Numerical Scheme for Time Domain Maxwell's Equations in 3D

Math 607: Course Project

Johnathan Kuttai
11074580
jkutt@eoas.ubc.ca

Abstract

From mineral exploration to unexploded ordnance recovery to engineering, electromagnetics plays an important role in geophysics as a modelling tool. Having a numerical scheme is important to providing solutions to Maxwell's equations in order to image subsurface targets. This study will start by deriving a numerical solution to solve Maxwell's equations in the time domain for a discretized medium. Once a scheme is derived, the reliability of the scheme can be determined by evaluating the numerical errors of the to assess the stability and consistency. Analyzing the stability will help determine the convergence of the derived scheme and the best way to achieve it. Since the solution will be solved for 3 dimensions, performance is important and will be briefly discussed. The final step is to apply the numerical scheme and simulate a real word application in geophysics. The toy simulation will mimic an exploration technique using an inductive source to excite a target within a medium.

1 Introduction

Geophysics is a popular tool in many industries for imaging subsurface targets. These methods are non-invasive while providing adequate information of the subsurface. One of the most popular methods involves the application of electromagnetics. For this project we focus on the time-domain form of Maxwell's equations. Geophysics applies these principles by the use of inductive sources in the form of closed loops either on the ground or in the air **(Figure 1)**. Here we operate under a diffusive assumption of the source. A signal is transmitted via these loops often in the form of a 50% duty cycle square wave meaning that induced power for half of the transmitted cycle is cut. The purpose of this is to energize a target using the "power on" times and then record the secondary field generated by the energized target during the "power off " times of the transmitted signal **(Figure 1)**. When the power is cut, eddy currents remain according to Faraday's law which try to maintain the generated magnetic field loss in the loop. The eddy currents are attenuated according to Ohm's law. The nature of this results in a downward/outward diffusion which we measure as they change in time. This is commonly explained and visualized as a "smoke ring" propagating into the depths of the subsurface ([5]). These changes are sensitive to the contrast of the inhomogeneous subsurface property conductivity symbolized as σ. Targets that are highly conductive will hold the decaying fields much longer than surrounding resistive materials. In mineral exploration this is often associated with mineralization hosting precious metals used in electronics.

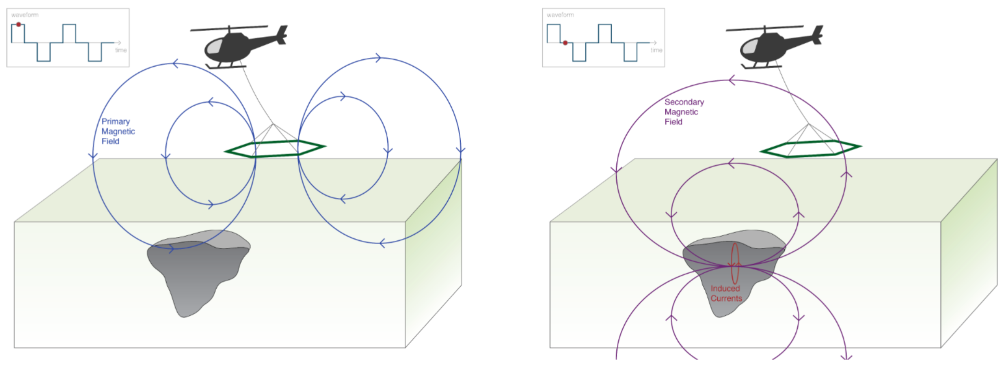


Figure 1: Schematic of an airborne time domain system (03/22 EOAS 240 lecture).

By using Maxwell's equations we can simulate this imaging technique by solving a system of complex partial differential equations (PDE's). Most commonly done is by numerically modelling the acquisition. We numerically solve because in most cases there is no analytical solution creating a simulation. Though in simple cases like a homogeneous half space we can use analytics to confirm the accuracy of our discretized numerical solution. From there we can apply the numerical scheme to more complicated sub surfaces and compute a simulation hopefully accurate enough to identify subsurface targets.

2 Time Domain Maxwell's equations

For the time domain problem discussed in the previous section, the focus will be on Faraday and Ampere's law. Our equation for Faraday's law represented as:

$$\nabla \times \mathbf{e} = -\frac{\partial \mathbf{b}}{\partial t} \tag{1}$$

and Ampere's law [4]:

$$\nabla \times \frac{\mathbf{b}}{\mu} - \sigma \mathbf{e} - \epsilon_0 \frac{\partial e}{\partial t} = j_s + j_m + j_p \tag{2}$$

However, assuming a quasi magneto-static state which neglects the electric displacement current term $\epsilon_0 \frac{\partial e}{\partial t}$ and ignore the magnetization currents and polarization's of the earth's material j_m and j_p we are left with the terms:

$$\nabla \times \frac{\mathbf{b}}{\mu} - \sigma \mathbf{e} = j_s \tag{3}$$

Given that the time domain problem for subsurface imaging the electrical conductivity σ is the physical property the simulation is sensitive to and varies more than the electrical permeability μ where we can assume $\mu = \mu_0$ everywhere. The remaining terms are the equations to solve for our numerical scheme as:

$$\nabla \times \mathbf{e} + \frac{\partial \mathbf{b}}{\partial t} = 0 \tag{4}$$

$$\nabla \times \frac{\mathbf{b}}{\mu_0} - \sigma \mathbf{e} = j_s \tag{5}$$

Now that the governing equations have been stated, we can discuss deriving the numerical scheme.

2.1 Discretization of the Mesh Operators

To create a numerical solution to our time domain problem we need to discuss the discretization of the mesh operators and the inner products of our solution grid. To solve equations (4) and (5) we need to consider 4 types of variables ([3]). The first two are scalar fields that live at cell-centers and on nodes. The remaining two are face variables assumed normal to faces and edges assigned to cell edges. This permits flux's that are in and out of faces and fields that travel a path around the surface. In our case the electric field $\vec{\mathbf{e}}$ is discretized to the cell edges while the magnetic field $\vec{\mathbf{b}}$ is discretized to faces. Finally, cell centers and nodes are where the continuous physical properties live. This is represented in **Figure 2**.

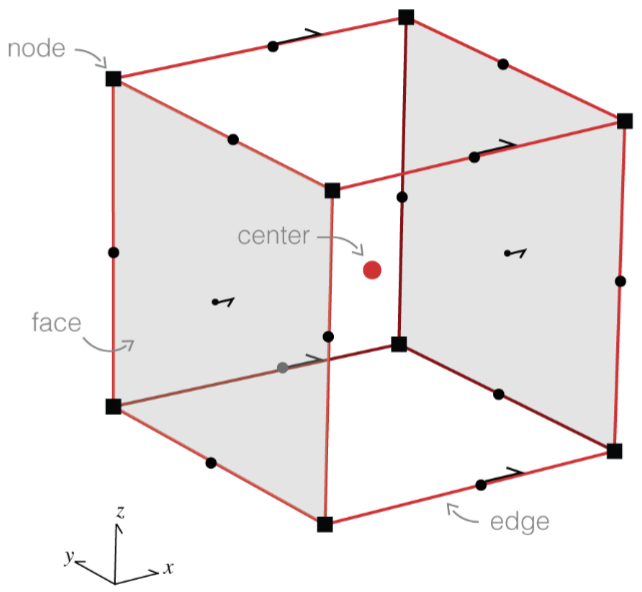


Figure 2: Discretization in 3D ([1]).

Using this we can discretize the differential operators of Maxwell's equations using their integral form. We have three main operators to discretize, though our primary equations don't require all three were derived this includes gradient, divergence and curl for a complete understanding. In order to do this we first need to consider the matrix representation of the operators which will include the use of the nested Kronecker product making up a stencil ([3]). This will be shown in more detail in the proceeding sections. In order to keep things brief, only the derivations of the discretized curl will be considered here. For the operational code a discretization object is coded to hold all parameters and the three divergence, curl and gradient operators of our discretization.

2.2 Discretization of the Curl Operator

Discretizing the curl operator starts by applying Gauss's rule and start integrating $\nabla \times \mathbf{e}$ ([3]):

$$\int_s \nabla \times \vec{e}\, ds = \oint_l \vec{e} \cdot dl \tag{6}$$

where $\mathbf{e} = (e_1, e_2, e_3)$ and s is the surface enclosed by path l such that one component of the curl $(\nabla \times e)_1$ is approximated as:

$$\frac{1}{|S|}\int_s \nabla \times \vec{e}\, ds = \frac{1}{a_{i+\frac{1}{2},j,k}}\left[(l_3e_3)_{i+\frac{1}{2},j+\frac{1}{2},k} - (l_3e_3)_{i+\frac{1}{2},j-\frac{1}{2},k} + (l_2e_2)_{i+\frac{1}{2},j,k-\frac{1}{2}} - (l_2e_2)_{i+\frac{1}{2},j,k+\frac{1}{2}}\right] \tag{7}$$

The remaining components are derived similarly by integrating on the y and z edges. By taking advantage of the nested Kronecker products and forming a stencil using a difference matrix $\mathbf{D}$ with ± 1 on the diagonals. The difference matrix D is constructed as:

$$D = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & \\ & \ddots & \ddots & \\ & & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{8}$$

Then we use the nested Kronecker products to construct the edge curl matrix as:

$$\mathbf{C} \approx \begin{bmatrix} I_3 \bigotimes I_2 \bigotimes D_1 \\ I_3 \bigotimes D_2 \bigotimes I_1 \\ D_3 \bigotimes I_2 \bigotimes I_1 \end{bmatrix} \tag{9}$$

We include the mesh geometry and parameters as matrix **F** with face areas **a** on the diagonals and **L** with edge lengths **l** on it's diagonals. This gives:

$$\mathbf{C}_{discretized} = F^{-1}CL \tag{10}$$

Notice here that we are required to do a solve for the inverse of the area of the faces. With large simulations this could be costly. In code we can construct this in the following:

```python
def getEdgeCurlMatrix(self):
        """

            calculates the curl matrix operator

        """

        n1,n2,n3 = self.n1, self.n2, self.n3

        def ddx(n):
            return sp.spdiags((np.ones((n + 1, 1)) * [1, -1]).T, [0, 1], n, n + 1, format="csr")

        nfx = (n1 + 1) * n2 * n3
        nfy = n1 * (n2 + 1) * n3
        nfz = n1 * n2 * (n3 + 1)
        nex = n1 * (n2 + 1) * (n3 + 1)
        ney = (n1 + 1) * n2 * (n3 + 1)
        nez = (n1 + 1) * (n2 + 1) * n3

        Dyz = sp.kron(ddx(n3), sp.kron(sp.eye(n2), sp.eye(n1 + 1)))
        Dzy = sp.kron(sp.eye(n3), sp.kron(ddx(n2), sp.eye(n1 + 1)))

        Dxz = sp.kron(ddx(n3),sp.kron(sp.eye(n2+1),sp.eye(n1)))
        Dzx = sp.kron(sp.eye(n3),sp.kron(sp.eye(n2 + 1), ddx(n1)))

        Dxy = sp.kron(sp.eye(n3 + 1), sp.kron(ddx(n2),sp.eye(n1)))
        Dyx = sp.kron(sp.eye(n3+1), sp.kron(sp.eye(n2), ddx(n1)))

        # curl on the edges
        Curl = sp.vstack(
            [sp.hstack([sp.dia_matrix((nfx,nex)), Dyz, -Dzy]),
             sp.hstack([-Dxz, sp.dia_matrix((nfy,ney)), Dzx]),
             sp.hstack([Dxy, -Dyx  , sp.dia_matrix((nfz,nez))])
             ]
        )

        return Curl
```

Then do the solve with code to get the curl discretization operator:

```python
# use scipy's spsolve to get the curl with mesh geometry
CURL = spsolve(F,(CURL@L))
```

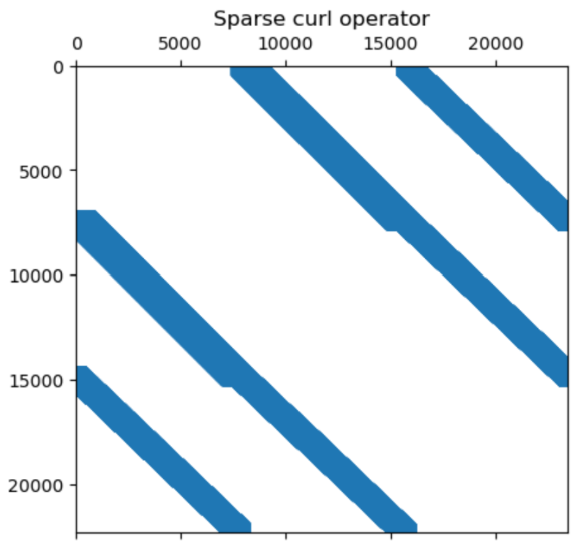


Figure 3: Plot of the curl operator diagonals.

We can again visualize the operator as plotted in **Figure 3**.

2.3 Discretization of the Inner products

Here we have chosen the e-b formulation and will take it one step further by representing equations (4) and (5) in their weak form so that we can define our finite volume discretization. This is done by taking the inner product of (4) and (5) with faces and edges space functions f and w over our defined domain. Finally we integrate by parts Ampere's law.

$$\int_\Omega (\nabla \times \mathbf{e}) \cdot \mathbf{f}\, dv - \int_\Omega \frac{\partial \mathbf{b}}{\partial t} \cdot \mathbf{f}\, dv = 0$$

$$\int_\Omega \frac{\mathbf{b}}{\mu_0} \cdot (\nabla \times \mathbf{w}) dv - \int_{\partial\Omega} \frac{\mathbf{w}}{\mu_0} \cdot (\mathbf{b} \times \hat{n}) ds - \int_\Omega \sigma \mathbf{e} \cdot \mathbf{w}\, dv = \int_\Omega j_s \cdot \mathbf{w}\, ds$$

Since b is divergence free as stated by Gauss's law and assuming natural boundary conditions $\mathbf{b} \times \hat{n} = 0 \in \partial\Omega$ we can cancel the boundary term. We use this to form our equations for a finite volume discretization described by the following ([2] [4]):

$$C_{discretized}\, \mathbf{e} + \frac{\partial \mathbf{b}}{\partial t} = 0 \tag{11}$$

$$C^T_{discretized} M^f_{\mu^{-1}} \mathbf{b} - M^e_\sigma \mathbf{e} = M^e_\sigma j_s \tag{12}$$

Where $C_{discretized}$ represents our curl operator we derived. M^e_σ is the edge inner product matrix for physical property σ and $M^f_{\mu^{-1}}$ is the face inner product for property μ^{-1}. We now need to derive the discretized operators for our inner products to complete the numerical scheme. To do this we will use the midpoint method to average our edge and face values to the cell centered physical property.

2.4 Face inner product

To get our face inner product we again utilize the use of the nested Kronecker products and use the midpoint method as our averaging technique. Then we include the physical property and the mesh geometry, specifically the volume to construct our $M^f_{\mu^{-1}}$. To accomplish this a 1D averaging diagonal matrix is constructed; one for each direction where the main diagonal and positive diagonal will be filled with $\frac{1}{2}, \frac{1}{2}$ values as described in (13).

$$A_{ave1D} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & \\ & \ddots & \ddots & \\ & & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{13}$$

To construct the averaging of the face to the cell center matrix in 3D we use the nested Kronecker products to give:

$$\mathbf{A}_{face2cell} \approx \begin{bmatrix} I_3 \bigotimes I_2 \bigotimes A_x \\ I_3 \bigotimes A_y \bigotimes I_1 \\ A_z \bigotimes I_2 \bigotimes I_1 \end{bmatrix} \tag{14}$$

This is accomplished by coding a function that computes and returns (14) in the following:

```python
def getFaceToCellCenterMatrix(self):

        n1,n2,n3 = self.n1, self.n2, self.n3

        def av(n):
            return sp.spdiags((np.ones([n+1,1])*np.array([0.5,0.5])).T,[0,1],n,n+1)

        A1 = sp.kron(sp.eye(n3),sp.kron(sp.eye(n2),av(n1)))
        A2 = sp.kron(sp.eye(n3),sp.kron(av(n2),sp.eye(n1)))
        A3 = sp.kron(av(n3),sp.kron(sp.eye(n2),sp.eye(n1)))

        # average from faces to cell-centers
        Afc = sp.hstack([A1, A2, A3])

        return Afc
```

Using the output of this function the inner product matrix is constructed by including our physical property μ^{-1} and mesh volumes as follows:

```python
M_muinv_f  = sp.diags(Afc.T*(Volumes*(1./mu)))
```

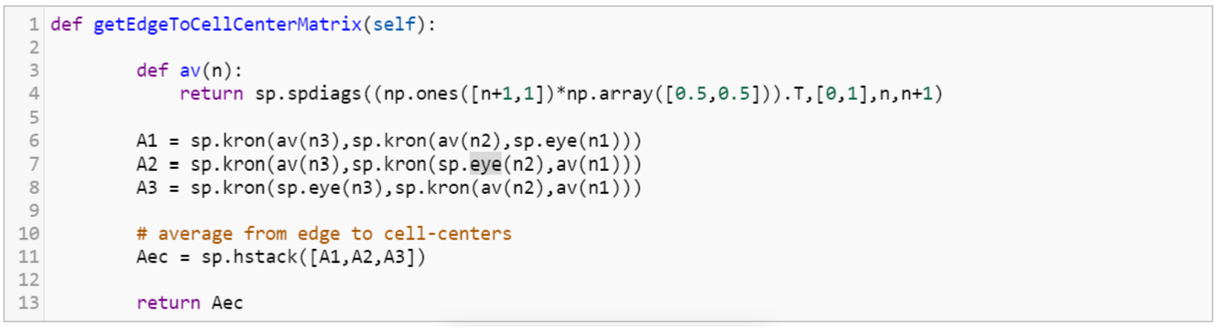
This results in our discretized inner product matrix for faces to cell centers.

2.5 Edge inner product

Constructing the edge inner products follows similar to our face inner product matrix by using nested Kronecker products and an averaging mechanism that uses the midpoint method (13). Again, we then incorporate our physical property and mesh geometry, specifically the volume, to construct M^e_σ. Though, this time our edge to cell center matrix is constructed as described in (15).

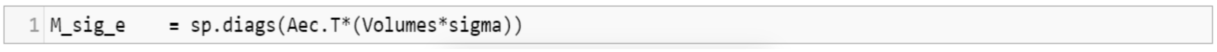
$$\mathbf{A}_{edge2cell} \approx \begin{bmatrix} A_z \bigotimes A_y \bigotimes I_1 \\ A_z \bigotimes I_2 \bigotimes A_x \\ I_3 \bigotimes A_y \bigotimes A_x \end{bmatrix} \tag{15}$$

This is accomplished in code by the following:



```python
def getEdgeToCellCenterMatrix(self):

        def av(n):
            return sp.spdiags((np.ones([n+1,1])*np.array([0.5,0.5])).T,[0,1],n,n+1)

        A1 = sp.kron(av(n3),sp.kron(av(n2),sp.eye(n1)))
        A2 = sp.kron(av(n3),sp.kron(sp.eye(n2),av(n1)))
        A3 = sp.kron(sp.eye(n3),sp.kron(av(n2),av(n1)))

        # average from edge to cell-centers
        Aec = sp.hstack([A1,A2,A3])

        return Aec
```

Then follows the final construction by including the volume and physical property σ to get:



```python
M_sig_e     = sp.diags(Aec.T*(Volumes*sigma))
```

We now have a discretized inner product for our edges to cell centers.

3 Simulation

Now that all the tools required for the numerical scheme are constructed we can now use equations (11) and (12) to set up a time domain simulation. First we isolate **e** in equation (12) so that it can be substituted into (11) giving us a linear partial differential equation (PDE) we can solve to get **b**.

$$\mathbf{e} = (M^e_\sigma)^{-1} C^T_{discretized} M^f_{\mu^{-1}} \mathbf{b} - (M^e_\sigma)^{-1} M^e_\sigma j_s \tag{16}$$

Since we are only interested in the time after the power on, the j_s term is therefore zero. This leaves us with:

$$\mathbf{e} = (M^e_\sigma)^{-1} C^T_{discretized} M^f_{\mu^{-1}} \mathbf{b} \tag{17}$$

Now equation (12) is :

$$\frac{\partial \mathbf{b}}{\partial t} + C_{discretized}\, (M^e_\sigma)^{-1} C^T_{discretized} M^f_{\mu^{-1}} \mathbf{b} = 0 \tag{18}$$

with initial condition $\mathbf{b}(0) = \nabla \times \phi$ where ϕ is the vector potential response from the transmitter loop's power on part of the duty cycle. Here we compute $\mathbf{b}(0)$ from an analytic solution ([6]) for the vector potential then use the discretized curl operator to perform $\nabla \times A = \mathbf{b}$.

The PDE is now fully formed and we can discretize in time by using backward Euler to solve.

$$\frac{\mathbf{b}^{t+1} - \mathbf{b}^t}{\Delta t} + C_{discretized}\, (M^e_\sigma)^{-1} C^T_{discretized} M^f_{\mu^{-1}} \mathbf{b}^t = 0 \tag{19}$$

$$\mathbf{b}^{t+1} = (1 - \Delta t\, C_{discretized}\, (M^e_\sigma)^{-1} C^T_{discretized} M^f_{\mu^{-1}}) \mathbf{b}^t \tag{20}$$

In code we provide the backward Euler function for solving at each time step by the following:

```python
#Backward Euler function
def Backward_Euler(u0, A, time):
    """
    u_{t+1} = u_t + k*f(u_{t+1})

    """
    u_list=[u0]
    u_prev=u0

    for i in range(len(time)):
        k = time[i][0]

        Id = sp.eye(A.shape[0],A.shape[1])

        Ainv = Pardiso(Id-k*A)
        print(f'solving for time step: {k}')

        for j in range(time[i][1]):
            u1 = Ainv*uprev
            u_list.append(u1)
            u_prev=u1

    return u_list
```

3.1 Stability

Backward Euler has a stability region consisting of the entire left side of the complex plain which makes it particularly good at stiff problems. We can determine our formulation is stiff by analyzing to (4) and (5) without the source term and isolating b and doing a substitution to find that:

$$\frac{\partial \mathbf{b}}{\partial t} + \nabla \times \left(\frac{1}{\sigma} \nabla \times \frac{1}{\mu} \mathbf{b} \right) = 0 \tag{21}$$

and using the identity $\nabla \times (\nabla \times \mathbf{b}) = \nabla(\nabla \cdot \mathbf{b}) - \nabla^2 \mathbf{b}$ but we have $\nabla \cdot \mathbf{b} = 0$ which leaves us with:

$$\frac{\partial \mathbf{b}}{\partial t} = \frac{1}{\mu\sigma} \nabla^2 \mathbf{b} \tag{22}$$

This resembles the heat equation. If we assume 1D and a homogeneous whole space and uniform cell size and move to a discrete form:

$$\frac{\mathbf{b}_n^{t+1} - \mathbf{b}_n^t}{\Delta t} = \frac{1}{\mu\sigma} \frac{\mathbf{b}_{n+1}^t - 2\mathbf{b}_n^t + \mathbf{b}_{n-1}^t}{\Delta h^2} \tag{23}$$

Using (23) we can derive that we need time step restriction of:

$$\Delta_t \leq \frac{\mu\sigma\Delta h^2}{2} \tag{24}$$

In the case of earth's materials, typical values of μ are on the order of $1 \times 10^{-8} H/m$ and σ on the orders of 1×10^1 to $1 \times 10^{-4} S/m$. Include that we also discretize at cell size of 10m we require a time step $\Delta t \leq 1 \times 10^{-7} seconds$. Therefore the time domain problem becomes very stiff. Since explicit methods are mostly not effective for stiff problems we require the use of implicit methods. In our case we use the backward Euler method as it is unconditionally stable ([3]). This is only first order accurate though. Methods like the midpoint method are second order accurate and cost the same in implementation. However, when Δt is large we would get $\mathbf{b}^{n+1} \approx -\mathbf{b}^n$ causing oscillations at each time step and losing the second order accuracy ([3]). Therefore we will focus only on the backward Euler method.

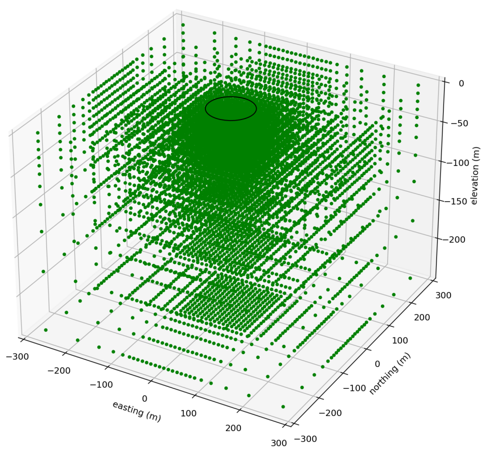


Figure 4: Simulation setup with a current carrying loop at surface and active nodes of mesh displayed.

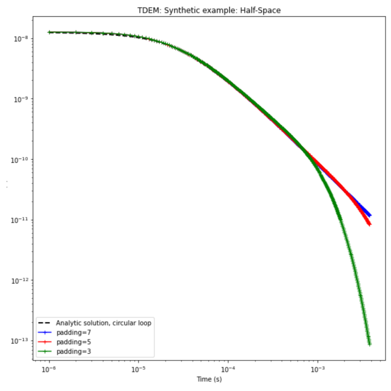


Figure 5: Effects of padding on the late time solutions.

3.2 Solving a homogeneous half-space Simulation

Now that all the required pieces are in place, we can use the discretized curl operator and inner product matrices to create our system matrix that will be solved for each time step using the backward Euler method. First we set up our simulation with a mesh that has cells of 10 m x 10 m x 10 m cell size mesh at its core. For the main analysis we choose 5 padding cells in each direction at a growth of 1.5x the previous pad cell. From our stability analysis we choose timesteps on the order of 1 $\times 10^{-7} seconds$ and solve for times spanning 1×10^{-7} to 1×10^{-3} seconds after the power is turned off. We choose this span because at very early times ($t \leq 1 \times 10^{-7}$) our quasi magneto-static assumption does not hold and Maxwell's wave-like behaviour would need to be considered. For the purposes of geophysical exploration we are primarily concerned with late times anyhow. The source for the simulation is a circular loop of radius 50 m centered at x=0, y=0, z=0 and an assumed current of 1 Amp **(Figure 4)**. A receiver will be used to access the numerical scheme at the center of this loop for optimal coupling. From here we designate our homogeneous half space as material in cells at and below 0 m elevation and anything greater than 0 m elevation is considered "air" cells. For simplicity, we will analyze the z component of the magnetic field as it is the strongest signal and the primary component used for interpretation in practice.

First lets consider the padding cells that we added. We can see in **Figure 5** the effect of padding cells on the latest time estimates. For better accuracy at late times we can see that 7 padding cells is more ideal. This is primarily due to the discretization being large enough to compensate for the boundary conditions term that was neglected.

With the padding for our half-space at 7 padding cells we can compare to the analytic solution to see how well the numerical scheme is performing. In **Figure 6** it is shown that the biggest differences are in the earliest times while we get good comparison from mid to late times. This is ideally what the goal is for targeting purposes.

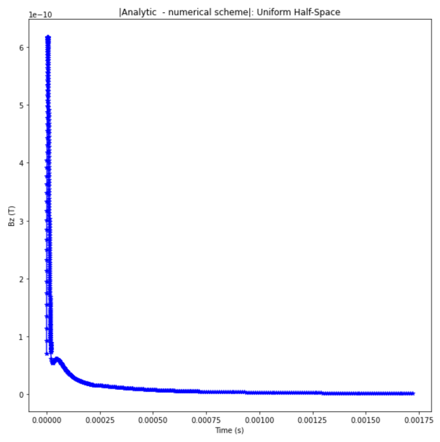


Figure 6: Effects of padding on the late time solutions.

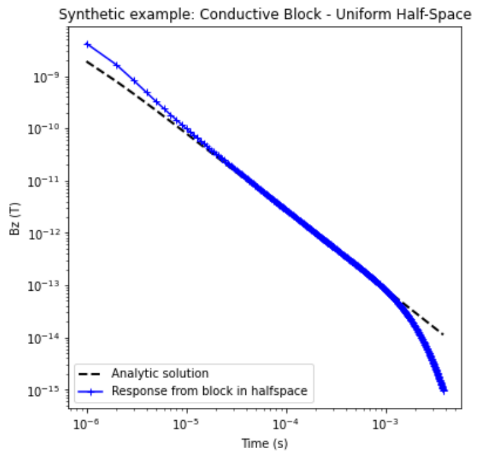


Figure 7: Effect of a conductive block within a uniform half-space.

3.3 Simulation of a conductive block in a uniform half-space

Now we take a look at the response from a synthetic example where we have a buried conductive block target of $\sigma = 1S/m$ in a resistive background of $\sigma_{background} = 0.001S/m$. The square block will be centered beneath the loop at a depth of 10 m and length, width, height of 40 m x 40 m x 40 m. The effects of the block at depth is compared to the expected response from uniform half-space in **Figure 7**. The effects are evident in early times, while the late time looks to be corrupted.

4 Conclusion

By using Maxwell's equations (4) and (5) in their weak form, a numerical scheme to simulate the physics of a geophysical time domain problem can be devised. The scheme is quite sensitive though to initialization and discretization. We saw the effect of the padding cells alone can greatly affect the solution for the latest times. We are also forced to use implicit schemes because of the stiffness of the problem due to the requirement of our time step derived from stability analysis. Luckily though, there exists methods to handle this like the backward Euler method.

Further study trying different integration techniques, cell sizes, source types could provide useful insight into further design of the numerical scheme. As well as exploring other implicit methods to solve our system. The family of BDF methods help damp higher frequencies that give us the error in our magneto-quasi-static assumption. Perhaps exploring Crank-Nicolson implicit methods could be useful here too.

Numerical schemes like the one we devised are important for many tasks in geophysics. It can be used to plan surveys by giving the user an idea of what to expect from a geophysics time-domain survey. It can also be used as the forward kernel in a geophysical inversion scheme where the collected data is used to produce the most plausible model of the subsurface given parameters and constraints. A stable scheme is certainly desired.

References

[1] Cockett, Rowan, Lindsey J. Heagy, and Douglas W. Oldenburg. 'Pixels and Their Neighbors: Finite Volume'. The Leading Edge 35, no. 8 (August 2016): 703–6. https://doi.org/10.1190/tle35080703.1.

[2] Cockett, Rowan, Seogi Kang, Lindsey J. Heagy, Adam Pidlisecky, and Douglas W. Oldenburg. 'SimPEG: An Open Source Framework for Simulation and Gradient Based Parameter Estimation in Geophysical Applications'. Computers & Geosciences 85 (December 2015): 142–54. https://doi.org/10.1016/j.cageo.2015.09.015.

[3] Haber, Eldad. Computational Methods in Geophysical Electromagnetics. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014. https://doi.org/10.1137/1.9781611973808.

[4] Heagy, Lindsey J., Rowan Cockett, Seogi Kang, Gudni K. Rosenkjaer, and Douglas W. Oldenburg. 'A Framework for Simulation and Inversion in Electromagnetics'. Computers & Geosciences 107 (October 2017): 1–19. https://doi.org/10.1016/j.cageo.2017.06.018.

[5] Nabighian, M., and Macnae, J. (1991) "Time domain electromagnetic prospecting methods, in Electromagnetic Methods in Applied Geophysics" Volume 2: Application, Part A, chapter six,M. N. Nabighian (ed.), Society of Exploration Geophysicists, Tulsa, OK.

[6] Simpson, James C., John Lane, Christopher Darby Immer, Robert C. Youngquist and Todd Steinrock. "Simple Analytic Expressions for the Magnetic Field of a Circular Current Loop." (2001).