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

Project Summary
ASSETs to Serve Humanity NSF REU 2016

Oliver Evans
Fred Weiss
Christopher Parker
Emmanuel Arkoh

Dr. Malena Español

May 12, 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 [9].

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` [8] package for sparse linear algebra. `IPython` [10] 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 $x \in \mathcal{R}^n$. Let Ω be the unit sphere in $\mathcal{R}^n$. Let $\omega \in \Omega$ be a unit vector in $\mathcal{R}^n$. 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 $a(x)$ and $b(x)$ denote the absorption and scattering coefficients respectively of the medium, which are both functions of P_k. Let $\beta(\Delta\theta)$ denote the normalized *volume scattering function* or *phase function*, which defines the probability of light scattering at an angle $\Delta\theta$ from it's initial direction in a scattering event.

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

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

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.

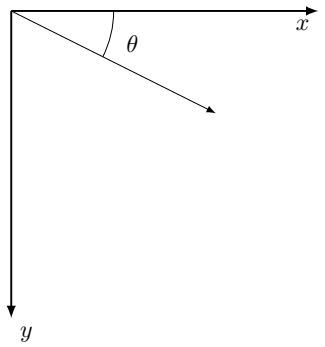

Figure 1: 2D coordinate system

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

$$\frac{\partial L}{\partial x}\cos\theta + \frac{\partial L}{\partial y}\sin\theta = -(a+b)L(x,y,\theta) + b\int_0^{2\pi}\beta(|\theta-\theta'|)\,d\theta', \qquad (1)$$

where $|\theta - \theta'|$ measures the smallest angular difference between $]thet$ 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

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

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,

$$L(x, 0, \theta) = 0, \quad \text{for} \quad \theta \in [\pi, 2\pi). \qquad (3)$$

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 (1) 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 chapter 2 of Chandrasekhar [5]. 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 $P_{n_\theta}(\theta)$. 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

$$x_i = (i-1)\,dx, \qquad i = 1, \ldots, n_x \tag{4}$$
$$y_j = (j-1)\,dy, \qquad j = 1, \ldots, n_y \tag{5}$$
$$\theta_k \text{ s.t. } P_{n_\theta}(\theta_k/2\pi) = 0, \qquad k = 1, \ldots, n_\theta \tag{6}$$

In the same notational vein, let

$$L^k_{ij} = L(x_i, y_j, \theta_k), \tag{7}$$
$$\beta_{kl} = \beta(|\theta_k - \theta_l|), \tag{8}$$
$$a_{ij} = a(x, y) \tag{9}$$
$$b_{ij} = b(x, y) \tag{10}$$

where $|\cdot|$ is periodic as in (1).

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) is

$$\frac{1}{2dx}\left(L^k_{i+1,j} - L^k_{i-1,j}\right)\cos\theta_k - \pi b \sum_{\substack{l=1 \\ l\neq k}}^{n_\theta} a_l \beta_{kl} L^l_{ij} + \frac{1}{2dy}\left(L^k_{i,j+1} - L^k_{i,j+1}\right)\sin\theta_k - (a_{ij} + b_{ij})L^k_{ij} = 0 \tag{11}$$

Similarly, we discretize using (FD2) and (BD2) at the boundaries.

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. Note that each matrix has a unique sparsity pattern which may lend itself to one algorithm over another. In general, it is difficult to predict *a priori* exactly how a sparsity pattern will affect a solution technique, although it seems clear that a method which explicitly accounts for a particular sparsity pattern would be advantageous.

2.4 Sparsity Plots

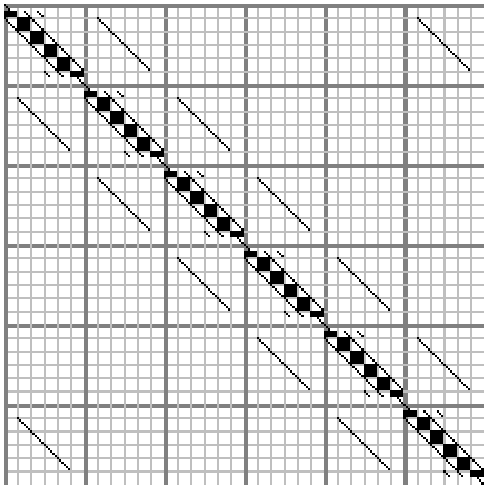

Figure 2: Sparsity plot: 6x6x6, ordering 012

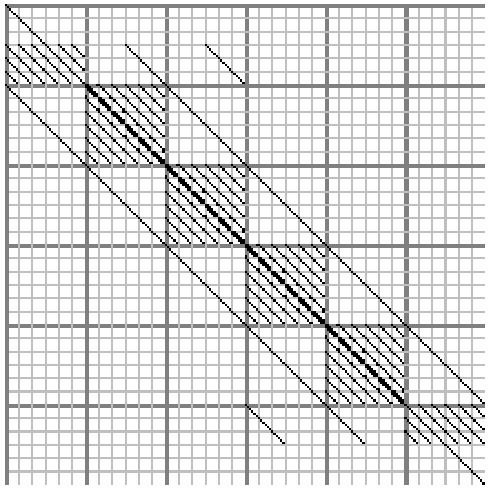

Figure 5: Sparsity plot: 6x6x6, ordering 120

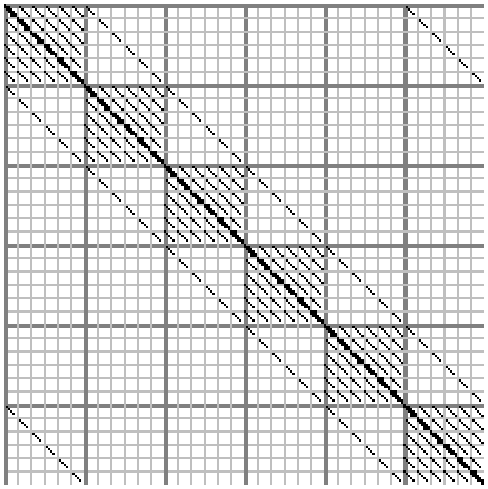

Figure 3: Sparsity plot: 6x6x6, ordering 021

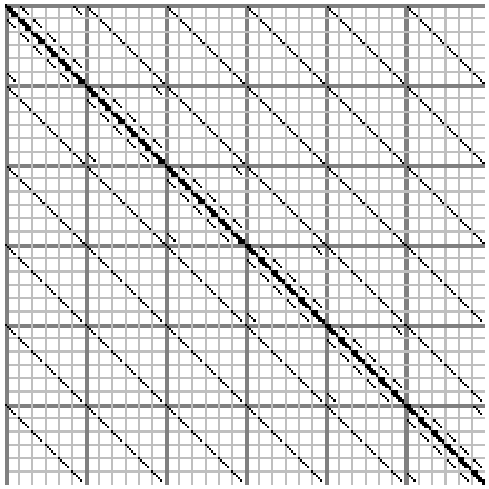

Figure 6: Sparsity plot: 6x6x6, ordering 201

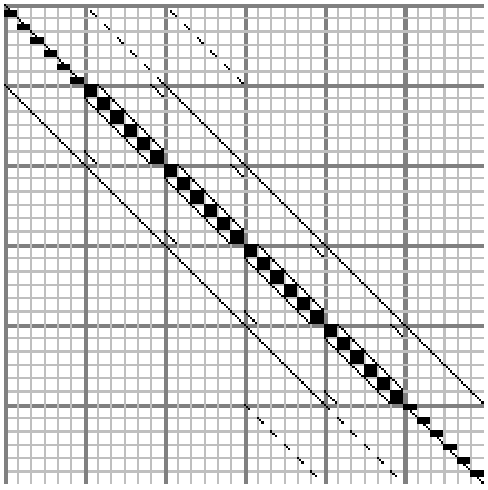

Figure 4: Sparsity plot: 6x6x6, ordering 102

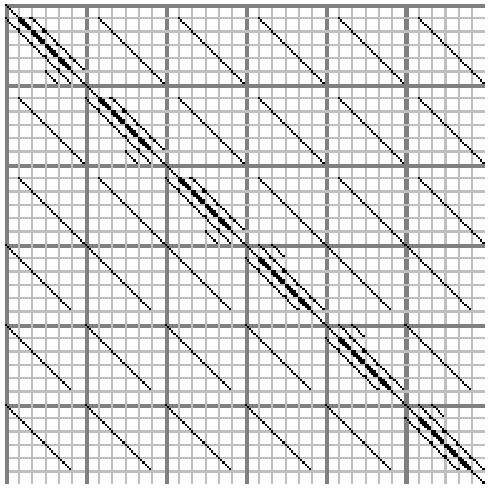

Figure 7: Sparsity plot: 6x6x6, ordering 210

2.5 Matrix Properties

2.5.1 Diagonal Dominance

2.5.2 Gershgorin Disks

2.5.3 Eigenvalue Clustering

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 6 to 10^5 times slower than computations in memory [7].

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 only keep small slices (fronts) of the matrix in memory at any given time, and act on several fronts in parallel, drastically reducing solution time.

4 Stationary Iterative Methods

4.1 Fixed-Point Iteration

4.2 Convergence and Preconditioning

5 Nonstationary Iterative Methods

5.1 Krylov Subspace Methods

5.2 Convergence and Preconditioning

6 Numerical Results

7 Conclusions

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] 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`.

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

[9] 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`. DOI: 10.1006/rwos.2001.0469.

[10] F. Prez 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`.