Survey of Solution Techniques for Linear Systems from Finite Difference Methods in 2D Numerical Radiative Transfer

Advanced Numerical Analysis II

Final Project

Oliver Evans
Fred Weiss
Christopher Parker
Emmanuel Arkoh

Dr. Malena Español

May 13, 2017

1 Introduction

We use monochromatic radiative transfer in order to model the light field in an aqueous environment populated by vegetation. The vegetation (kelp) is modeled by a spatial probability distribution, which we assume to be given. The two quantities we seek to compute are *radiance* and *irradiance*. Radiance is the intensity of light in at a particular point in a particular direction, while irradiance is the total light intensity at a point in space, integrated over all angles. The Radiative Transfer Equation is an integro-partial differential equation for radiance, which has been used primarily in stellar astrophysics; it's application to marine biology is fairly recent [12].

We study various methods for solving the system of linear equations resulting from discretizing the Radiative Transfer Equation. In particular, we consider direct methods, stationary iterative methods, and nonstationary iterative methods. Numerical experiments are performed using Python's `scipy.sparse` [11] package for sparse linear algebra. `IPython` [15] was used for interactive numerical experimentation.

Among those implemented, the nonstationary `LGMRES` [4] algorithm is the only algorithm determined to be suitable for this application without further work. We discuss limitations and potential improvements, including preconditioning, alternative discretization, and reformulation of the RTE.

1.1 Radiative Transfer

Let n be the number of spatial dimensions for the problem (i.e., 2 or 3). Let $x \in \mathbb{R}^n$. Let Ω be the unit sphere in $\mathbb{R}^n$. Let $\omega \in \Omega$ be a unit vector in $\mathbb{R}^n$. Let $L(x, \omega)$ denote *radiance* position x in the direction ω. Let $I(x)$ denote *irradiance* at position x. Let $P_k(x)$ be the probability density of kelp at position x. Let a_w and a_k be the absorption coefficients of water and kelp respectively. Let b_w and b_k be the scattering coefficients of water and kelp respectively. Then we define the effective absorption and scattering coefficients as

$$a(x) = P_k(x)a_k + (1 - P_k(x))a_w \tag{1}$$
$$b(x) = P_k(x)b_k + (1 - P_k(x))b_w \tag{2}$$

Then, the Monochromatic Time-Independent Radiative Transfer Equation (RTE) is

$$\omega \cdot \nabla_x L(x, \omega) = -(a(x) + b(x))L(x, \omega) + b \int_\Omega \beta(\omega \cdot \omega')L(x, \omega')\, d\omega' \tag{RTE}$$

Note that in 2 spatial dimensions, this is a 3-dimensional problem (x, y, θ). Likewise, in 3 spatial dimensions, it is a 5-dimensional problem (x, y, z, θ, ϕ).

In this paper, we consider only the 2-dimensional problem, with the hope that sufficiently robust solution techniques for the 2-dimensional problem will be effective in the solution of the 3-dimensional problem as well.

1.2 2D Problem

We use the downward-pointing coordinate system shown in figure 1, measuring $\theta \in [0, 2\pi)$ from the positive x axis towards the positive y axis. Further, we assume that the problem is given on the rescaled spatial domain $[0, 1) \times [0, 1)$, where $y = 0$ is the air-water interface, and y measures depth from the surface.

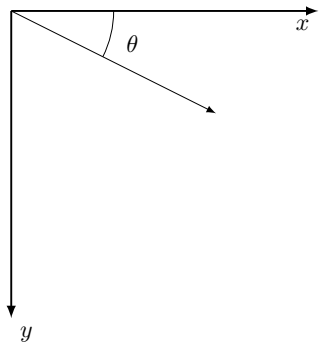

Figure 1: 2D coordinate system

The 2-dimensional form of (RTE) is given by

$$\frac{\partial L}{\partial x}\cos\theta + \frac{\partial L}{\partial y}\sin\theta = -(a + b)L(x, y, \theta) + b\int_0^{2\pi} \beta(|\theta - \theta'|)\, d\theta', \tag{3}$$

where $|\theta - \theta'|$ measures the smallest angular difference between θ and θ' considering periodicity.

Note that in Cartesian coordinates, there are only spatial, not angular derivatives in the gradient. In other coordinate systems, this is generally not the case.

1.3 Boundary Conditions

We assume that the downwelling light from the surface is known, and is defined to be uniform in space by the Dirichlet boundary condition

$$L(x, 0, \theta) = f(\theta), \quad \text{for} \quad \theta \in [0, \pi). \tag{4}$$

Note that we cannot apply the same idea to upwelling light at the surface, as it cannot be specified from information about the atmospheric light field. Therefore, we apply the PDE at $y = 0$ for $\theta \in [\pi, 2\pi)$.

At $y = 1$, we assume no upwelling light. That is,

$$L(x, 0, \theta) = 0, \quad \text{for} \quad \theta \in [\pi, 2\pi). \tag{5}$$

As with the upper y-boundary, we apply the PDE for $\theta \in [0, \pi)$ so as not to prohibit downwelling light.

In the horizontal direction, we assume periodic boundary conditions. Assuming that a single discrete group of plants is being simulated, adjusting the width of the domain effectively modifies the spacing between adjacent groups of plants.

2 System of Linear Equations

2.1 Discretization

In order to solve (3) numerically, we discretize the spatial derivatives using 2nd order finite difference approximations, and we discretize the integral according to the Legendre-Gauss quadrature, as described in Chandrasekhar [5, Chapter 2]. With this in mind, in order to create a spatial-angular grid with n_x, n_y, and n_θ discrete values for x, y, and θ respectively, we use a uniform square spatial discretization with spacing dx, dy, and a non-uniform angular discretization according to the roots of the Legendre Polynomial of degree n_θ, denoted $P_{n_\theta}(\theta)$. When considering square spatial grids, we denote the number of spatial grid points in each dimension by $n_s = n_x = n_y$ and the number of angular grid points by $n_a = n_\theta$. In each variable, we discard the uppermost grid point, as indicated by the half-open intervals in the previous sections.

Then, we have the grid

$$x_i = (i-1)\,dx, \qquad i = 1, \ldots, n_x \tag{6}$$
$$y_j = (j-1)\,dy, \qquad j = 1, \ldots, n_y \tag{7}$$
$$\theta_k \text{ s.t. } P_{n_\theta}(\theta_k/\pi - 1) = 0, \qquad k = 1, \ldots, n_\theta \tag{8}$$

In the same notational vein, let

$$L^k_{ij} = L(x_i, y_j, \theta_k), \tag{9}$$
$$\beta_{kl} = \beta(|\theta_k - \theta_l|), \tag{10}$$
$$a_{ij} = a(x, y) \tag{11}$$
$$b_{ij} = b(x, y) \tag{12}$$

where $|\cdot|$ is periodic as in (3).

For the spatial interior of the domain, we use the 2nd order central difference formula (CD2) to approximate the derivatives, which is

$$f'(x) = \frac{f(x+dx) - f(x-dx)}{2dx} + \mathcal{O}(dx^3). \tag{CD2}$$

When applying the PDE on the upper or lower boundary, we use the forward and backward difference (FD2 and BD2) formulas respectively. Omitting $\mathcal{O}(dx^3)$, we have

$$f'(x) = \frac{-3f(x) + 2f(x+dx) - f(x+2dx)}{2dx} \tag{FD2}$$

$$f'(x) = \frac{3f(x) - 2f(x-dx) + f(x-2dx)}{2dx} \tag{BD2}$$

As for the angular integral, we substitute a weighted finite sum of the function evaluated at the angular grid points. For each k, let a_k be the appropriate Legendre-Gauss weight according to Chandrasekhar [5, Chapter 2]. Then, applying the change of variables theorem to transform from the standard quadrature interval $[0, 1]$ to the correct angular interval $[0, 2\pi]$, we have

$$\int_0^{2\pi} f(\theta)\,d\theta \approx \pi \sum_{k=1}^{n} a_k f(\theta_k) \tag{LG}$$

2.2 Difference Equation

Given the above discrete approximations, the difference equation for (RTE) in the spatial interior of the domain is

$$\begin{aligned} 0 = {} & \frac{1}{2dx}\left(L^k_{i+1,j} - L^k_{i-1,j}\right)\cos\theta_k - \pi b \sum_{\substack{l=1 \\ l\neq k}}^{n_\theta} a_l \beta_{kl} L^l_{ij} \\ & + \frac{1}{2dy}\left(L^k_{i,j+1} - L^k_{i,j-1}\right)\sin\theta_k + (a_{ij} + b_{ij})L^k_{ij} \end{aligned} \tag{13}$$

Similarly, we discretize using (FD2) and (BD2) at the boundaries. For example, when $j = 1$, we have

$$\begin{aligned} 0 = {} & \frac{1}{2dx}\left(L^k_{i+1,j} - L^k_{i-1,j}\right)\cos\theta_k - \pi b \sum_{\substack{l=1 \\ l\neq k}}^{n_\theta} a_l \beta_{kl} L^l_{ij} \\ & + \frac{1}{2dy}\left(4L^k_{i,j+1} - L^k_{i,j+2}\right)\sin\theta_k + (a_{ij} + b_{ij} - \frac{3}{2dy})L^k_{ij} \end{aligned} \tag{14}$$

Note that when discretizing the integral, we exclude the $l = k$ term of the sum. This is because that term corresponds to "scattering" straight ahead ($\Delta\theta = 0$), which is in fact not scattering at all. Whether some adjustment to the quadrature is necessary to account for this is unclear.

2.3 Structure of Linear System

For each (i, j, k), we have a distinct equation with $4 + n_\theta$ variables. This corresponds to a sparse matrix equation $Ax = b$, each row having $4 + n_\theta$ nonzero entries. Note that b is zero at each row except those which correspond to boundary conditions in y.

Each element in x and b correspond to a particular triple (i, j, k), as are each row and column of the coefficient matrix A. In some sense, when we create this linear system, we are unwrapping a 3-dimensional quantity (radiance) into a 1-dimensional vector (the solution, x). Different orders in which the equations are listed can be chosen to generate equivalent systems, so long as the ordering is consistent in A, b, and x.

The most obvious way to order the equations is to do so via a triple `for` loop, which has the effect of creating blocks in the matrix corresponding to planes in (x, y, θ) space. For example, if the equations are ordered such that the outer `for` loop is over x, and the inner two are over y and θ, then the first $(n_y n_\theta)$ rows of the matrix correspond to the equations for $i = 1$.

By switching the order of the `for` loops, we can generate 6 equivalent matrix systems, each of which can be identified with a permutation of the triple $(0, 1, 2)$, where 0 corresponds to x, 1 corresponds to y, and 2 corresponds to θ, and the order of the numbers indicates the order in which the loops were nested, from outer to inner.

The coefficient matrices A, generated by each of these choices are shown below for a trivially small system with $n_x = n_y = n_\theta = 6$. Each black square represents a nonzero entry, whereas gray lines are purely for visual aid.

2.4 Sparsity Plots

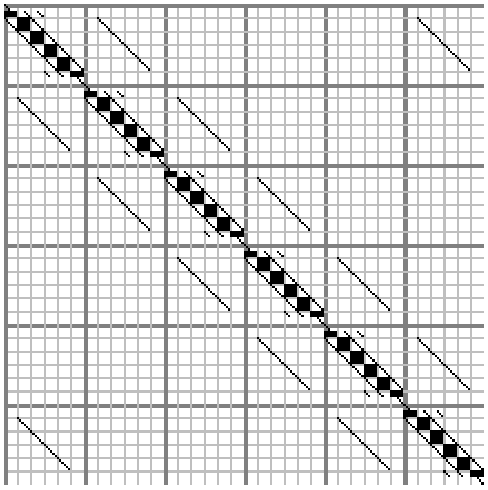

Figure 2: Sparsity plot: 6x6x6, ordering 012

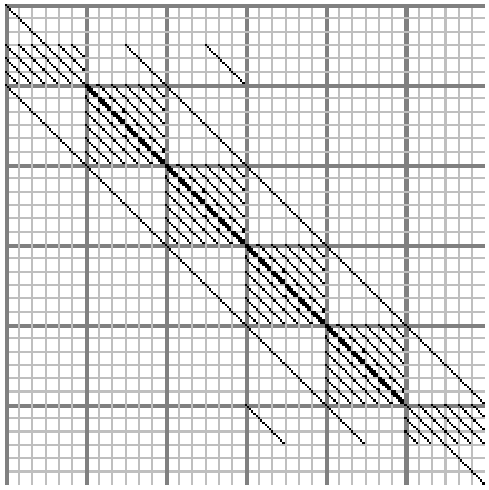

Figure 5: Sparsity plot: 6x6x6, ordering 120

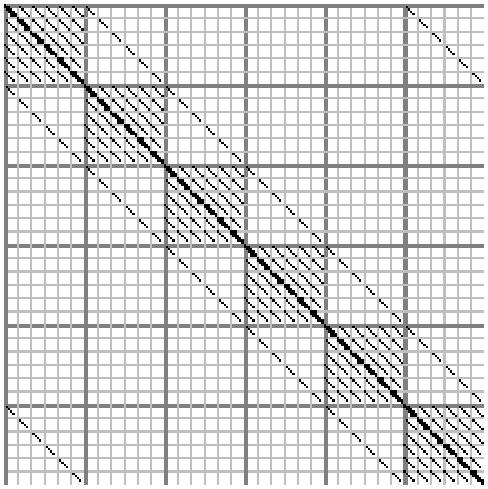

Figure 3: Sparsity plot: 6x6x6, ordering 021

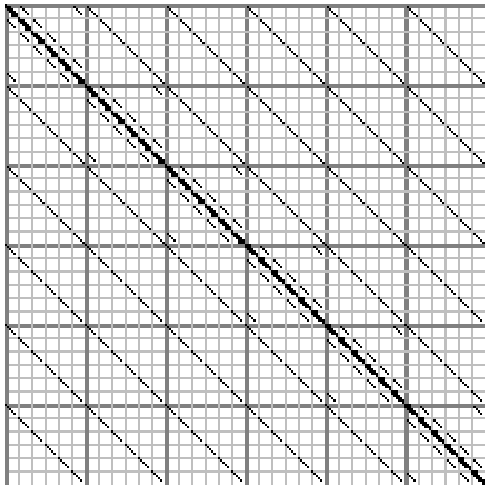

Figure 6: Sparsity plot: 6x6x6, ordering 201

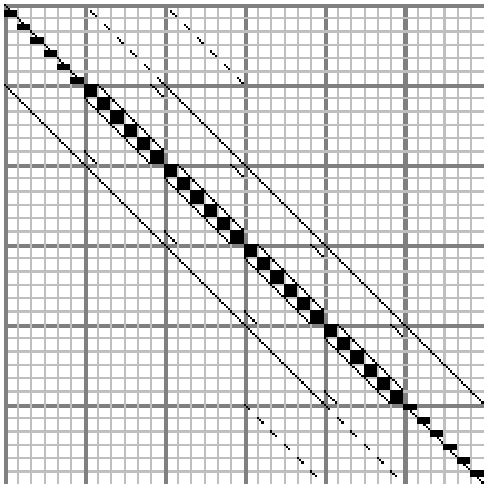

Figure 4: Sparsity plot: 6x6x6, ordering 102

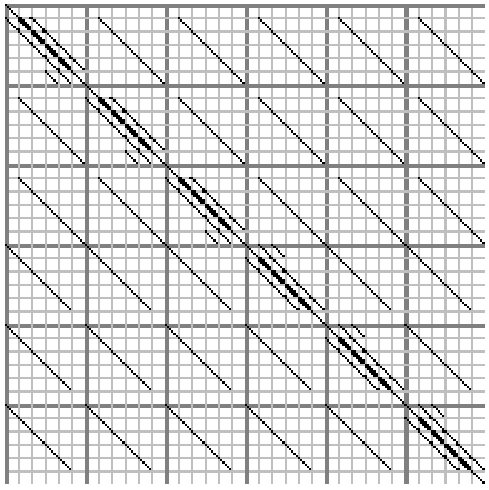

Figure 7: Sparsity plot: 6x6x6, ordering 210

2.5 Physical Setup

In the upcoming sections, all numerical experiments are based on the following parameters.

We define the probability density of kelp as

$$P_k(x,y) = \begin{cases} 2-y & \text{if } |x-.5| \le 5(1-y)^2\exp(5y-4) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

We use the normalized volume scattering function defined by

$$\beta(\theta) = \frac{\exp(-\theta/2)}{2\left(1-\exp(-\theta/2)\right)} \tag{16}$$

While this captures the general shape of the VSF of highly scattering water, in practice, we should either model the VSF as a function of parameters, perhaps using Mie Scattering Theory [13], or else interpolate field data from water similar to the water in the region of interest, such as Petzold [16].

As a surface boundary condition, we use $L(x,0,\theta) = \sin(\theta)$ for $\theta \in [0,\pi)$, which represents a wide angular distribution of incoming light, centered in the downward direction.

For this set of parameters, the kelp distribution P_k and the highest resolution solution obtained using the methods described below is shown.

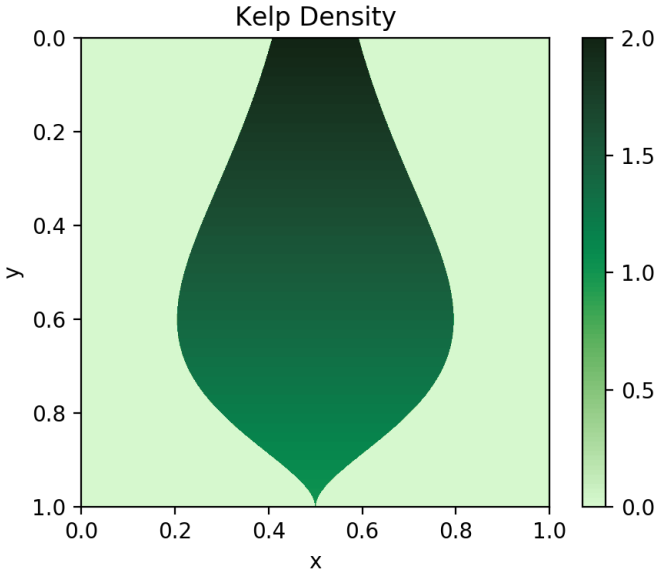

Figure 8: Kelp density P_k for numerical experiments

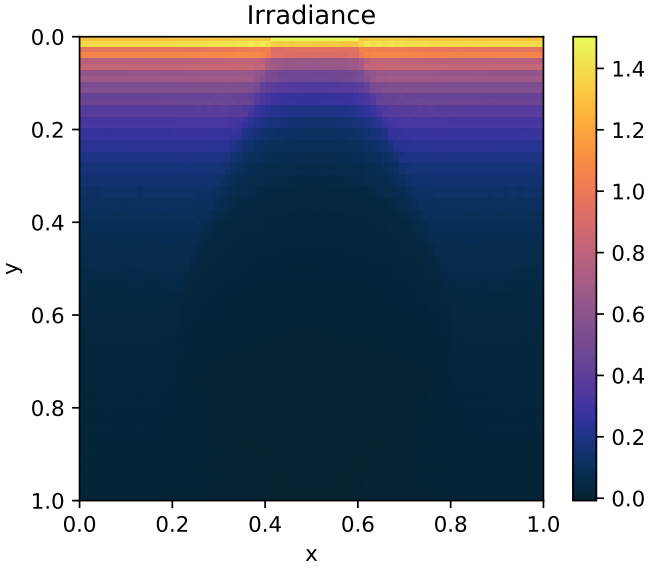

Figure 9: Irradiance plot for $n_s = 80, n_a = 80$ using `LGMRES`

2.6 Matrix Properties

There are several properties which a real matrix can have which simplify or accelerate the solution procedure. Simply from the sparsity patters, we see that none of the above matrices are symmetric. A symmetric matrix can additionally be positive definite, in which case numerous highly efficient algorithms exist for solving the system, such as the Conjugate Gradient method [14]. Another important property is diagonal dominance.

2.6.1 Diagonal Dominance

A matrix $A = (a_{ij})$ is said to be diagonally dominant if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \text{ for all } i \tag{17}$$

For each row i of the matrix, define the quantity

$$\mathcal{D}_i = \left| a_{ii} - \sum_{i \neq j} |a_{ij}| \right| \tag{18}$$

Then, A is diagonally dominant if and only if

$$\min_{1 \le i \le n} \mathcal{D}_i > 0. \tag{19}$$

Consider the row of the matrix corresponding to an arbitrary interior point, from (13). Then, we have

$$D = a_{ij} + b_{ij} - \frac{\cos\theta_k}{dx} - \frac{\sin\theta_k}{dy} - \pi b_{ij} \sum_{\substack{l=1 \\ l \neq k}}^{n_\theta} a_l \beta_{kl} L_{ij}^l \tag{20}$$

Hence, we wonder whether we can reasonably restrict ourselves to values for a and b which generate diagonal dominance. We attempt to bound D from below. First, we must state that

$$\sum_{\substack{l=1 \\ l\neq k}}^{n_\theta} a_l \beta_{kl} \leq \sum_{l=1}^{n_\theta} a_l \beta_{kl}$$

$\because \beta, a_i \geq 0$

$$\approx \int_0^{2\pi} \beta(|\theta - \theta_k|)\, d\theta$$

by Legendre-Gauss quadrature

$$= \int_0^{2\pi} \beta(|\theta|)\, d\theta$$

$\because \beta(|\cdot|)$ is π-periodic

$$= 2\int_0^{\pi} \beta(\theta)\, d\theta$$

$\because \beta(\cdot)$ is even

$$= 2$$

$\because \beta$ is normalized on $[0, \pi]$

Then, we can bound D from below as follows.

$$D \geq a_{ij} + b_{ij} - \frac{1}{dx} - \frac{1}{dy} - 2\pi b_{ij} \tag{21}$$

$$= a_{ij} - (2\pi - 1)b_{ij} - \frac{1}{dx} - \frac{1}{dy} \tag{22}$$

$$= a_w - (2\pi - 1)b_k - \frac{1}{dx} - \frac{1}{dy} \tag{23}$$

$$:= \mathcal{Q} \tag{24}$$

since, by (1) and (2), $a_{ij} \geq a_w$ and $b_{ij} \leq b_k \, \forall i, j$, assuming that $a_k \geq a_w$ and $b_k \geq b_k$.

If we assume that $dx = dy = ds$, then $\mathcal{Q} \geq 0$ is equivalent to

$$a_w - (2\pi - 1)b_k \geq \frac{2}{ds} = 2n_s.$$

For simplicity, we take $a_w = 2\pi b_k$. Then, we can be sure that $D \geq 0$ if we take

$$b_k = 4n_s. \tag{25}$$

Then, for example, with $n_s = 20$, we take $b_k = 80$, and $a_w = 160\pi$. For simplicity, we choose $b_w = 80$, and $a_k = 2a_w = 320\pi$. Clearly, these coefficients are orders of magnitude higher than the ones we chose in section 2.5.

We verify, however, that the system *is*, in fact, diagonally dominant under these conditions, and iterative methods are successful. However, obtaining a numerical solution and plotting the irradiance, we have the following disappointing picture.

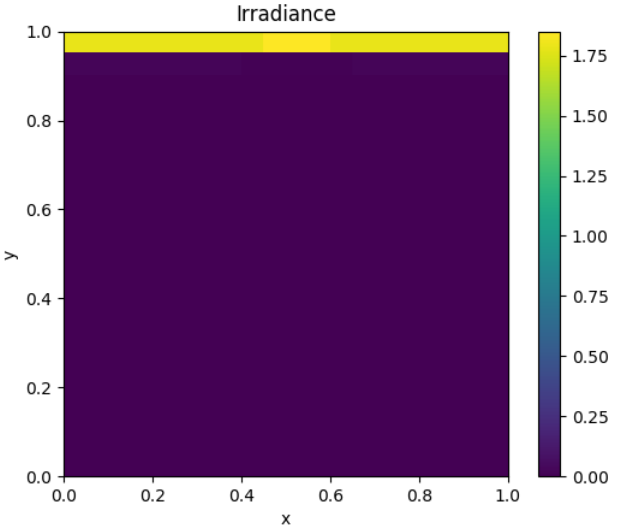

Figure 10: Diagonally dominant systems are very dark.

As we can see, we are not interested in the small subset of cases for which the coefficient matrix is diagonally dominant, as they require huge absorption and scattering coefficients, far beyond what one would expect from a realistic system.

3 Direct Methods

3.1 Factorizations

The most straightforward way to solve any linear system is via Gaussian Elimination, or equivalently, by performing an LU factorization. However, the number of arithmetic operations required for full LU factorization algorithm grows as $\mathcal{O}(n^3)$, making it impractical for most large systems. However, specific sparse algorithms exist which discard the zero elements, drastically reducing the computational cost. Similarly, other factorizations which one might generally use to solve a dense system, such as QR, SVD, and Cholesky have analogous sparse algorithms which ignore all zero elements.

Once a factorization has been computed, solving the system is extremely cheap. This makes factorization particularly appealing in cases where it is necessary to solve numerous systems with the same coefficient matrix A, but for different right-hand side vectors b. This occurs in particular for linear time-dependent PDEs. However, since we are considering the time-independent RTE, that appeal is lost on us.

Another major limitation of factorizations is the large amount of memory they require. Fundamentally, factorizations require the storage of multiple factors, which need not necessarily preserve the sparsity pattern of the original matrix. This limitation is somewhat avoidable with out-of-core computations, whereby matrices are stored directly on the hard drive rather than in memory. Computations directly on disk, however, are anywhere from six times to 10^5 times slower than computations in memory [10].

3.2 Software Packages

Many implementations of direct methods exist for particular circumstances. `LAPACK` [3], the well known free and open source `FORTRAN` linear algebra library, contains subroutines for banded, triangular, symmetric, tridiagonal, and general matrices. Of those, the banded solver is potentially applicable to the system we consider. A variety of more sophisticated free and open source packages for solving sparse systems directly exist, such as `UMFPACK` [6] and `MUMPS` [1]. Both of these algorithms in particular are *multi-frontal*, which is to say that they only keep small slices (fronts) of the matrix in memory at any given time, and perform computations on several fronts in parallel, drastically reducing solution time.

3.3 Numerical comparison

We briefly compare computational time between standard LU, QR, and SVD factorization algorithms, and then show the dramatic speed increase achieved by sparse algorithms.

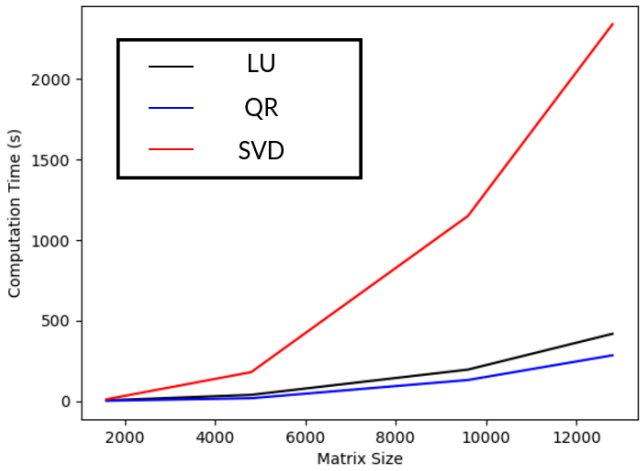

Figure 11: Standard factorization algorithms applied to sparse matrices are slow.

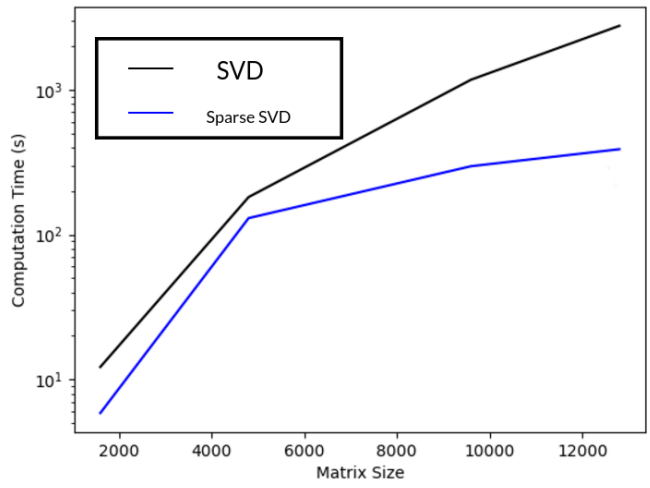

Figure 12: Sparse-specific algorithms perform much better.

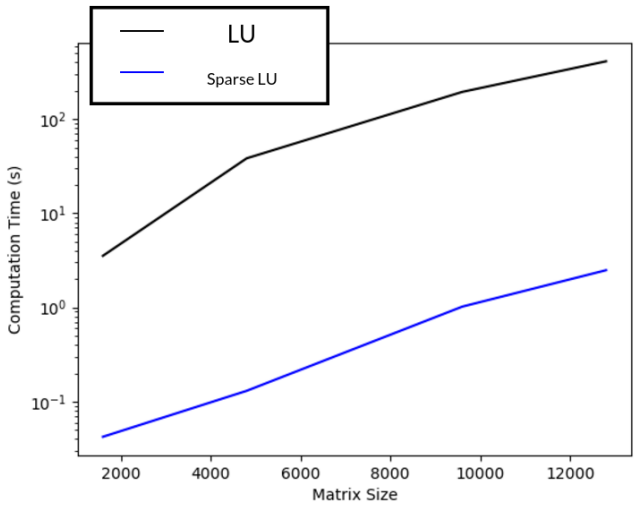

Figure 13: Note that sparse LU is two orders of magnitude faster than sparse QR.

4 Stationary Iterative Methods

4.1 Fixed-Point Iteration

Iterative methods are another way in which to solve linear systems of equations. Iterative methods work to successively improve upon their approximate solution, stopping when the results are calculated to within a tolerance which can be set as desired. Iterative methods are a generally good option for solving a linear system $Ax = b$ particularly when A is large. Since these systems meet that criteria, this was a path that was desirable to explore to see how these results compared.

We follow Anderson [2, Section 1.1] in describing basic Stationary Iterative Methods operate as follows. Given $Ax = b$, we choose N and P such that

$$A = N - P, \tag{26}$$

where N is invertible.

Then,

$$\begin{aligned} Ax &= b \\ (N - P)x &= b \\ Nx &= Px + b \end{aligned}$$

Extending the idea of fixed point iteration to systems, we take the two instances of x in the above equation to be consecutive iterates, and obtain

$$x^{(i+1)} = N^{-1}Px^{(i)} + N^{-1}b \tag{27}$$

Letting $G = N^{-1}P$ and $C = N^{-1}b$, we have the general form of stationary iterative methods.

$$x^{(i+1)} = Gx^{(i)} + C \tag{28}$$

The choice of N and P in the above equations determine the particular iterative method. In fact, given that the two are related by (26), the choice of N uniquely defines a method.

The most common methods are listed here, where D, L, and U represent the diagonal, strictly lower triangular, and strictly upper triangular pieces of A respectively, I is the identity matrix, ω and p are free parameters for which the optimal value is problem-dependent.

Method	N
Jacobi	D
JOR	$\frac{1}{\omega}D$
Gauss-Seidel	$D + L$
SOR	$\frac{1}{\omega}D + L$
Richardson	$-\frac{1}{p}I$

Figure 14: Basic Linear Stationary Iterative Methods [17]

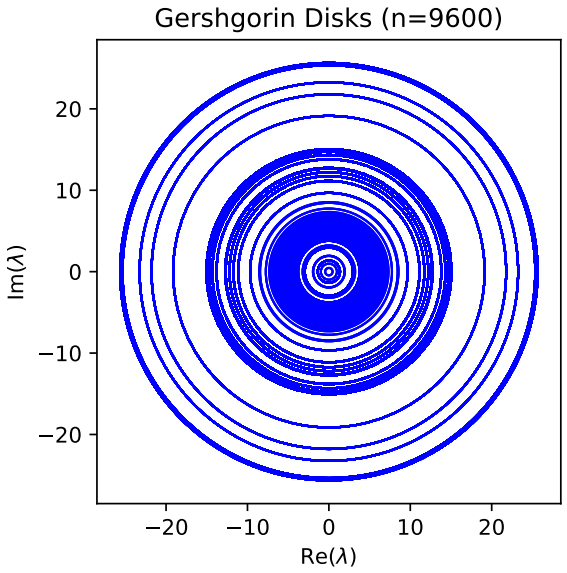

Figure 15: Gershgorin disks for Jacobi method for standard system

4.2 Convergence (or lack thereof)

It is a well-known result [2] that the iterative method defined by (28) converges to the exact solution x if and only if $\rho(G) < 1$, where $\rho(G)$ is the *spectral radius* of G, defined as the largest magnitude of an eigenvalue of G. A sufficient but unnecessary condition for convergence of (28) is the diagonal dominance of A.

As we observed in section 2.6.1, A is in general not diagonally dominant for this problem. Further, we show that it's eigenvalues need not be bounded in the unit circle.

4.2.1 Gershgorin Disks

We can bound the location of eigenvalues in the complex plane using *Gershgorin's Circle Theorem* [7], which states that if we define $D(x_0, r)$ as the disk of radius r centered at x_0 in the complex plane, then for each eigenvalue λ of a matrix $A = (a_{ij})$ with dimension N,

$$\lambda \in \bigcup_{i=1}^{N} D(a_{ii}, R_i), \tag{29}$$

where

$$R_i = \sum_{\substack{j=1 \\ j \neq i}} a_i j. \tag{30}$$

For a given iterative method and it's corresponding iteration matrix G, we can use this theorem to bound $\rho(G)$. We show plots for both the standard system considered in this paper, as well as the forcibly diagonally dominant system described in section 2.6.1 for the Jacobi method.

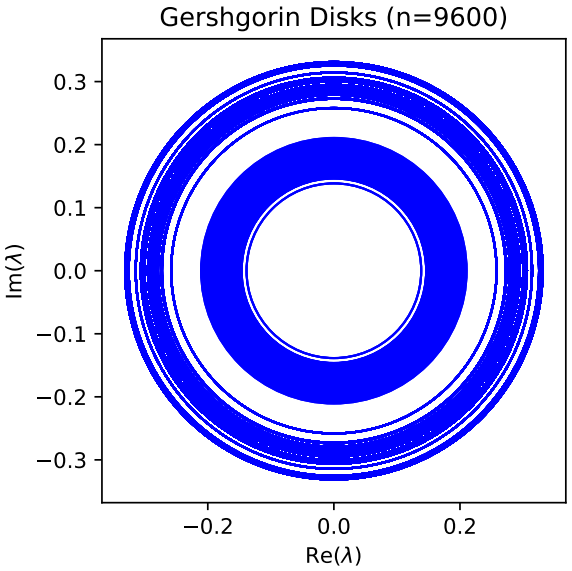

Figure 16: Gershgorin disks for Jacobi method for diagonally dominant system

Note that the original system may have eigenvalues outside of the unit circle, while the modified system must have all eigenvalues within the unit circle. This demonstrates, as shown previously, that iterative methods will fail in the former case and succeed in the latter case.

4.2.2 Preconditioning

Although the matrix is not naturally suited for stationary iterative methods, it is possible to precondition the matrix in order to induce diagonal dominance. We consider two such preconditioning strategies.

The first option is the computation of the SVD of A, using U and V^T as preconditioners. However, given the large computational cost of SVD and the lack of readily available efficient sparse algorithms, this was deemed infeasible.

We did, though, explore the method of Jacobi iterations discussed in Yuan and Yalamov [18] to identify and gradually reduce the off-diagonal terms. This serves to make the system increasingly diagonally dominant until becoming solvable with Stationary Iterative Methods.

Our implementation of the Jacobi iteration algorithm successfully preconditioned the matrix corresponding to a 10×16 spatial-angular grid for diagonal dominance, however the time required for preconditioning was on the order of hours, whereas the time required to solve directly or by a nonstationary iterative method was on the order of seconds. Clearly, this is an inappropriate approach for our problem. Nonetheless, after preconditioning, both Jacobi and Gauss-Seidel iteration were successfully executed with a tolerance of `1e-3` and an initial guess of the zero vector.

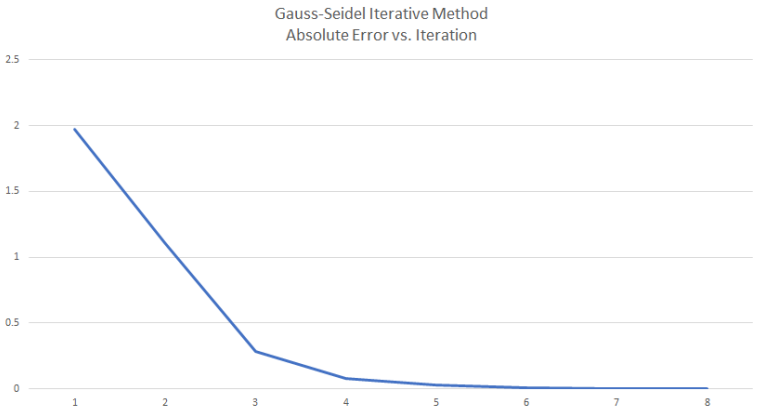

Figure 17: The Jacobi method ran in 47 iterations, taking a total computation time of 3.85 seconds.

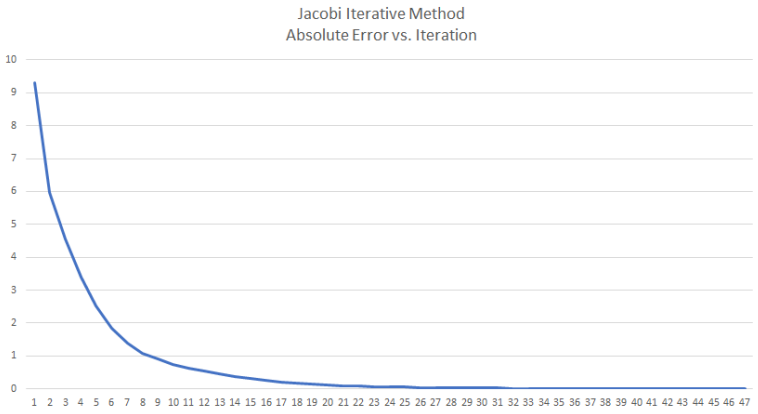

Figure 18: GS method completed in 8 iterations and taking a total time of 0.75 seconds.

In this case, the Gauss-Seidel method was faster and less expensive than Jacobi. It is also worth noting that fewer Jacobi iterations were necessary in the preconditioning process to allow the Gauss-Seidel method to begin to converge, further making this method more effective for this particular system.

5 Nonstationary Methods

Clearly, both direct methods and Stationary Iterative Methods have significant limitations in their ability to solve large sparse systems arising from general multidimensional PDEs. We therefore consider nonstationary methods. A nonstationary iterative method is any iterative method which does not have the form (28). Nonstationary methods are also called Krylov Subspace methods. This class of methods that have become increasingly popular in recent decades due to their flexibility and speed [9]. Like stationary iterative methods, they do not, in general, require storing the entire matrix at once, as direct methods do. They do, however, show improved convergence over stationary methods in many cases.

5.1 Krylov Subspace Methods

The k^{th} Krylov Subspace generated by a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $v \in \mathbb{R}^n$ is defined as

$$K_k(A, v) = \text{span}\left\{A^i v\right\}_{i=0}^{i=k-1} \tag{31}$$

The general approach of a Krylov Subspace method is to successively generate Krylov Subspaces by constructing basis vectors for them, and to find the vector contained in each subspace with the minimum residual $r = \|Ax - b\|$. These basis vectors are usually generated with either Arnoldi iteration or bi-Lanczos iteration, both of which apply variants of the Gram-Schmidt orthogonalization process [8].

The Krylov subspace methods we consider are `GMRES`, `LGMRES`, `BiCG`, `BiCGSTAB`, `QMR`, and `CGS`. Two other well known Krylov Subspace Methods are `CG` and `MINRES`, although they are applicable only to symmetric positive definite systems.

5.2 Convergence and Preconditioning

Unlike nonstationary methods for symmetric positive definite matrices, the convergence properties of `GMRES`, `BiCG` and other nonsymmetric nonstationary methods are not well understood. Generally, performance tends to worsen with increasing condition number and improve with increasing eigenvalue clustering. Preconditioners are very commonly used with Krylov Subspace methods to improve the conditioning of the system in these two regards. Common preconditioners are Gauss-Seidel, incomplete LU factorization (ILU), and Algebraic Multigrid (AMG), which are fully explored in Ghai et al. [8] We avoid preconditioning in this paper, although based on the Ghai et al. [8], the right conditioner may drastically improve convergence speed, and even induce convergence for a diverging method. It is therefore an area which we seek to explore in the future.

6 Numerical Results

We use Python's `scipy.linalg.sparse` to apply all of the methods described above to a systems with spatial and angular resolution between $n_s = n_a = 10$ and $n_s = n_a = 100$. Calculations were performed using the quad-core, 8GB ram computers in the University of Akron Applied Math Research Lab (AMRL) Many of these methods failed, and are thus not shown.

For small systems (60×60 or smaller), direct methods worked effectively. In particular, we used `lu` and `spsolve` from `scipy.linalg.solve`. While `spsolve` is capable of calling `UMFPACK`, I was unable to install it on our computers in the AMRL, despite my efforts. This likely would have shown significant performance improvements over the standard *lu*. However, for the largest systems, the calculations failed, yielding a memory error.

As stated, no stationary iterative methods were successful. The nonstationary iterative methods we applied were `GMRES`, `LGMRES`, `BiCG`, `BiCGSTAB`, `QMR`, and `CGS`. Of them, all but `LGMRES`, `GMRES`, and `CGS` failed explicitly. `CGS`, however, essentially returned the zero vector with small perturbations. `GMRES`, on the other hand, ran indefinitely. It may have been eventually successful, though even in the smaller systems, we waited significantly longer than were necessary for the other methods. `LGMRES` was the only successful nonstationary method, which was very successful at that. It was able to handle large matrices very quickly, overcoming the drawbacks of both direct and stationary method.

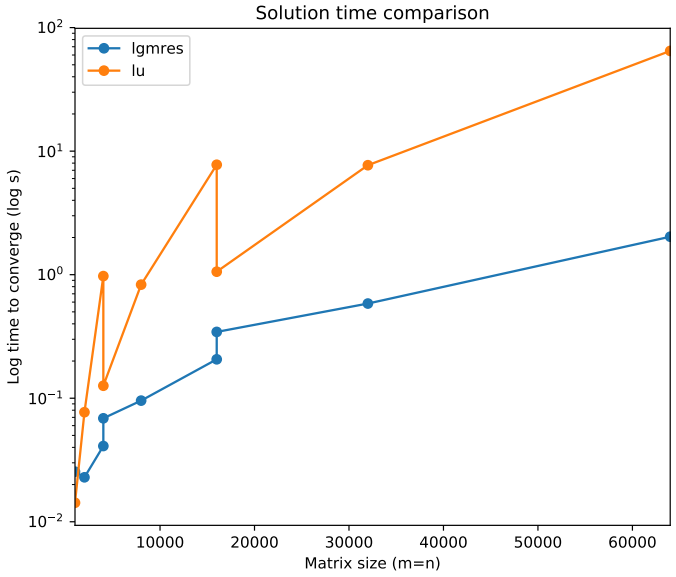

Figure 19: Computational time for `LU` and `GMRES`

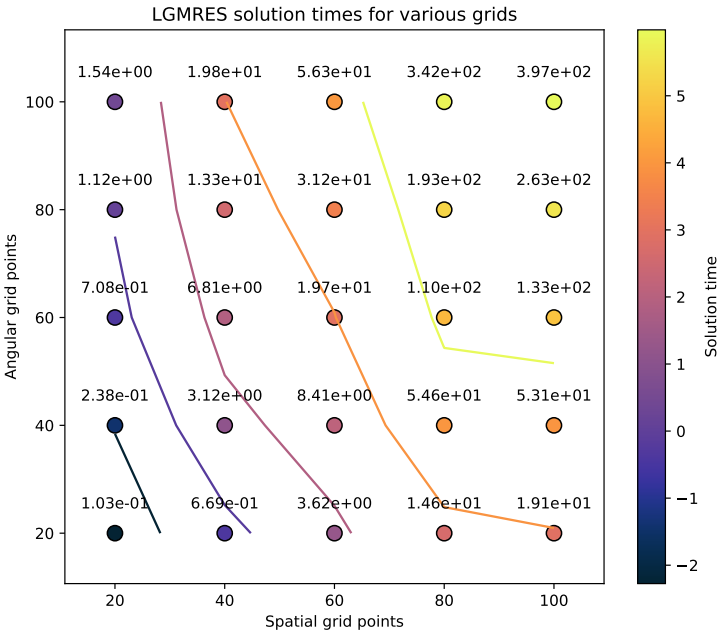

Figure 20: Convergence using `LGMRES` (seconds)

We compare convergence times for `LU` and `LGMRES` for the smaller matrices as a function of matrix dimension. For all matrices, we give convergence times as functions of n_s and n_g. Note that computational time seems to increase more rapidly with n_s than n_a, which is to be expected since n_s represents both n_x and n_y. Thus, doubling n_s quadruples the size of the system. Note that matrix size depends on both n_s and n_a, which clearly affect CPU time independently, since a single matrix sizes corresponds to multiple times in figure 19.

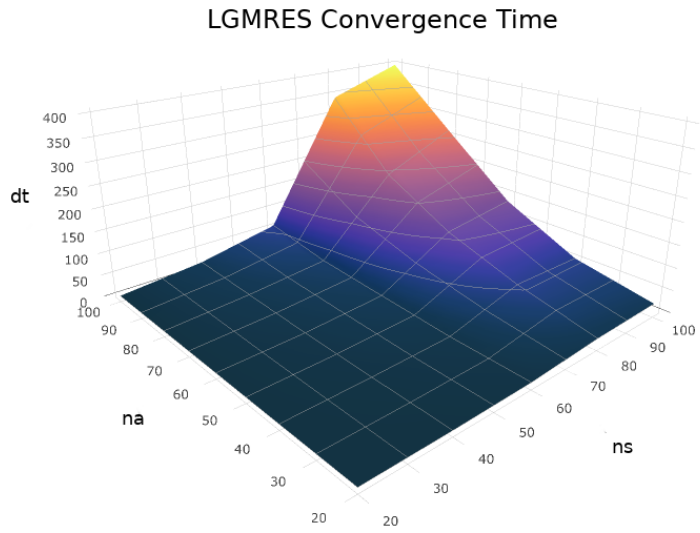

Figure 21: Irradiance plots for several values of n_s and n_a, using `LGMRES`.

6.1 Convergence Study

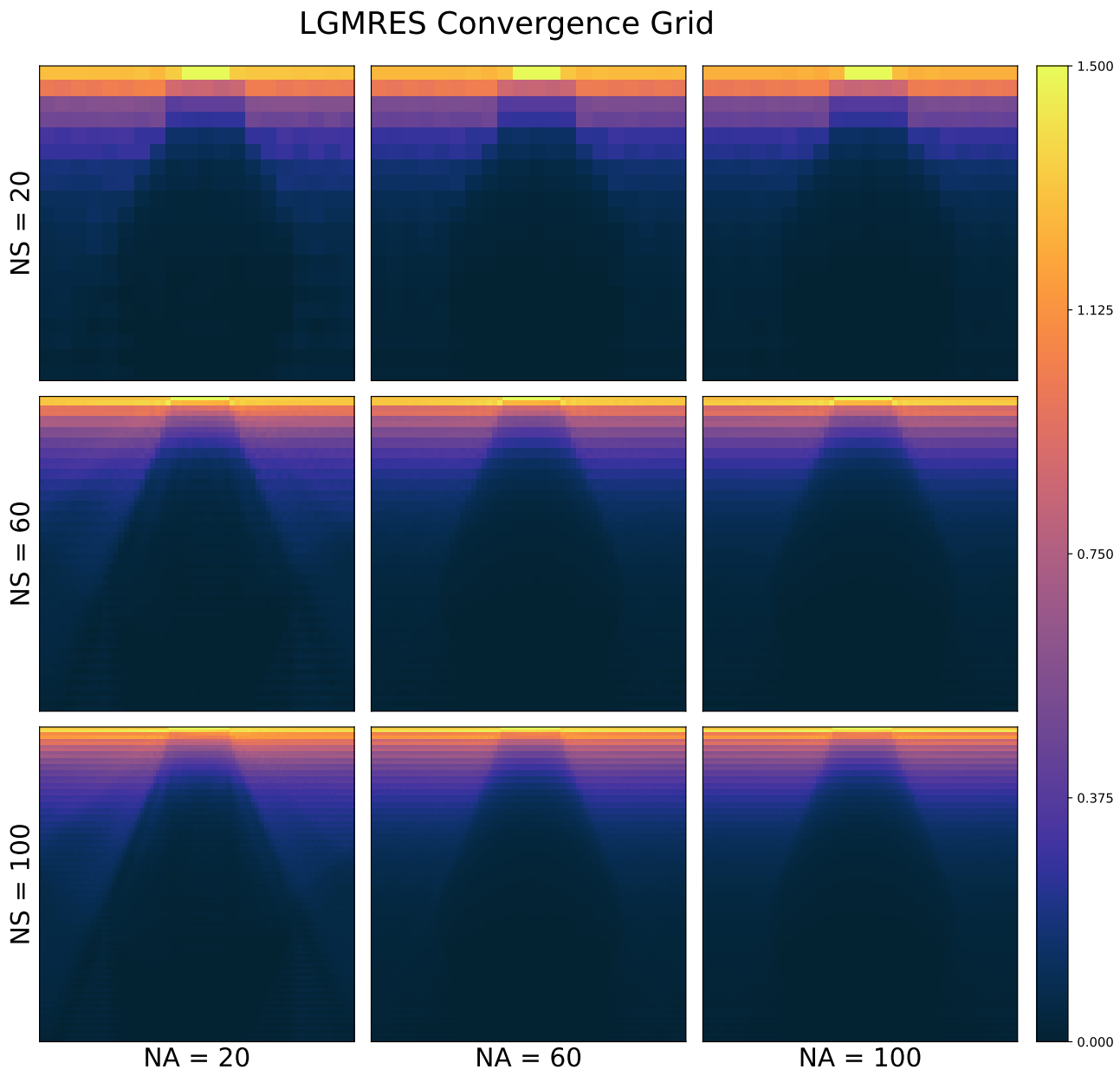

Figure 22: Irradiance plots for several spatial and angular resolutions.

We show irradiance plots for several systems obtained using `LGMRES`. Recall that irradiance is defined as a function of space only, not angle, hence we can display it as a two-dimensional image for our two-dimensional problem. Precisely,

$$I(x, y) = \int_0^{2\pi} L(x, y, \theta)\, d\theta \tag{32}$$

Visually, the 100×100 plot seems sufficiently resolved, although of course, the definition of "sufficiently" varies from case to case. Further, we observe that in the cases with lower angular resolution (e.g. $na = 20, ns = 100$), the paths along which light can travel are clearly visible as discontinuities occurring along straight lines. This leads to the conclusion that angular resolution is more important than spatial resolution for obtaining accurate solutions.

We also compare irradiance values among all grid resolutions as follows. We consider the set of spatial grid points in the lowest resolution trial, namely 20×20 here, and compare irradiances from all spatial and angular resolutions at those particular locations. This is possible since the number of spatial grid points is in each case a multiple of the number in the coarsest case. We treat the highest resolution, here $100x100$, as the "true" solution, and differences respectively. We report the average of these errors over space for each resolution. Comparing maximum values results in a similar image. We also display in figure 23 that `LU` and `LGMRES` produce very similar values, leading us to trust solutions from `LGMRES` in higher resolutions where `LU` is not feasible.

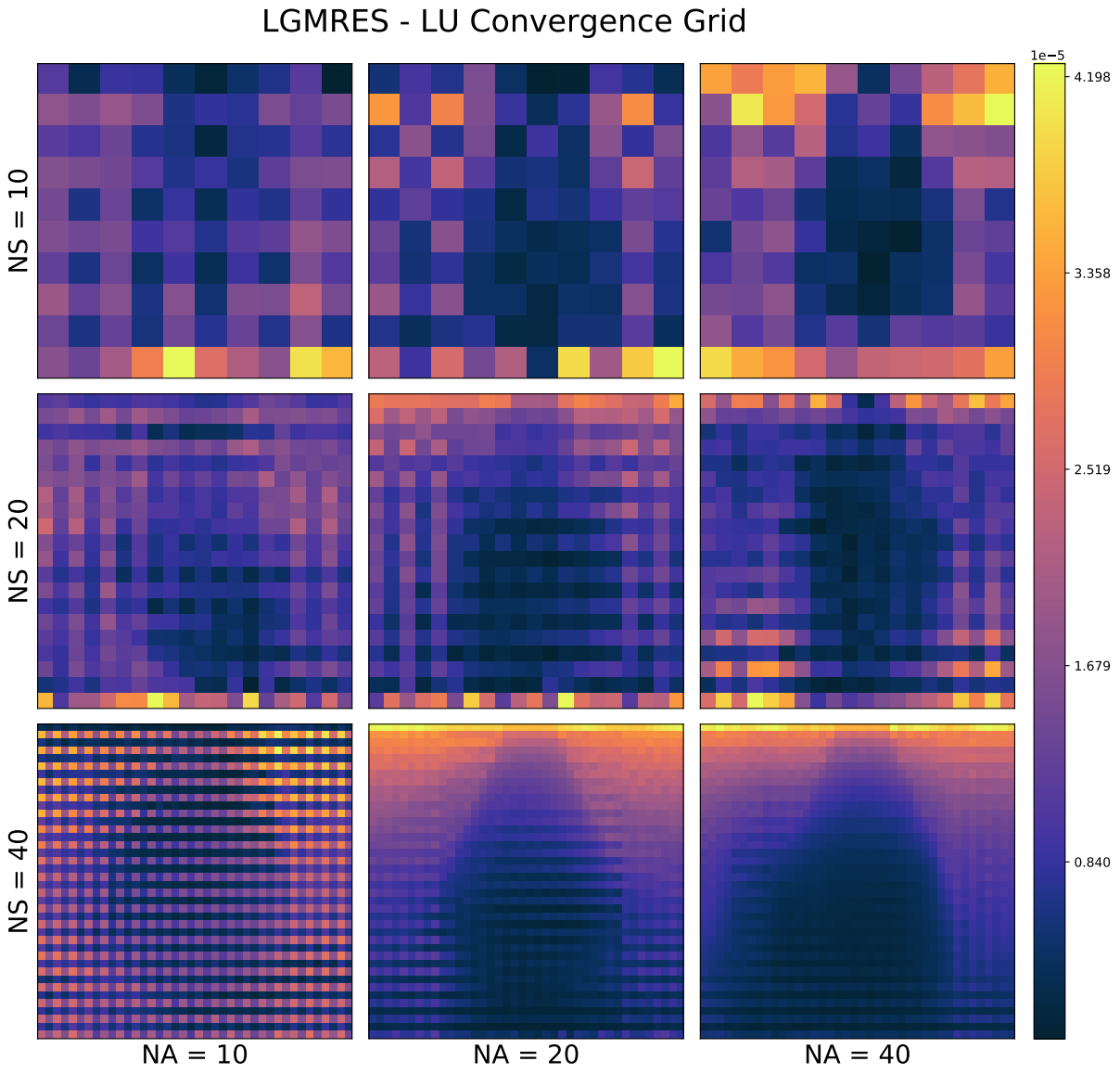

Figure 23: Difference between `LU` and `LGMRES`. All values are less than `5e-5`, indicating nearly identical results.

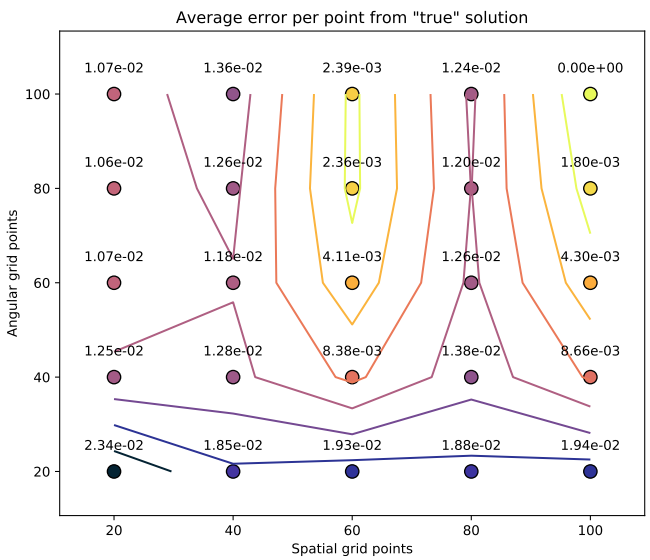

Figure 24: Average error per grid point v.s. 100×100.

Figure 25: Average error per grid point v.s. 100×100.

7 Conclusions and Further Work

We explore a variety of numerical solution techniques for the linear system generated by discretizing the Monochromatic Time-Independent Radiative Transfer Equation using finite differences. We find that in general, the system fails to be diagonally dominant, and thus stationary iterative methods fail to converge. While direct methods are effective for small systems, they are both too slow and require too much memory for larger problems. Hence, Krylov subspace methods are found to be the only feasible solution.

In particular, `LGMRES` was consistently convergent for the set of parameters used in the numerical experiments in this paper. It is not certain whether the method would continue to converge upon varying parameters. Further, the numerical experiments were carried out using the 210 ordering for the coefficient matrix. It may be the case that different orderings have differing convergence properties.

In future work, we will explore different kelp distributions, which may or may not alter the convergence of each method. We also intend to experiment with preconditioning to improve convergence. In particular `AMG` seems appropriate, as it is designed with block structure in mind.

Once we are confident in the accuracy of our numerical solution technique, we will begin developing a constitutive model for irradiance as a function the spatial kelp distribution. This will serve to improve our understanding of the role of light and self-shading in kelp aquaculture.

References

[1] P. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. MUMPS: A General Purpose Distributed Memory Sparse Solver. In *Proceedings of the 5th International Workshop on Applied Parallel Computing, New Paradigms for HPC in Industry and Academia*, PARA '00, pages 121–130, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-41729-X. URL `http://dl.acm.org/citation.cfm?id=645782.666826`.

[2] C. J. Anderson. *Analysis of Controlled Over-Relaxation*. PhD thesis, University of Akron, 2012. URL `http://rave.ohiolink.edu/etdc/view?acc_num=akron1342456200`.

[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

[4] A. H. Baker, E. R. Jessup, and T. Manteuffel. A Technique for Accelerating the Convergence of Restarted GMRES. *SIAM Journal on Matrix Analysis and Applications*, 26(4):962–984, Jan. 2005. ISSN 0895-4798, 1095-7162. doi: 10.1137/S0895479803422014. URL `http://epubs.siam.org/doi/10.1137/S0895479803422014`.

[5] S. Chandrasekhar. *Radiative Transfer*. Dover, 1960. URL `https://archive.org/details/RadiativeTransfer`.

[6] T. A. Davis. Algorithm 832: UMFPACK V4. 3an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):196–199, 2004. URL `http://dl.acm.org/citation.cfm?id=992206`.

[7] S. A. Gershgorin. Uber die abgrenzung der eigenwerte einer matrix. (6):749–754, 1931. URL `http://www.mathnet.ru/rus/im5235`.

[8] A. Ghai, C. Lu, and X. Jiao. A Comparison of Preconditioned Krylov Subspace Methods for Nonsymmetric Linear Systems. *arXiv:1607.00351 [math]*, July 2016. URL `http://arxiv.org/abs/1607.00351`.

[9] M. H. Gutknecht. A brief introduction to Krylov space methods for solving linear systems. In *Frontiers of Computational Science*, pages 53–62. Springer, 2007. URL `http://link.springer.com/content/pdf/10.1007/978-3-540-46375-7_5.pdf`.

[10] A. Jacobs. The pathologies of big data. *Communications of the ACM*, 52(8):36–44, 2009. URL `http://dl.acm.org/citation.cfm?id=1536632`.

[11] E. Jones, T. Oliphant, P. Peterson, and others. *SciPy: Open source scientific tools for Python*. 2001. URL `http://www.scipy.org`.

[12] C. Mobley. Radiative Transfer in the Ocean. In *Encyclopedia of Ocean Sciences*, pages 2321–2330. Elsevier, 2001. ISBN 978-0-12-227430-5. URL `http://linkinghub.elsevier.com/retrieve/pii/B012227430X004694`.

[13] C. Mobley. The Volume Scattering Function and Models for Scattering, July 2013. URL `ftp://misclab.umeoce.maine.edu/users/optics/classFTP2013/Lectures/Lec6_VSF_ScatModels.pdf`.

[14] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer series in operations research. Springer, New York, 1999. ISBN 978-0-387-98793-4.

[15] F. Pérez and B. E. Granger. IPython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29, May 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.53. URL `http://ipython.org`.

[16] T. J. Petzold. Volume Scattering Function for Selected Ocean Waters. Technical report, DTIC Document, 1972. URL `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0753474`.

[17] D. Young. *Iterative solution of large linear systems*. Computer science and applied mathematics. Academic Press, 1971. URL `https://books.google.com/books?id=oALvAAAAMAAJ`.

[18] J. Y. Yuan and P. Y. Yalamov. A method for constructing diagonally dominant preconditioners based on Jacobi rotations. *Applied Mathematics and Computation*, 174(1):74–80, Mar. 2006. ISSN 00963003. doi: 10.1016/j.amc.2003.10.070. URL `http://linkinghub.elsevier.com/retrieve/pii/S009630030500442X`.