diecon; }
]],
```

```
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 15]] = 1.;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 16]] = 1.; }]]];
}];

If[conCounter[[i + nci, j]] == 16,
{If[testerule[[imax/nci * (i − 1) + 2, jmax/ncj * (j − 1) + 2, 17]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 17]] = diecon;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 18]] = diecon;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 19]] = diecon;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 20]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 17]] = 1.;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 18]] = 1.;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 19]] = 1.;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 20]] = 1.; }]]];
}];

If[conCounter[[i + nci, j]] == 17,
{
If[testerule[[imax/nci * (i − 1) + 2, jmax/ncj * (j − 1) + 2, 21]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 21]] = diecon;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 22]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 21]] = 1.;
testerule[[imax/nci * (i − 1) + 1 + ci, jmax/ncj * (j − 1) + 1 + cj, 22]] = 1.; }]]];
}];

If[conCounter[[i + nci, j]] ≥ 18,
{If[testerule[[imax/nci * (i − 1) + 2, jmax/ncj * (j − 1) + 2,
5 + conCounter[[i + nci, j]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
```

```
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj,
5 + conCounter[[i + nci, j]]]] = diecon; ]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
testerule[[imax/nci * (i - 1) + 1 + ci, jmax/ncj * (j - 1) + 1 + cj,
5 + conCounter[[i + nci, j]]]] = 1.; ]]];
}];

testelectrodeDielectric = Join[Table[2 * testerule[[i, j, 1]] * testerule[[i, j, 2]]
/(testerule[[i, j, 1]] + testerule[[i, j, 2]]), {i, 2, imax + 1}, {j, 2, jmax + 1}], Table[2
*testerule[[i, j, kmax + 1]](*testerule[[i, j, kmax + 2]]/testerule[[i, j, kmax + 1]]
+testerule[[i, j, kmax + 2]]), {i, 2, imax + 1}, {j, 2, jmax + 1}]];

spmat = sparsematvals[testerule, hx, hy, hz, imax, jmax, kmax];
sparsemat = SparseArray[spcoord[[1]] → spmat[[1]], {imax * jmax * kmax, imax * jmax
*kmax}];
bvecv = bvecvals[testerule, ϕ, hx, hy, hz, imax, jmax, kmax, numconductors];
bvec = Table[SparseArray[bvcoord[[1]] → bvecv[[1, l]], imax * jmax * kmax],
{l, 1, numconductors}];
isol = LinearSolve[sparsemat];
intersol = Table[isol[Normal[bvec[[l]]]], {l, 1, numconductors}];

RefSolCap1 =
Table[Join[Transpose[Partition[Partition[intersol[[l]], imax * jmax][[1]], imax]], Transpose[
Partition[Partition[intersol[[l]], imax * jmax][[kmax]], imax]]], {l, 1, numconductors}];

deltaRefSolCap1 = Table[(RefSolCap1[[l]] - Phitable[[l]]) * testelectrodeDielectric,
{l, 1, numconductors}];

RefCapTable1 = Table[Table[Sum[
deltaRefSolCap1[[l, jmax/ncj(jc - 1) + jb, imax/nci(ic - 1) + ib]] * voxelweighttab[[ib, jb]],
{jb, 1, jmax/ncj}, {ib, 1, imax/nci}]/(jmax/ncj * imax/nci), {jc, 1, 2 * ncj}, {ic, 1, nci}],
{l, 1, numconductors}];

FullCapDiff = -2(conArea * ϵ)/((hz[[1]] + hz[[2]]) * vin) * (RefCapTable1
-EmptyCapTable);
SelfCapDiff1 = Join[Table[FullCapDiff[[nci * (jc - 1) + ic, ic, jc]], {ic, 1, nci}, {jc, 1, ncj}],
Table[FullCapDiff[[nci * (jc - 1) + ic + ncj * nci, ic + nci, jc]], {ic, 1, nci}, {jc, 1, ncj}]];

EnergyCompare[[i + nci, j]] = Total[Flatten[Abs[(SelfCapDiff - SelfCapDiff1)]]]/
Total[Flatten[Abs[(SelfCapDiff)]]];
}]]}]
```

A.10 Starting guess initialization

This section computes initialization parameters for the starting guess - an array full of dielectric material and compares it to the data to determine the starting figure of merit, tracked in the variable systemEnergy1.

```
newerules1 = ϵ;
newelectrodeDielectric = Join[Table[2 * newerules1[[i, j, 1]]*
newerules1[[i, j, 2]]/(newerules1[[i, j, 1]] + newerules1[[i, j, 2]]), {i, 2, imax + 1},
{j, 2, jmax + 1}], Table[2 * newerules1[[i, j, kmax + 1]] * newerules1[[i, j, kmax + 2]]
/(newerules1[[i, j, kmax + 1]] + newerules1[[i, j, kmax + 2]]), {i, 2, imax + 1},
{j, 2, jmax + 1}]];

Clear[newsol, newSampleSol, newRefSolCap, newRefCapTable, newDiffCapTable,
newSelfCapDiff, systemEnergy1, sparsemat, bvec];

spmat = sparsematvals[newerules1, hx, hy, hz, imax, jmax, kmax];
sparsemat = SparseArray[spcoord[[1]] → spmat[[1]], {imax * jmax * kmax,
imax * jmax * kmax}];
bvecv = bvecvals[newerules1, ϕ, hx, hy, hz, imax, jmax, kmax, numconductors];
bvec = Table[SparseArray[bvcoord[[1]] → bvecv[[1, l]], imax * jmax * kmax],
{l, 1, numconductors}];
newsol = LinearSolve[sparsemat];
newSampleSol = Table[newsol[Normal[bvec[[l]]]], {l, 1, numconductors}];

newRefSolCap = Table[Join[Transpose[Partition[Partition[newSampleSol[[l]],
imax * jmax][[1]],
imax]], Transpose[Partition[Partition[newSampleSol[[l]], imax * jmax][[kmax]], imax]]],
{l, 1, numconductors}];

newdeltaRefSolCap = Table[(newRefSolCap[[l]] − Phitable[[l]]) * newelectrodeDielectric,
{l, 1, numconductors}];

newRefCapTable = Table[Table[Sum[
newdeltaRefSolCap[[l, jmax/ncj(jc − 1) + jb, imax/nci(ic − 1) + ib]]
*voxelweighttab[[ib, jb]], {jb, 1, jmax/ncj}, {ib, 1, imax/nci}]/(jmax/ncj * imax/nci),
{jc, 1, 2 * ncj}, {ic, 1, nci}], {l, 1, numconductors}];

newDiffCapTable = −2(conArea * ϵ)/((hz[[1]] + hz[[2]]) * vin) * (newRefCapTable
−EmptyCapTable);

newMCapTable1 = ReplacePart[newDiffCapTable, 0., Partition[
Flatten[Table[{{nci * (j − 1) + i, i, j}, {nci * (j − 1) + i + ncj * nci, i + nci, j}}, {i, 1, nci},
```

```
{j, 1, ncj}]], 3]];

newSelfCapDiff = Join[Table[newDiffCapTable[[nci * (j - 1) + i, i, j]], {i, 1, nci},
{j, 1, ncj}], Table[newDiffCapTable[[nci * (j - 1) + i + ncj * nci, i + nci, j]],
{i, 1, nci}, {j, 1, ncj}]];
selfCapDiffStart = (-SelfCapDiff + newSelfCapDiff);
systemEnergy1 = Total[Flatten[Abs[(SelfCapDiff - newSelfCapDiff)]]]
/Total[Flatten[Abs[SelfCapDiff]]]

energyArray1 = {systemEnergy1}
```

A.11 Main body of code

Insert details about the main body here.

```
index1 = 1;
looptest = 1;
While[looptest > 0,
Clear[EnergyCompare];

(*Calls subroutine that toggles each dielectric value and compares self-cap. values*)
SelfCapDiffComp[conCounter, spcoord, bvcoord, Phitable, systemEnergy1, SelfCapDiff,
DiffCapTable, MCapTable, SCposition, EmptyCapTable, EnergyRef, newerules1, nci, ncj,
imax, jmax, ϕ, hx, hy, hz, kmax, numconductors, EnergyCompare];

mettab = scaleMatrix * (systemEnergy1 − EnergyCompare);
Print["scaleMatrix=", MatrixForm[scaleMatrix]];
Print["scaleMatrix*(systemEnergy-FitMetric)=", MatrixForm[mettab]];
looptest = Total[Flatten[Abs[mettab]]];
If[looptest == 0, Break[]];
bigflattab1 = Flatten[mettab];
bigflattab = Sort[bigflattab1, Greater];
Print["bigflattab=", bigflattab];

(*This section of code tests to see if the magnitude of maximum is greater than zero.
 If not, the minimum becomes part of the solution set not to be changed*)
{maxcoord1 = Position[mettab, Min[bigflattab]];
If[maxcoord1[[1, 1]]<=nci,
conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] =
conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] + 1,
conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] =
conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] − 1];
If[conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] > 3&&
conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] < 29,
If[conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] ≤ 16,
scaleMatrix[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] =
sensMatrix[[conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] − 3,
maxcoord1[[1, 1]], maxcoord1[[1, 2]]]],
scaleMatrix[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] =
sensMatrix[[kmax − 5 − conCounter[[maxcoord1[[1, 1]],
maxcoord1[[1, 2]]]], maxcoord1[[1, 1]], maxcoord1[[1, 2]]]]],
scaleMatrix[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] = 0.;
];
Print["changing conCounter"];
```

```
Print["conCounter=", MatrixForm[conCounter]];
Continue[];
}];

maxcoord1 = Table[Position[mettab, bigflattab[[i]]][[1]], {i, 1, index1}];
Print["maxcoord1=", maxcoord1];
newSysEnergy = (systemEnergy1 − EnergyCompare)[[maxcoord1[[1, 1]],
maxcoord1[[1, 2]]]];

For[i = 1, i ≤ Length[maxcoord1], i++,
If[maxcoord1[[i, 1]] ≤ nci,
{If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] > 17,
{If[newerules1[[imax/nci ∗ (maxcoord1[[i, 1]] − 1) + 2,
jmax/ncj ∗ (maxcoord1[[i, 2]] − 1) + 2,
5 + conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
newerules1[[imax/nci ∗ (maxcoord1[[i, 1]] − 1) + 1 + ci,
jmax/ncj ∗ (maxcoord1[[i, 2]] − 1) + 1 + cj,
5 + conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] = diecon;
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
newerules1[[imax/nci ∗ (maxcoord1[[i, 1]] − 1) + 1 + ci,
jmax/ncj ∗ (maxcoord1[[i, 2]] − 1) + 1 + cj,
5 + conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] = 1.;
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] + 1;
}];

If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] == 17,
{If[newerules1[[imax/nci ∗ (maxcoord1[[i, 1]] − 1) + 2,
jmax/ncj ∗ (maxcoord1[[i, 2]] − 1) + 2, 21]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci ∗ (maxcoord1[[i, 1]] − 1) + 1 + ci,
jmax/ncj ∗ (maxcoord1[[i, 2]] − 1) + 1 + cj, 21]] = diecon;
newerules1[[imax/nci ∗ (maxcoord1[[i, 1]] − 1) + 1 + ci,
jmax/ncj ∗ (maxcoord1[[i, 2]] − 1) + 1 + cj, 22]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
```

```
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 21]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 22]] = 1.; }
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] + 1;
};

If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] == 16,
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2, 17]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 17]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 18]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 19]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 20]] = diecon;
}]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 17]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 18]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 19]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 20]] = 1.; }
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] + 1;
};

If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] == 15,
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2, 15]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
```

```
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 15]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 16]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 15]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 16]] = 1.; }
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] + 1;
}];

If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] < 15,
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2,
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj,
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] = diecon;
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj,
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] = 1.;
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] + 1;
}];
},
{
If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] < 15,
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2,
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
```

```
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj,
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]] = diecon;
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj,
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]] = 1.;
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] - 1;
}];

If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] == 15,
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2, 15]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 15]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 16]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 15]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 16]] = 1.; }
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] - 1;
}];


If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] == 16,
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2, 17]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 17]] = diecon;
```

```
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 18]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 19]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 20]] = diecon;
}]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 17]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 18]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 19]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 20]] = 1.; }
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] - 1; }];

If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] == 17,
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2, 22]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 22]] = diecon;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 21]] = diecon; }
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
{newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 22]] = 1.;
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj, 21]] = 1.; }
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] - 1;
}];

If[conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] > 17,
```

```
{If[newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 2,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 2,
5 + conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] == 1.,
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj,
5 + conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] = diecon;
]],
For[ci = 1, ci ≤ imax/nci, ci++,
For[cj = 1, cj ≤ jmax/ncj, cj++,
newerules1[[imax/nci * (maxcoord1[[i, 1]] - 1 - nci) + 1 + ci,
jmax/ncj * (maxcoord1[[i, 2]] - 1) + 1 + cj,
5 + conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]]]] = 1.;
]]];
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] =
conCounter[[maxcoord1[[i, 1]], maxcoord1[[i, 2]]]] - 1;
}];
}]];
If[conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] > 3&&
conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] < 29,
If[conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] ≤ 16,
scaleMatrix[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] =
sensMatrix[[conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] - 3,
maxcoord1[[1, 1]], maxcoord1[[1, 2]]]],
scaleMatrix[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] =
sensMatrix[[kmax - 5 - conCounter[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]],
maxcoord1[[1, 1]], maxcoord1[[1, 2]]]]],
scaleMatrix[[maxcoord1[[1, 1]], maxcoord1[[1, 2]]]] = 0.;
];
Print["conCounter=", MatrixForm[conCounter]];
newelectrodeDielectric =
Join[
Table[2 * newerules1[[i, j, 1]]*
newerules1[[i, j, 2]]/(newerules1[[i, j, 1]] + newerules1[[i, j, 2]]),
{i, 2, imax + 1}, {j, 2, jmax + 1}],
Table[2 * newerules1[[i, j, kmax + 1]]*
newerules1[[i, j, kmax + 2]]/
(newerules1[[i, j, kmax + 1]] + newerules1[[i, j, kmax + 2]]),
{i, 2, imax + 1}, {j, 2, jmax + 1}]];

systemEnergy1 = systemEnergy1 - newSysEnergy;
Print["systemEnergy1=", systemEnergy1];
```

```
energyArray1 = Append[energyArray1, systemEnergy1];
Print[energyArray1];

Put[energyArray1, "/Volumes/Macintosh HD/newselfcaptest9-20arev.mx"];
Put[newerules1, "/Volumes/Macintosh HD/newselfcaptest9-20brev.mx"];
Put[conCounter, "/Volumes/Macintosh HD/newselfcaptest9-20crev.mx"];
Put[scaleMatrix, "/Volumes/Macintosh HD/newselfcaptest9-20drev.mx"];

If[systemEnergy1 == 0, {Print["Convergence Achieved"], Break[]}];
}];
```

A.12 Image reconstruction output

Here is the code and some samples for viewing the results from the algorithm.

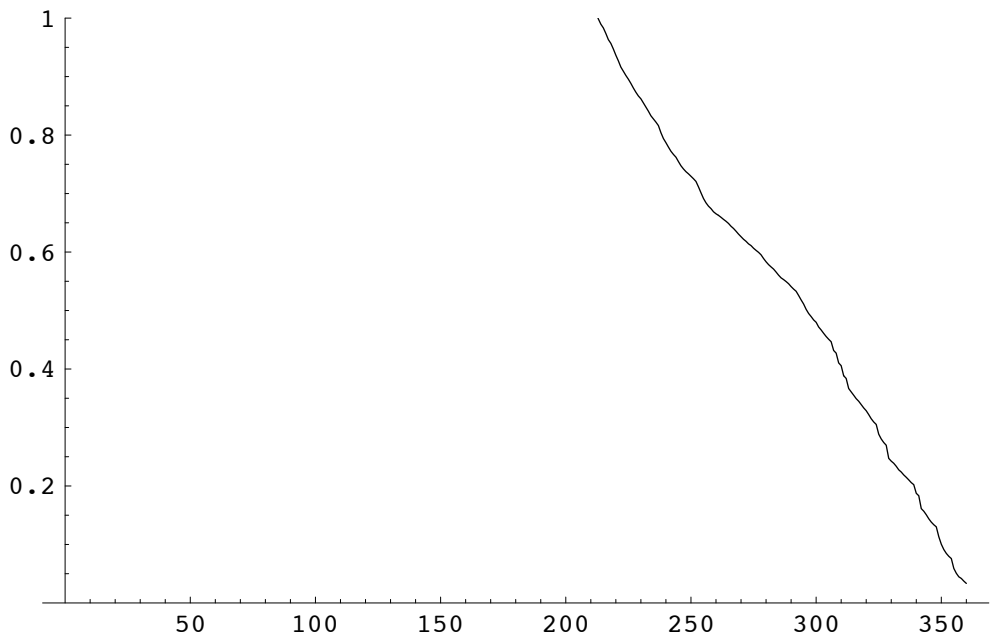
systemEnergy1

0.0334045

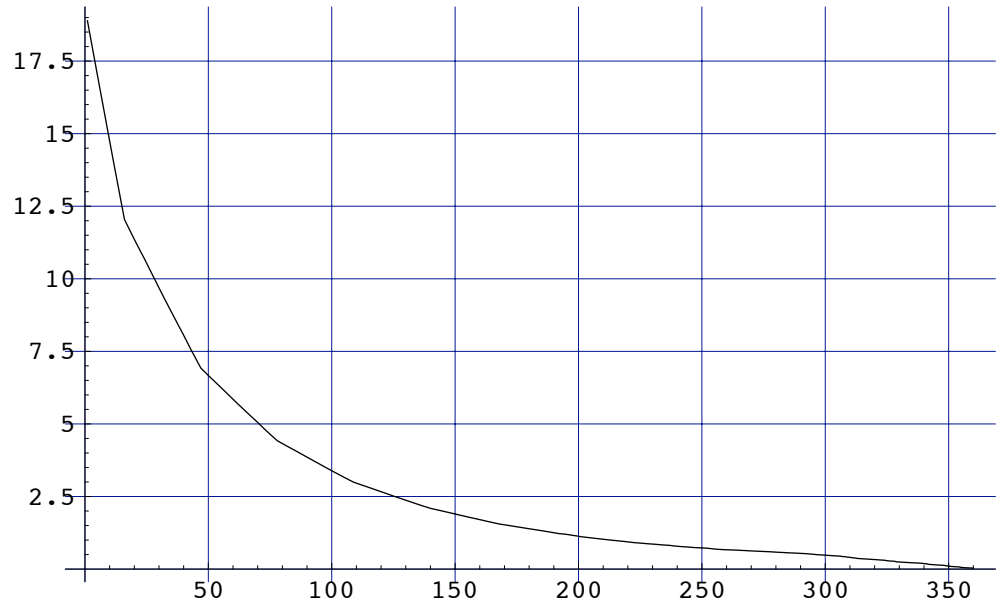
For[$j = 1, j \leq$ jmax $+ 2, j{+}{+}$,
ListDensityPlot[Transpose[−newerules1][[j]]]]

energyArray1

ListPlot[energyArray1, PlotJoined → True, PlotRange → {0, 1}];



ListPlot[energyArray1, PlotJoined → True, PlotRange → All,
GridLines → Automatic];

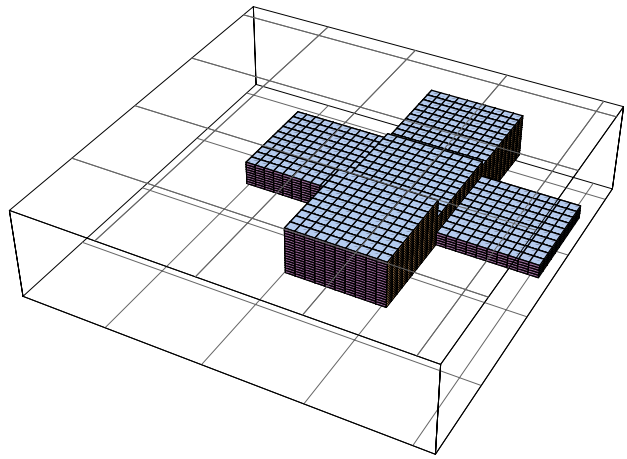



```
ListPlot[Log[energyArray1], PlotJoined → True, PlotRange → All,
GridLines → Automatic];
```

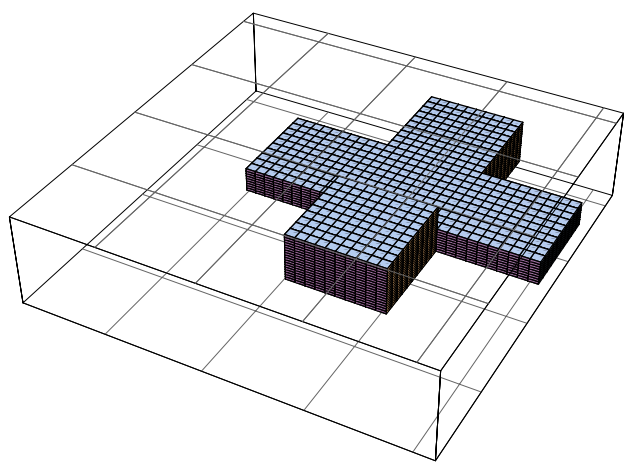

```
energydiff1 = Table[energyArray1[[i]] − energyArray1[[i + 1]],
{i, 1, Length[energyArray1] − 1}]
Print["lowest FOM=", systemEnergy1];
ecoord = Position[newerules1, diecon];
p2 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
```

```
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{1.300, −2.400, 2.000}];
Show[p2];
```

lowest FOM=0.0334045

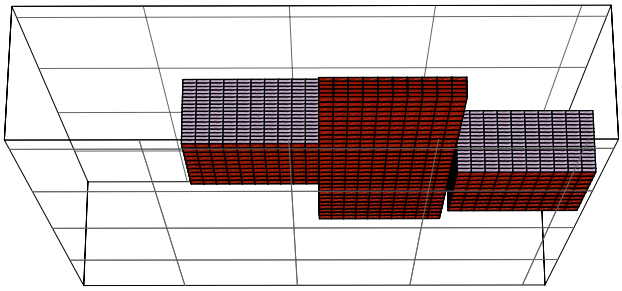


```
ecoord = Position[eprofile, diecon];
p1 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{1.300, −2.400, 2.000}];
Show[p1];
```

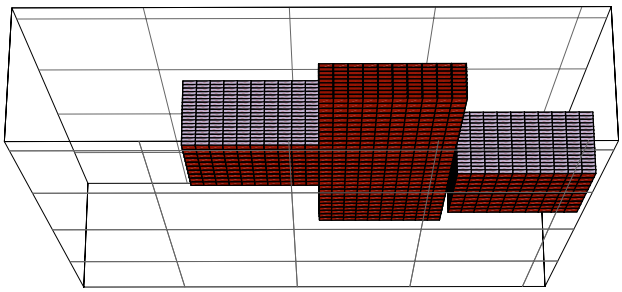


```
Print["lowest FOM=", systemEnergy1];
ecoord = Position[newerules1, diecon];
p2 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{3.225, 0.018, −1.025}];
Show[p2];
```

lowest FOM=0.0334045

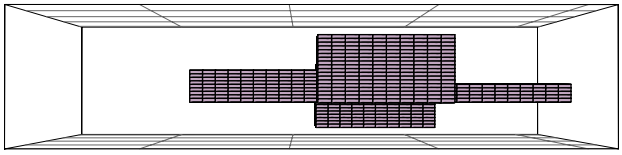


```
ecoord = Position[eprofile, diecon];
p1 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{3.225, 0.018, −1.025}];
Show[p1];
```

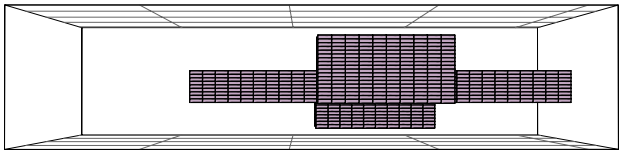


```
Print["lowest FOM=", systemEnergy1];
ecoord = Position[newerules1, diecon];
p2 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{−0.012, −3.384, −0.026}];
Show[p2];
```

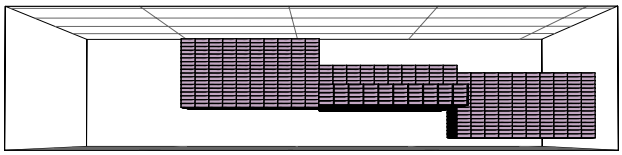
lowest FOM=0.0334045



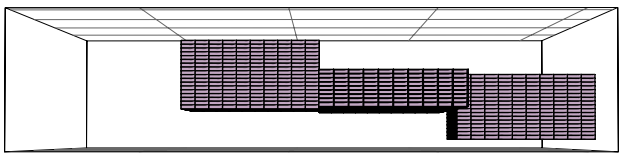
```
ecoord = Position[eprofile, diecon];
p1 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{−0.012, −3.384, −0.026}];
Show[p1];
```



```
ecoord = Position[newerules1, diecon];
p2 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{3.382, 0.019, −0.093}];
Show[p2];
```

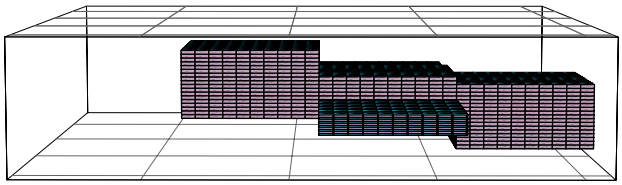


```
ecoord = Position[eprofile, diecon];
p1 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{3.382, 0.019, −0.093}];
Show[p1];
```



```
Print["lowest FOM=", systemEnergy1];
ecoord = Position[newerules1, diecon];
p2 = Graphics3D[Table[Cuboid[ecoord[[i]]], {i, Length[ecoord]}],
PlotRange → {{1, imax + 2}, {1, jmax + 2}, {1, kmax + 2}}, BoxRatios → {1, 1, .2353},
FaceGrids → {{0, 0, −1}, {0, 0, +1}}, ViewPoint->{3.369, 0.019, 0.312}];
Show[p2];
```

lowest FOM=0.0334045



LIST OF REFERENCES

[1] R. C. Youngquist. Water detection and removal from shuttle tiles. In *American Institute of Physics Conference Proceedings: Space Technology and Applications INT. Forum (STAIF 2003)*, volume 654, pages 269–275, 2003.

[2] R. C. Youngquist and M. A. Nurge. Capacitance based cyrogenic density, flow, and mass sensor work at the kennedy space center. Technical report, University of Alabama, Hunstville, May 2003.

[3] M. A. Nurge, R. C. Youngquist, and D. Walters. Capacitance based mass metering for cryogenic fluids. *Cryogenics*, 43:501–506, 2003.

[4] R. A. Williams and M. S. Beck, editors. *Process Tomography Principles, Techniques and Applications*. Butterworth-Heinemann Ltd., 1995.

[5] S. M. Huang, A. B. Plaskowski, C. G. Xie, and M. S. Beck. Tomographic imaging of two-component flow using capacitance sensors. *J. Phys. E: Sci. Instrum.*, (22):173–177, 1989.

[6] L. S. Fan, W. Warsito, Q. Marashdeh, A. H. A. Park, and B. Du. Electrical capacitance volume tomography (ecvt) for process imaging. In *2006 AIChE Spring National Meeting Conference Proceedings*, volume II, 2006. URL `http://shop.omnipress.com/index.asp?PageAction=VIEWPROD&ProdID=32&HS=1`.

[7] W. Q. Yang and L. Peng. Image reconstruction algorithms for electrical capacitance tomography. *Measurement Science and Technology*, (14):R1–R13, 2003.

[8] J. D. Jackson. *Classical Electrodynamics*. John Wiley and Sons, Inc., 3rd edition, 1998.

[9] L. D. Landau and E. M. Lifshitz. *Electrodynamics of Continuous Media*, volume 8. Butterworth-Heinemann, 2nd edition, 1984.

[10] Y. Hua, S. FuQun, X. Hui, and W. Shi. Three-dimensional analysis of electrical capacitance tomography sensing fields. *Meas. Sci. Technol.*, (10):717–725, 1999.

[11] I. N. Sneddon. *The use of integral transforms*. McGraw-Hill, 1972. ISBN 0070594368.

[12] K. F. Riley, M. P. Hobson, and S. J. Bence. *Mathematical Methods for Physics and Engineering*. Cambridge University Press, 1st edition, 1998. ISBN 0521555299. 590-591 pp.

[13] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, 6th edition, 2000.

[14] P. K. Kythe, P. Puri, and M. R. Schäferkotter. *Partial Differential Equations and Boundary Value Problems with Mathematica*. Chapman and Hall/CRC, 2nd edition, 2003.

[15] I. Kaufman. Finite volume poisson solver for ionic channels, 2004. URL `http://www.lancs.ac.uk/~kaufmani/psn/fvm.pdf`.

[16] R. V. Southwell. *Relaxation Methods In Engineering Science, A Treatise on Approximate Computation*. Oxford, 1940.

[17] G. Liebmann. Solution of partial differetial equations with a resistance network analogue. *British Journal of Applied Physics*, 1(4), April 1950.

[18] G. Liebmann. Electrical analogues. *British Journal of Applied Physics*, 4, July 1953.

[19] I. G. Bashmakova. *Diophantus and Diophantine Equations*. Math. Assoc. Amer., 1997.

[20] M. Soleimani and W. R. B. Lionheart. Nonlinear image reconstruction for electrical capacitance tomography using experimental data. *Measurement Science and Technology*, 16:1987–1996, 2005.

[21] T. Davis. Unsymmetric multi-frontal package (umfpack). URL `http://www.cise.ufl.edu/research/sparse/umfpack/`.

[22] S. M. Huang, C. G. Xie, R. Thorn, D. Snowden, and M. S. Beck. Design of sensor electronics for electrical capacitance tomography. *IEE Proceedings-G*, 139(1):83–88, February 1992.

[23] *Measure Capacitive Sensors With A Sigma-Delta Modulator*, 2005. URL `http://www.elecdesign.com/Articles/ArticleID/10185/10185.html`.

[24] Data sheet for analog devices ad7746 24-bit, 2 channel capacitance to digital converter. URL `http://www.analog.com/UploadedFiles/Data_Sheets/21450359AD7745_6_0.pdf`.

[25] P. M. Aziz, H. V. Sorensen, and J. Van Der Spiegel. An overview of sigma-delta converters. *IEEE Signal Processing Magazine*, pages 61–84, January 1996.

[26] M. Brychta. Extended capacitive range. Analog Devices Preliminary Technical Information, AD7745/46.