Solving the Radiative Transfer Equations via the Radon Transform

Eappen Nelluvelil
Rice University
esn2@rice.edu

Megan Oeltjenbruns
Wayne State College
meoelt01@wsc.edu

Jacob Spainhour
Florida State University
jcs15c@my.fsu.edu

Abstract

We aim to efficiently solve multi-dimensional partial differential equations using the Radon transform. By applying the Radon transform to such a 2D PDE, we reduce the problem to a collection of 1D PDEs that can be time-stepped up to a desired final time. Following this, the inverse of the Radon transform is applied to return the new solution to physical space. We specifically apply this method to the P_N approximation for radiative transfer, a hyperbolic system of linear PDEs.

1 Introduction

1.1 Imaging

The imaging problem is to reconstruct a distribution from several profiles taken at different angles. The Radon transform is generally used to solve the imaging problem by producing profiles. This approach can be used because certain problems are easier to solve profile-by-profile. An example of this is a medical imaging device. The device gets profiles at several angles and the profiles need to be reconstructed to create an accurate image.

1.2 Previous Work

The inspiration for this project came from *Dimensional Splitting of Hyperbolic Partial Differential Equations using the Radon Transform* by Donsub Rim.[4] Rim was able to dimensionally split partial differential equations using the Radon transform and use large time stepping methods while solving the partial differential equations. He was able to accomplish $1^{st} - 2^{nd}$ order accuracy.

2 The Radon Transform and Backprojection

2.1 The Radon Transform

To properly introduce the Radon transform, we begin with the standard xy coordinate system. If we rotate our axes by a positive angle ω with respect to the positive x-axis, we obtain the sz coordinate system.

Given a point (s, z) in the sz coordinate system, we can write the corresponding point in the xy coordinate system using a rotation matrix:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\omega) & -\sin(\omega) \\ \sin(\omega) & \cos(\omega) \end{bmatrix} \begin{bmatrix} s \\ z \end{bmatrix}$$

Likewise, given a point (x, y) in the xy coordinate system, we can write the corresponding point in the sz coordinate system:

$$\begin{bmatrix} s \\ z \end{bmatrix} = \begin{bmatrix} \cos(\omega) & \sin(\omega) \\ -\sin(\omega) & \cos(\omega) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Suppose $f(x,y) : \mathbb{R}^2 \to \mathbb{R}$ is a function with compact support. The **Radon transform** of f, denoted $\mathcal{R}(f)(s,\omega) = \widehat{f}(s,\omega)$, is defined as follows:

$$\mathcal{R}(f)(s,\omega) := \int_{-\infty}^{\infty} f\left(s\cos(\omega) - z\sin(\omega), s\sin(\omega) + z\cos(\omega)\right)\, dz$$

Geometrically, for a fixed angle ω, the Radon transform of f is defined through integrals along lines perpendicular to the s axis following the rotation by ω.[3] We can easily verify that the Radon transform is a linear operator.

2.2 Intertwining Property of the Radon Transform

A useful property of the Radon transform is how it simplifies spatial partial derivatives of functions. This can be demonstrated by some simple calculations.

Suppose $f(x,y) : \mathbb{R}^2 \to \mathbb{R}$ is function with compact support, and further suppose $f_{,x}$ and $f_{,y}$ exist. Note from earlier that

$$\begin{aligned} x(s,z;\omega) &= s\,\cos(\omega) - z\,\sin(\omega) \\ y(s,z;\omega) &= s\,\sin(\omega) + z\,\cos(\omega) \end{aligned}$$

We can use chain rule to compute partial derivatives with respect to x and y, respectively:

$$\begin{aligned} \frac{\partial}{\partial x} &= \frac{\partial s}{\partial x}\frac{\partial}{\partial s} + \frac{\partial z}{\partial x}\frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} &= \frac{\partial s}{\partial y}\frac{\partial}{\partial s} + \frac{\partial z}{\partial y}\frac{\partial}{\partial z} \end{aligned}$$

Also note from earlier that

$$\begin{aligned} s(x,y;\omega) &= x\cos(\omega) + y\sin(\omega) \\ z(x,y;\omega) &= -x\sin(\omega) + y\cos(\omega) \end{aligned}$$

Thus,

$$\begin{aligned} \frac{\partial s}{\partial x} &= \cos(\omega) & \frac{\partial z}{\partial x} &= -\sin(\omega) \\ \frac{\partial s}{\partial y} &= \sin(\omega) & \frac{\partial z}{\partial y} &= \cos(\omega) \end{aligned}$$

Substituting the above into the expressions for the partial derivatives with respect to x and y, we obtain the following:

$$\begin{aligned} \frac{\partial}{\partial x} &= \cos(\omega)\,\frac{\partial}{\partial s} - \sin(\omega)\,\frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} &= \sin(\omega)\,\frac{\partial}{\partial s} + \cos(\omega)\,\frac{\partial}{\partial z} \end{aligned}$$

We now consider what happens when we compute $\mathcal{R}(f_{,x})(s,\omega)$.

$$\begin{aligned} \mathcal{R}(f_{,x})(s,\omega) &= \int_{-\infty}^{\infty} f_{,x}\, dz \\ &= \int_{-\infty}^{\infty} \left(\cos(\omega)\, f_{,s} - \sin(\omega)\, f_{,z}\right)\, dz \\ &= \cos(\omega)\int_{-\infty}^{\infty} f_{,s}\, dz - \sin(\omega)\int_{-\infty}^{\infty} f_{,z}\, dz \\ &= \cos(\omega)\frac{\partial}{\partial s}\int_{-\infty}^{\infty} f\, dz - \sin(\omega)\left[f(z=+\infty) - f(z=-\infty)\right] \\ &= \cos(\omega)\,\frac{\partial}{\partial s}\mathcal{R}(f)(s,\omega) \\ &= \cos(\omega)\,\widehat{f}_{,s}(s,\omega) \end{aligned}$$

We also consider what happens when we compute $\mathcal{R}\left(f_{,y}\right)(s,\omega)$:

$$\begin{aligned}
\mathcal{R}\left(f_{,y}\right)(s,\omega) &= \int_{-\infty}^{\infty} f_{,y}\, dz \\
&= \int_{-\infty}^{\infty} \left(\sin(\omega)\, f_{,s} - \cos(\omega)\, f_{,z}\right)\, dz \\
&= \sin(\omega) \int_{-\infty}^{\infty} f_{,s}\, dz - \cos(\omega) \int_{-\infty}^{\infty} f_{,z}\, dz \\
&= \sin(\omega) \frac{\partial}{\partial s} \int_{-\infty}^{\infty} f\, dz - \cos(\omega)\left[f(z=+\infty) - f(z=-\infty)\right] \\
&= \sin(\omega)\, \frac{\partial}{\partial s} \mathcal{R}(f)(s,\omega) \\
&= \sin(\omega)\, \widehat{f}_{,s}\,(s,\omega)
\end{aligned}$$

In summary,

$$\begin{aligned}
\mathcal{R}\left(f_{,x}\right)(s,\omega) &= \cos(\omega)\, \widehat{f}_{,s}\,(s,\omega) \\
\mathcal{R}\left(f_{,y}\right)(s,\omega) &= \sin(\omega)\, \widehat{f}_{,s}\,(s,\omega)
\end{aligned}$$

2.3 Discretizing the Radon Transform

We begin by restricting our non-zero domain to be the unit circle, and select points within at which to calculate our Radon transform. As ω is constant across each profile of the Radon transform, we uniformly space our discretization along diameters of the unit circle into N_ω evenly spaced diameters. In addition, the domain needs N_s points along each diameter.

Immediately we observe that uniformly spacing these points is not ideal, as Runge phenomena would quickly be observed in many of the following methods introduced. Instead, Chebyshev points of the second kind are placed along each diameter, which are generated according to the following formula:

$$\begin{gathered}
\cos\left(\frac{j\pi}{N_s}\right) \\
j = 1, 2, \ldots, N_s
\end{gathered}$$

By using Chebyshev points, these methods are instead spectrally accurate. That is to say, for C^∞ functions, the error of these methods rapidly converges to zero.

2.4 Quadrature and Interpolation

Because we assume our compact support to be on the unit circle, it is possible to determine the Radon transform at a point by computing the requisite line integral across chords of the domain, whose length is denoted as 2τ, rather than to infinity. The integration is then performed using the Clenshaw-Curtis quadrature, whose N_q nodes are Chebyshev points and weights are derived through the fast Fourier transform.[6]

$$\mathcal{R}(f) = \int_{-\infty}^{\infty} f(x,y)\, dz = \int_{-\tau}^{\tau} f(x,y)\, dz \approx \sum_{i=1}^{N_q} w_i\, f(z_i)$$

Applied to our Radon transform, the formula is as follows:

$$\mathcal{R}(f)(s,\omega) \approx \tau \sum_{k=0}^{N_q} w_k\, f(s_i \cos(\omega_j) - \tau t_k \sin(\omega_j),\ s_i \sin(\omega_j) + \tau t_k \cos(\omega_j))$$

where t_k are Chebyshev points on the interval $[-1, 1]$. This process is implemented through the following psuedocode:

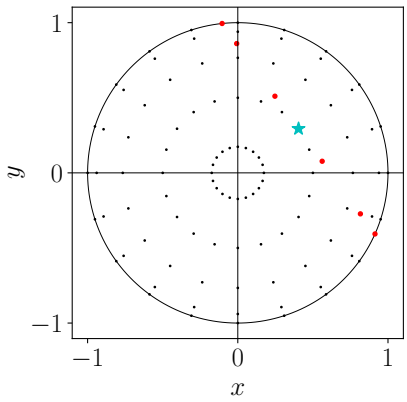


Figure 1: 6 point quadrature rule for the point (x, y), denoted by a blue star.
Quadrature nodes labeled red, grid points labeled black

For the purposes of computing the proper inverse Radon transform, we perform the forward Radon transform as if we only know f at the predefined grid points. However, as shown in the figure above, these points rarely line up exactly with the points needed for quadrature. Therefore, interpolation needs to be done to find those function values. To obtain as much accuracy in the interpolation as possible, a series of one-dimensional interpolation schemes. The point at which f needs to be approximated is first projected onto the four closest diameters. Barycentric interpolation is then performed along these diameters. The stabilized barycentric interpolation formula is reproduced below.

$$H_n(x) = \frac{\sum_{j=0}^{n} \alpha_j \frac{f_j}{x - x_j}}{\sum_{j=0}^{n} \alpha_j \frac{1}{x - x_j}}$$

$$\alpha_0 = \frac{1}{2}, \quad \alpha_{1:n-1} = (-1)^{1:n-1}, \quad \alpha_n = \frac{1}{2}(-1)^n$$

Next, four point polynomial interpolation is performed across these four points. Let θ_0 be the midpoint of the arc containing h_i, $i = 1, \ldots, 4$, obtained through barycentric interpolation

$$\begin{aligned} p(\theta) = &-\frac{1}{16}(h_1 - 9h_2 - 9h_3 + h_4) \\ &+\frac{1}{24d\omega}(h_1 - 27h_2 + 27h_3 - h_4)(\theta - \theta_0) \\ &+\frac{1}{4d\omega^2}(h_1 - h_2 - h_3 + h_4)(\theta - \theta_0)^2 \\ &-\frac{1}{6d\omega^3}(h_1 - 3h_2 + 3h_3 - h_4)(\theta - \theta_0)^3 \end{aligned}$$

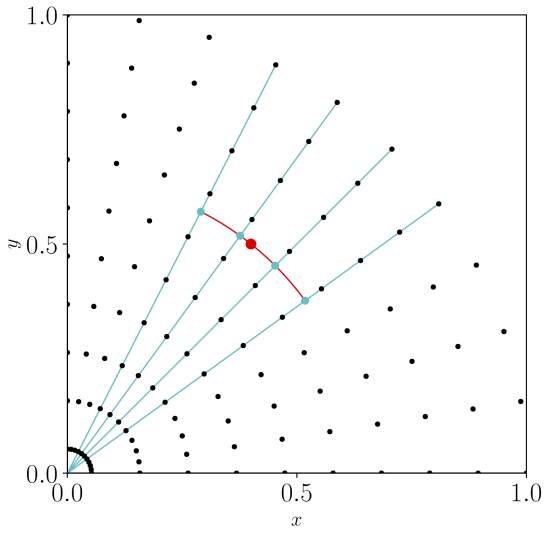


Figure 2: Approximates f at a point (x, y), labeled with a red dot, using the values calculated at the four points labeled with a blue dot

This interpolation procedure is performed for each quadrature point for a single Radon transform evaluation. The quadrature method itself is applied to each point in the grid, resulting in the Radon transform of our discretized domain.

2.5 Backprojection

The adjoint of the Radon transform, denoted $\mathcal{R}^*$, is known as **backprojection**.

Given a function $\widehat{f}(s, \omega) : \mathbb{R} \times [0, \pi] \to \mathbb{R}$, the backprojection of $\widehat{f}$ is defined as follows:

$$\mathcal{R}^* \left(\widehat{f} \right) (x, y) := \int_0^\pi \widehat{f} \left(x \cos(\omega) + y \sin(\omega), \omega \right) \, d\omega$$

Geometrically, the backprojection of $\widehat{f}$ at a point (x, y) is the integral along the circle whose diameter is the line segment connecting (x, y) and the origin.

The backprojection is a useful building block in computing the inverse Radon transform [3].

2.6 Discretizing the Backprojection

As was the case with the Radon transform, we need to discretize the backprojection for computational purposes. Recall that our underlying mesh is parameterized by two integers:

- N_ω, the number of equi-spaced angles
- N_s, the number of Chebyshev points of the second kind along any one angle

This means our mesh consists of $N_p := N_\omega \times N_s$ points.

Given a function $\widehat{f}(s, \omega)$, we want to compute the *approximate* backprojection $\widehat{f}$ at a mesh point (x_k, y_k) where $k = 1, 2, \ldots, N_p$. We can do this in the following manner:

1. Draw a circle that passes through (x_k, y_k) and the origin
2. Select N_b many equi-spaced (in angle) points along the circle's circumference
3. Interpolate function values at the discrete points along the circle's circumference
4. Use quadrature rules to approximate backprojection at (x_k, y_k) using interpolated function values

Note: We use the same four-point interpolation scheme discussed earlier to interpolate function values along the above circle.

Moreover, the quadrature rule used in the above scheme is very simple. Once we have interpolated $\widehat{f}$ at the N_b many points, we simply take their sum and divide by N_b to obtain the approximate backprojection of $\widehat{f}$ at (x_k, y_k).

2.7 Inverse Radon Transform

Recall that we have discretized the continuous Radon transform. That is, we can approximate $\mathcal{R}(f)(s, \omega)$ on a mesh where

- $f(x, y) : \mathbb{R}^2 \to \mathbb{R}$ is a function with compact support on the unit disk, and
- the mesh is parameterized by N_ω and N_s and consists of $N_p := N_\omega \times N_s$ points

We typically have global knowledge of f, but computationally, we only use information about f at the N_p many mesh points to compute the approximate Radon transform of f on the mesh.

That is, we only use $\underline{f} \in \mathbb{R}^{N_p}$ (a vector whose i^{th} entry is f evaluated at the i^{th} mesh point) to compute $\underline{\widehat{f}} \in \mathbb{R}^{N_p}$ (a vector whose i^{th} entry is an approximation of $\mathcal{R}(f)$ at the i^{th} mesh point).

Concretely,

$$\underline{\underline{R}}\,\underline{f} = \underline{\widehat{f}} \tag{1}$$

where $\underline{\underline{R}} \in \mathbb{R}^{N_p \times N_p}$ is a matrix whose j^{th} column is the discretized Radon transform evaluated at $\underline{e_j} \in \mathbb{R}^{N_p}$, the j^{th} canonical basis vector.

Note: We typically use a spectrally accurate method (rather than the fourth-order minimum method described earlier) to compute $\underline{\widehat{f}}$.

We are interested in knowing if we can compute $\underline{f}$ given only $\underline{\widehat{f}}$. Computationally, this comes down to solving $\underline{\underline{R}}\,\underline{f} = \underline{\widehat{f}}$, and we can try a number of direct and iterative solution techniques to solve the system:

- Gaussian elimination
- LU decompositions
- Row-action methods e.g. Kaczmarz's method
- Krylov subspace methods e.g. BiCGSTAB, GCRTOMK, etc.

However, the above system is *highly* ill-conditioned and the above solvers yielded poor solutions.

We can try to alleviate this issue by applying a pre-conditioner to the system $\underline{\underline{R}}\,\underline{f} = \underline{\widehat{f}}$, but there are several difficulties with this approach:

- Pre-conditioners are typically selected on a case-by-case basis, and there does not appear to be much existing literature on how to select a pre-conditioner for this problem.
- Numerical tests show that $\underline{\underline{R}}$ does not have "nice" properties such as symmetry, positive (semi-)definiteness, etc.
- Numerical tests also show that $\underline{\underline{R}}$ is not particularly sparse; as we increase the mesh resolution, the sparsity of $\underline{\underline{R}}$ appears to asymptotically reach $\approx 50\%$.

We investigated various pre-conditioners (despite the aforementioned difficulties) to see if they were feasible, including:

- Sparse, incomplete LU (SPILU) decompositions
- Approximate pseudo-inverses

However, these approaches did not fare much better. Solving these pre-conditioned systems yielded better qualitative, but not better numerical, solutions.

Moreover, solving these pre-conditioned systems requires us to explicitly work with $\underline{\underline{R}}$. This is undesirable because $\underline{\underline{R}}$ is expensive to compute (as we generate it column by column) and *very* large.

However, note that

$$\underline{\underline{R}}\,\underline{f} = \widehat{\underline{f}} \quad \Longleftrightarrow \quad \underline{\underline{R}}^T\,\underline{\underline{R}}\,\underline{f} = \underline{\underline{R}}^T\,\widehat{\underline{f}} \tag{2}$$

In this context, pre-multiplication by $\underline{\underline{R}}^T$ means computing the approximate, discretized backprojection.

The system on the right is known as the **normal equations**, and there is one particular advantage to working with the normal equations compared to the system on the left. Namely, $\underline{\underline{R}}^T\,\underline{\underline{R}}$ is symmetric, positive (semi-)definite. This is useful if we choose to solve the normal equations via iterative methods that take advantage of this property e.g. the conjugate gradient method.

While we can generate $\underline{\underline{R}}$ for a particular mesh and solve the normal equations $\underline{\underline{R}}^T\,\underline{\underline{R}}\,\underline{f} = \underline{\underline{R}}^T\,\widehat{\underline{f}}$, this is not feasible for aforementioned reasons. However, we do not need the matrix $\underline{\underline{R}}^T\,\underline{\underline{R}}$; we only need the matrix-vector products $\underline{\underline{R}}\,\underline{f}$ and $\underline{\underline{R}}^T\,\underline{\underline{R}}\,\underline{f}$, which are far cheaper to compute.

Note: There is a caveat here. We have described ways of computing the approximate, discretized Radon transform and backprojection earlier, but computationally, they are **not** exact adjoints of one another. This is due to the nature of our discretization scheme.

Since we only need knowledge of the matrix-vector products $\underline{\underline{R}}\,\underline{f}$ and $\underline{\underline{R}}^T\,\underline{\underline{R}}\,\underline{f}$, we can solve the system

$$\underline{\underline{R}}^T\,\underline{\underline{R}}\,\underline{f} = \underline{\underline{R}}^T\,\widehat{\underline{f}} \tag{3}$$

using iterative methods.

Here, the biconjugate gradient stabilized (BiCGSTAB) method was used to solve the above system with a starting guess of $\underline{f}^{(0)} = \underline{0}$.

3 Linear Hyperbolic Partial Differential Equations

3.1 The Radiative Transfer Problem

$$F_{,t} + \underline{\Omega} \cdot \underline{\nabla} F + \sigma_t F = \frac{\sigma_s}{4\pi} \int_{\mathbb{S}^2} F \, d\underline{\Omega}$$

The radiative transfer problem is one of great interest in computational mathematics. It serves as a kinetic model for subatomic particles propagating through a homogeneous medium, where $F(t, \underline{x}, \underline{\Omega}) : \mathbb{R}^+ \times \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}$ is a distribution for particles at position $\underline{x}$ with velocity $\underline{\Omega}$. The equation itself is linear, but the solution function is expressed in $5 + 1$ dimensions. As a result, it is exceptionally difficult to discretize and solve immediately, and a significant amount of effort is put in to reduce the number of dimensions present.

The first such effort comes from the spherical harmonics expansion, a series expression that isolates the dependencies of the $\underline{\Omega} = [\mu\ \phi]^T$ variables. Because the terms Y_ℓ^m are already known by construction, we are left to calculate the F_ℓ^m terms. Moreover, we only consider the 2D case.

$$F(t, \underline{x}, \underline{\Omega}) \approx \sum_{\ell=0}^{N} \sum_{m=-\ell}^{\ell} F_\ell^m(t, \underline{x}) Y_\ell^m(\mu, \phi)$$

Taking the first N terms of this series, we can substitute it into the original radiative transfer equation. Algebraic manipulation[1] reveals $\frac{1}{2}(N+1)(N+2)$ equations combined into a system of linear partial differential equations of the following form:

$$\underline{F}_{,t} + \underline{\underline{A}}\,\underline{F}_{,x} + \underline{\underline{B}}\,\underline{F}_{,y} = \underline{\underline{C}}\,\underline{F}$$

Note that we have reduced our $5 + 1$ dimensional equation to one in $2 + 1$ dimensions. We then apply our Radon transform to the PDE to remove an additional spatial derivative, as we will see.

3.2 Radon Transforms of Partial Derivatives

We consider the following system of M first-order, linear hyperbolic PDEs:

$$\begin{aligned} \underline{q}_{,t} + \underline{\underline{A}}\,\underline{q}_{,x} + \underline{\underline{B}}\,\underline{q}_{,y} &= \underline{\underline{C}}\,\underline{q} \\ \underline{q}\,(t = t_0, x, y) &= \underline{q_0}\,(x, y) \end{aligned}$$

where

- $\underline{q}(t, x, y) : \mathbb{R}^+ \times \mathbb{R}^2 \to \mathbb{R}^M$ is a vector valued function of time and space
- $\underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}} \in \mathbb{R}^{M \times M}$ are constant matrices
- $\underline{q_0}(x, y) \in \mathbb{R}^M$ is a vector of initial conditions

We can apply the Radon transform to the above PDE and and use the properties of the transform to simplify spatial partial derivatives.

$$\mathcal{R}\Big(\underline{q}_{,t} + \underline{\underline{A}}\,\underline{q}_{,x} + \underline{\underline{B}}\,\underline{q}_{,y} = \underline{\underline{C}}\,\underline{q}\Big) \tag{4}$$

$$\implies \mathcal{R}\Big(\underline{q}_{,t}\Big) + \mathcal{R}\Big(\underline{\underline{A}}\,\underline{q}_{,x}\Big) + \mathcal{R}\Big(\underline{\underline{B}}\,\underline{q}_{,y}\Big) = \underline{\underline{C}}\,\mathcal{R}(\underline{q}) \tag{5}$$

$$\implies \mathcal{R}\Big(\underline{q}_{,t}\Big) + \underline{\underline{A}}\,\mathcal{R}\Big(\underline{q}_{,x}\Big) + \underline{\underline{B}}\,\mathcal{R}\Big(\underline{q}_{,y}\Big) = \underline{\underline{C}}\,\mathcal{R}(\underline{q}) \tag{6}$$

$$\implies \widehat{\underline{q}}_{,t} + \cos(\omega)\ \underline{\underline{A}}\,\widehat{\underline{q}}_{,s} + \sin(\omega)\ \underline{\underline{B}}\,\widehat{\underline{q}}_{,s} = \underline{\underline{C}}\,\widehat{\underline{q}} \tag{7}$$

$$\implies \widehat{\underline{q}}_{,t} + \big(\cos(\omega)\ \underline{\underline{A}} + \sin(\omega)\ \underline{\underline{B}}\,\big)\,\widehat{\underline{q}}_{,s} = \underline{\underline{C}}\,\widehat{\underline{q}} \tag{8}$$

In (2) and (3), we use the linearity of the Radon transform to apply it separately to each term of the PDE. In (4) and (5), we use the intertwining property of the Radon transform (as discussed earlier) to simplify spatial partial derivatives.

Note: We have only taken the Radon transform of the PDE at one angle ω.

After applying the Radon transform to the PDE and its corresponding initial condition, we obtain the following:

$$\widehat{\underline{q}}_{,t} + \big(\cos(\omega)\ \underline{\underline{A}} + \sin(\omega)\ \underline{\underline{B}}\,\big)\,\widehat{\underline{q}}_{,s} = \underline{\underline{C}}\,\widehat{\underline{q}} \tag{9}$$

$$\widehat{\underline{q}}\,(t = 0, s, \omega) = \widehat{\underline{q_0}}\,(s, \omega) \tag{10}$$

Take $\widetilde{\underline{\underline{A}}}\,(\omega) := \cos(\omega)\ \underline{\underline{A}} + \sin(\omega)\,\underline{\underline{B}}$ and substitute $\widetilde{\underline{\underline{A}}}\,(\omega)$ into (6) to obtain the following:

$$\widehat{\underline{q}}_{,t} + \widetilde{\underline{\underline{A}}}\,(\omega)\,\widehat{\underline{q}}_{,s} = \underline{\underline{C}}\,\widehat{\underline{q}} \tag{11}$$

$$\widehat{\underline{q}}\,(t = 0, s, \omega) = \widehat{\underline{q_0}}\,(s, \omega) \tag{12}$$

3.3 Discretization of the PDE

3.3.1 Hyperbolicity

Recall that we are working with a system of first-order, linear hyperbolic PDEs. In this context, the system is **hyperbolic** if $\widetilde{\underline{\underline{A}}}\,(\omega)$ (which was defined earlier) is diagonalizable with only real eigenvalues for all $\omega \in \mathbb{R}$.

This means $\widetilde{\underline{\underline{A}}}\,(\omega)$ admits the following factorization:

$$\widetilde{\underline{\underline{A}}}\,(\omega) = \underline{\underline{P}}\,\underline{\underline{\Lambda}}\,\underline{\underline{P}}^{-1} \tag{13}$$

where

- $\underline{\underline{P}}$ is a matrix whose columns are the eigenvectors of $\widetilde{\underline{\underline{A}}}\,(\omega)$
- Λ is a diagonal matrix whose entries are the eigenvalues of $\widetilde{\underline{\underline{A}}}\,(\omega)$

3.3.2 Using the Hyperbolicity of the PDE

Recall that we have transformed our system of M first-order, linear hyperbolic PDEs into the following family of systems of first-order, linear hyperbolic PDEs

$$\underline{\widehat{q}}_{,t} + \underline{\underline{\widetilde{A}}}(\omega)\,\underline{\widehat{q}}_{,s} = \underline{\underline{C}}\,\underline{\widehat{q}} \tag{14}$$

$$\underline{\widehat{q}}\,(t=0,s,\omega) = \underline{\widehat{q_0}}\,(s,\omega) \tag{15}$$

for each angle ω.

We can use the eigendecomposition of $\underline{\underline{\widetilde{A}}}(\omega)$ to rewrite (11):

$$\underline{\widehat{q}}_{,t} + \underline{\underline{\widetilde{A}}}(\omega)\,\underline{\widehat{q}}_{,s} = \underline{\underline{C}}\,\underline{\widehat{q}} \tag{16}$$

$$\Rightarrow \underline{\widehat{q}}_{,t} + \underline{\underline{P}}\,\underline{\underline{\Lambda}}\,\underline{\underline{P}}^{-1}\,\underline{\widehat{q}}_{,s} = \underline{\underline{C}}\,\underline{\widehat{q}} \tag{17}$$

$$\Rightarrow \underline{\underline{P}}^{-1}\,\underline{\widehat{q}}_{,t} + \underline{\underline{\Lambda}}\,\underline{\underline{P}}^{-1}\,\underline{\widehat{q}}_{,s} = \underline{\underline{P}}^{-1}\,\underline{\underline{C}}\,\underline{\widehat{q}} \tag{18}$$

$$\Rightarrow \frac{\partial}{\partial t}\left(\underline{\underline{P}}^{-1}\,\underline{\widehat{q}}\right) + \underline{\underline{\Lambda}}\,\frac{\partial}{\partial s}\left(\underline{\underline{P}}^{-1}\,\underline{\widehat{q}}\right) = \underline{\underline{P}}^{-1}\,\underline{\underline{C}}\,\underline{\widehat{q}} \tag{19}$$

We can take $\underline{w} := \underline{\underline{P}}^{-1}\,\underline{\widehat{q}}$ to be the vector of **characteristic variables** and rewrite (16):

$$\underline{w}_{,t} + \underline{\underline{\Lambda}}\,\underline{w}_{,s} = \underline{\underline{P}}^{-1}\,\underline{\underline{C}}\,\underline{\underline{P}}\,\underline{w}$$

We can now introduce $\underline{\underline{F}} := \underline{\underline{P}}^{-1}\,\underline{\underline{C}}\,\underline{\underline{P}}$ and obtain the following family of systems of partially decoupled first-order, linear PDEs

$$\underline{w}_{,t} + \underline{\underline{\Lambda}}\,\underline{w}_{,s} = \underline{\underline{F}}\,\underline{w}$$
$$\underline{w}(t=0,s,\omega) = \underline{w_0}(s,\omega)$$

for each angle ω.

Note: The above system is generally only partially decoupled. The system is fully decoupled if $\underline{\underline{C}} \equiv \underline{\underline{0}}$.

3.3.3 Spatial Discretization of the PDE

Recall for a fixed angle ω, we are working with the following system of first-order, linear hyperbolic PDEs:

$$\underline{w}_{,t} + \underline{\underline{\Lambda}}\,\underline{w}_{,s} = \underline{\underline{F}}\,\underline{w}$$
$$\underline{w}(t=0,s,\omega) = \underline{w_0}(s,\omega)$$

The p^{th} equation in the system is given explicitly as follows:

$$w_{p,t} + \lambda_p w_{p,s} = \sum_{q=1}^{M} F_{pq} w_q$$

We need to discretize the above system to computationally work with it.

We do this by discretizing the p^{th} equation in space, and we naturally approximate w_p at N_s many Chebyshev points of the second kind along the line determined by the angle ω:

$$w_p \to \underline{W_p} \in \mathbb{R}^{N_s} \tag{20}$$

Thus,

$$w_{p,t} + \lambda_p w_{p,s} = \sum_{q=1}^{M} F_{pq} w_q \quad \to \quad \underline{W_p}_{,t} + \lambda_p\,\underline{W_p}_{,s} = \sum_{q=1}^{M} F_{pq}\,\underline{W_q} \tag{21}$$

We now need to discretize the spatial derivative operator, and we naturally use the Chebyshev spectral differentiation matrix $\underline{\underline{D}} \in \mathbb{R}^{N_s \times N_s}$

Thus,

$$\underline{W_p}_{,t} + \lambda_p \underline{W_p}_{,s} = \sum_{q=1}^{M} F_{pq} \underline{W_q} \quad \rightarrow \quad \underline{W_p}_{,t} + \lambda_p \underline{\underline{D}} \underline{W_p} = \sum_{q=1}^{M} F_{pq} \underline{W_q} \tag{22}$$

As usual, we need to also enforce boundary conditions (in particular, Dirichlet boundary conditions). Since we are working a system of hyperbolic PDEs, we enforce inflow conditions:

- If $\lambda_p > 0$, we enforce left inflow boundary conditions
- If $\lambda_p < 0$, we enforce right inflow boundary conditions

We can enforce the above inflow boundary conditions by modifying the appropriate columns and rows of $\underline{\underline{D}}$ in the p^{th} equation:

- If $\lambda_p > 0$, we zero out the last column and last row of $\underline{\underline{D}}$
- If $\lambda_p < 0$, we zero out the first column and first row of $\underline{\underline{D}}$

Note: The above enforcement scheme makes sense in this context because the Chebyshev points of the second kind are ordered in *decreasing* order from 1 to -1.

Thus,

$$\underline{W_p}_{,t} + \lambda_p \underline{\underline{D}} \underline{W_p} = \sum_{q=1}^{M} F_{pq} \underline{W_q} \quad \rightarrow \quad \underline{W_p}_{,t} + \lambda_p \underline{\underline{D_p}} \underline{W_p} = \sum_{q=1}^{M} F_{pq} \underline{W_q} \tag{23}$$

3.3.4 Temporal Discretization of the PDE

Recall that we spatially discretized the p^{th} equation in the above system to obtain

$$\underline{W_p}_{,t} + \lambda_p \underline{\underline{D_p}} \underline{W_p} = \sum_{q=1}^{M} F_{pq} \underline{W_q} \tag{24}$$

We have not yet temporally discretized the system, but we have more freedom with picking time-stepping methods. For this project, we adapted a **third-order**, **L-stable** IMEX Runge-Kutta scheme presented in [2]. The scheme is given as follows:

- We have knowledge of the solution vectors $\underline{W_p}^{(n)}$ at some initial time-step n for each equation $p = 1, 2, \ldots, M$
- In the first stage, we perform the following updates for equations $p = 1, 2, \ldots, M$:

$$\underline{W_p}^{(1)} = \underline{W_p}^{(n)} - \Delta t \, a_{11} \, \lambda_p \underline{\underline{D_p}} \underline{W_p}^{(1)} \tag{25}$$

- For each stage $i = 2, \ldots, \nu$ (where $\nu = 4$ is the last stage), we perform the following updates for equations $p = 1, 2, \ldots, M$:

$$\underline{W_p}^{(i)} = \underline{W_p}^{(n)} + \Delta t \sum_{l=1}^{i-1} \widetilde{a}_{il} \sum_{q=1}^{M} F_{pq} \underline{W_q}^{(l)} - \Delta t \sum_{l=1}^{i} a_{il} \, \lambda_p \underline{\underline{D_p}} \underline{W_p}^{(l)} \tag{26}$$

- To obtain the solution vectors $\underline{W_p}^{(n+1)}$ at the next time-step $(n+1)$, we perform the following updates for equations $p = 1, 2, \ldots, M$:

$$\underline{W_p}^{(n+1)} = \underline{W_p}^{(n)} + \Delta t \sum_{i=1}^{\nu} \widetilde{w}_i \sum_{q=1}^{M} F_{pq} \underline{W_q}^{(i)} - \Delta t \sum_{i=1}^{\nu} w_i \lambda_p \underline{\underline{D_p}} \underline{W_p}^{(i)} \tag{27}$$

3.3.5 Solving the System via the Matrix Exponential

We can also solve the above system using the matrix exponential. Recall that the p^{th} equation in the spatially discretized system is given by

$$\underline{W_p}_{,t} + \lambda_p \underline{\underline{D_p}}\, \underline{W_p} = \sum_{q=1}^{M} F_{pq}\, \underline{W_q} \tag{28}$$

We can rearrange the above to obtain

$$\underline{W_p}_{,t} = -\lambda_p \underline{\underline{D_p}}\, \underline{W_p} + \sum_{q=1}^{M} F_{pq}\, \underline{W_q} \tag{29}$$

We can then compute the solution of the entire system at the final time t_f via the matrix exponential:

$$\begin{bmatrix} \underline{W_1} \\ \underline{W_2} \\ \vdots \\ \underline{W_M} \end{bmatrix} = \exp\left(-t_f \begin{bmatrix} -\lambda_1 \underline{\underline{D_1}} + F_{11}\, \underline{\underline{I}} & F_{12}\, \underline{\underline{I}} & \cdots & F_{1m}\, \underline{\underline{I}} \\ F_{21}\, \underline{\underline{I}} & -\lambda_2 \underline{\underline{D_2}} + F_{22}\, \underline{\underline{I}} & \cdots & F_{2m}\, \underline{\underline{I}} \\ \vdots & \vdots & \ddots & \vdots \\ F_{m1}\, \underline{\underline{I}} & \cdots & F_{m(m-1)}\, \underline{\underline{I}} & -\lambda_m \underline{\underline{D_m}} + F_{mm}\, \underline{\underline{I}} \end{bmatrix} \right) \underline{W_0}$$

where

$$\underline{W_0} = \begin{bmatrix} \underline{W_1}^{(0)} \\ \underline{W_2}^{(0)} \\ \vdots \\ \underline{W_m}^{(0)} \end{bmatrix} \in \mathbb{R}^{(M \cdot N_s)} \tag{30}$$

is a vector of discretized initial conditions corresponding to the M equations in the system.

4 Results

4.1 Examples

We first demonstrate a qualitative example of this approach applied to a 2D wave equation with a radially symmetric initial condition.

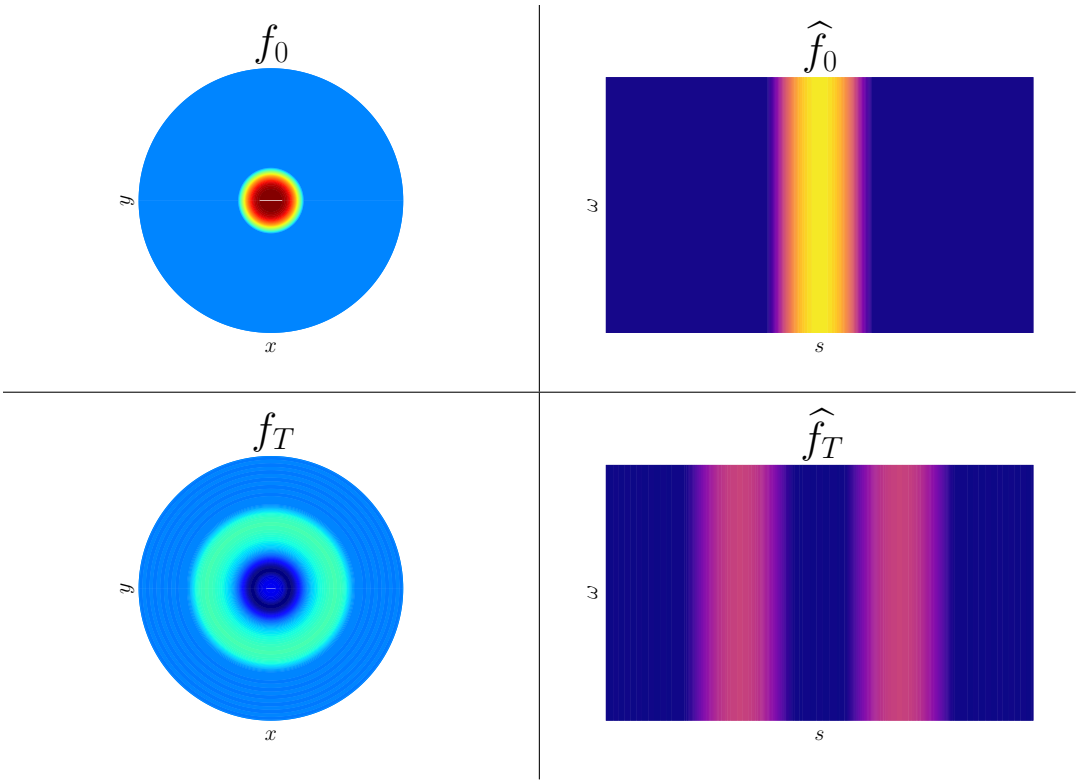


Figure 3: Wave equation initial conditions (top) and final solution (bottom) in physical (left) and Radon space (right)

In the above figures, we can observe the effect of Radial symmetry on our method. In particular, each sample of radon space is identical, as the transform is independent of the angle ω. Moreover, the inversion process is much simpler, as the Radon transform matrix $\underline{\underline{R}}$ is initially well conditioned and can be simply inverted.

We can repeat this process for the radially symmetric P_1 equations. We use the standard initial condition, the following approximate to a Dirac delta function with $\alpha = 0.03$:

$$q_0(x, y) = \frac{1}{4\pi\alpha^2} e^{\frac{-(x^2+y^2)}{4\alpha^2}}$$

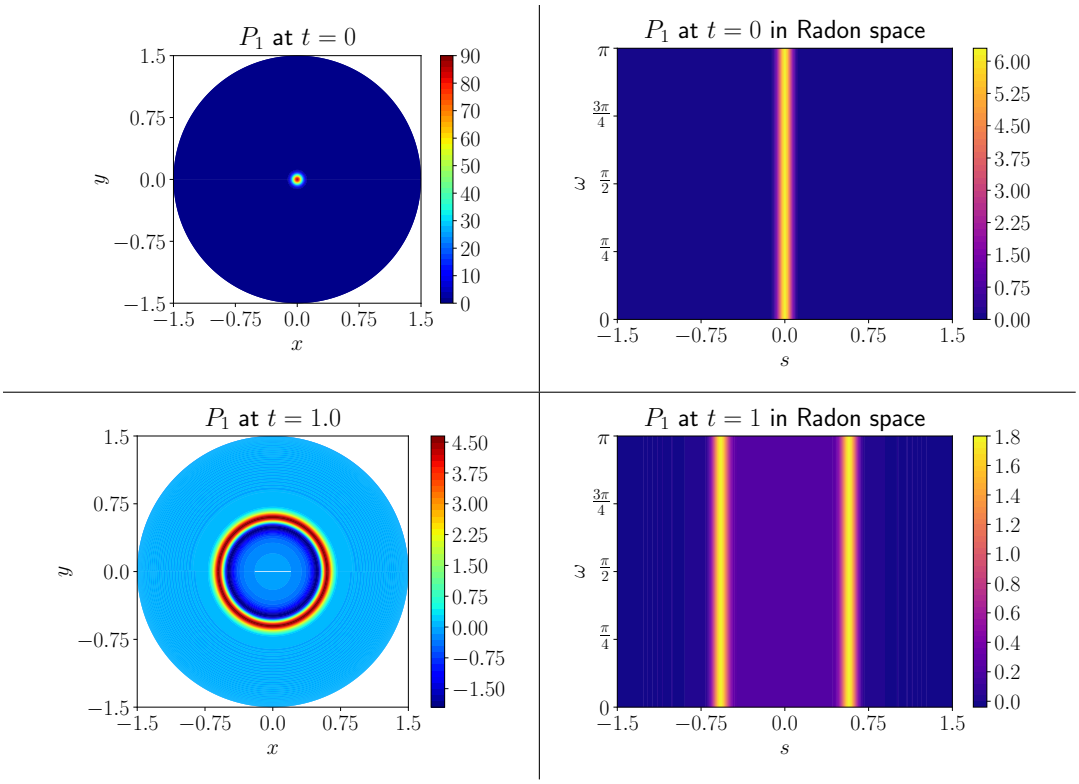
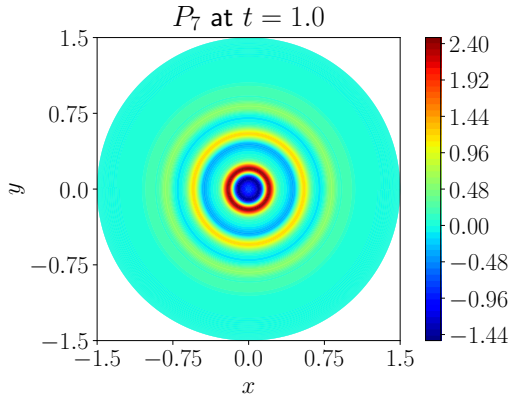
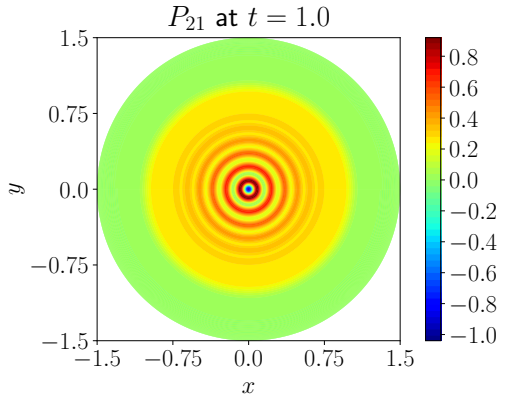
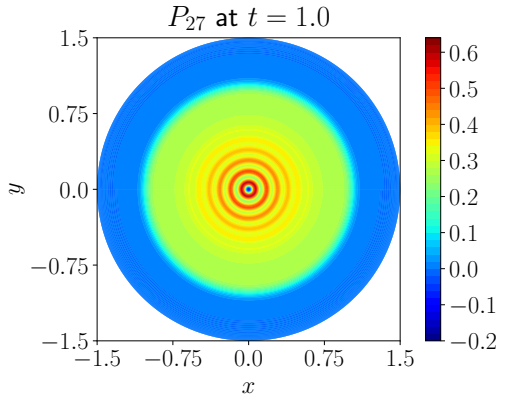


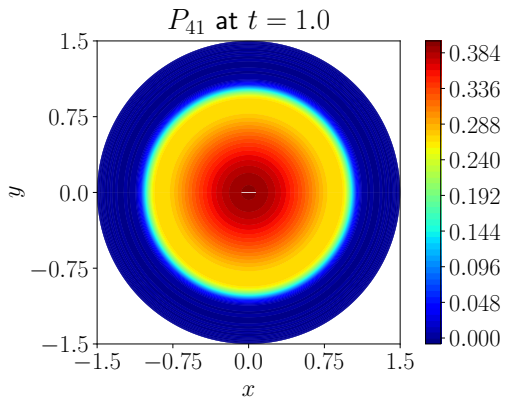
Figure 4: P_1 initial conditions (top) and final solution (bottom) in physical (left) and Radon space (right)

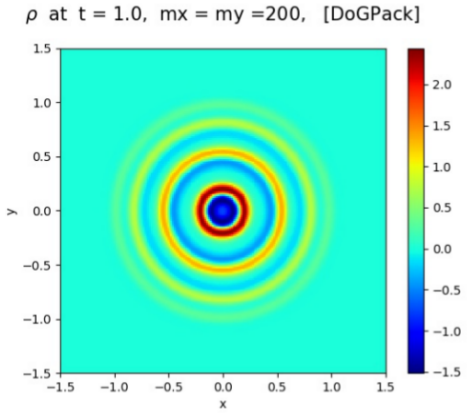
Because the P_N equations themselves are only approximations of radiative transfer, we increase N and compare our results to those in[5].

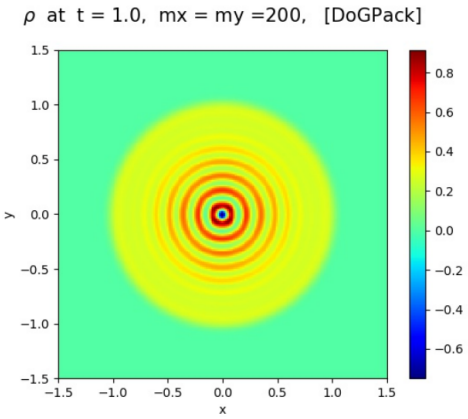

P_7 at $t = 1.0$
1.5
0.75
0.0
−0.75
−1.5
y
−1.5 −0.75 0.0 0.75 1.5
x
2.40
1.92
1.44
0.96
0.48
0.00
−0.48
−0.96
−1.44

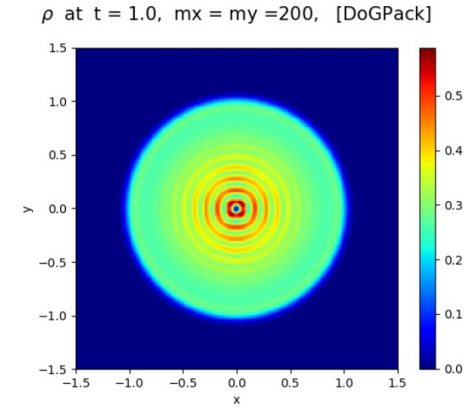

P_{21} at $t = 1.0$
1.5
0.75
0.0
−0.75
−1.5
y
−1.5 −0.75 0.0 0.75 1.5
x
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0

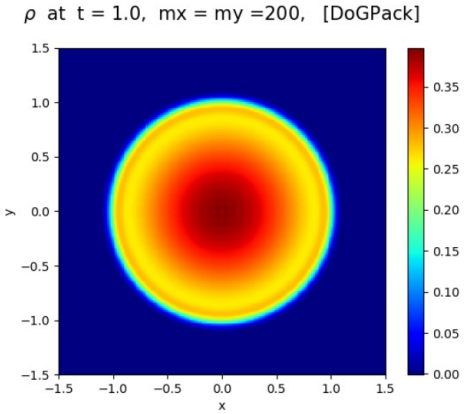

P_{27} at $t = 1.0$
1.5
0.75
0.0
−0.75
−1.5
y
−1.5 −0.75 0.0 0.75 1.5
x
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2


P_{41} at $t = 1.0$
1.5
0.75
0.0
−0.75
−1.5
y
−1.5 −0.75 0.0 0.75 1.5
x
0.384
0.336
0.288
0.240
0.192
0.144
0.096
0.048
0.000


ρ at t = 1.0, mx = my =200, [DoGPack]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5


ρ at t = 1.0, mx = my =200, [DoGPack]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


ρ at t = 1.0, mx = my =200, [DoGPack]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x
0.5
0.4
0.3
0.2
0.1
0.0


ρ at t = 1.0, mx = my =200, [DoGPack]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

4.2 Convergence

In general, forming exact solutions to the P_N equations is difficult. In place of an exact solution, a high-resolution numerical solution to the radially symmetric P_1 equations is found with the following system of PDEs. This formulation takes advantage of the radial symmetry to rewrite the system in polar coordinates, resulting in a 1D slice of the solution. This PDE can be solved through any of the standard timestepping methods discussed above. For their use as a comparison tool, this reference solution is interpolated across the 1D domain. The solution in Radon space can then be obtained by performing a forward Radon transform of this physical solution.

$$\begin{bmatrix} p \\ u_r \end{bmatrix}_{,t} + \begin{bmatrix} 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & 0 \end{bmatrix} \begin{bmatrix} p \\ u_r \end{bmatrix}_{,r} = \begin{bmatrix} -\frac{1}{\sqrt{3}} \frac{1}{r} u_r \\ -\sigma u_r \end{bmatrix}$$

To simplify the convergence studies, only a single slice of the solution obtained through the Radon method is compared. When the L_∞ norm is used, the error between the reference solution and the approximated solution is identical between any diameter of the mesh. Thus the L_∞ error across any single diameter is equal to that over the entire domain. Additionally, radial symmetry indicates that the discretization in ω is arbitrary, and the following convergence test only takes into account the discretization across each diameter.

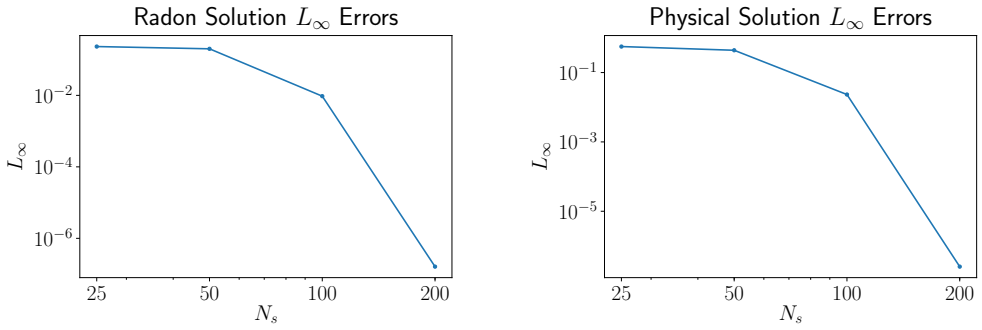


Figure 5: L_∞-Norm for radially symmetric initial conditions

As expected, the convergence of the overall method is spectrally accurate in N_s. This is because the only error present in the solution comes from insufficiently approximating the initial condition. All other calculations, such as the interpolation, spectral differentiation, and quadrature, are all performed with spectral accuracy.

Computing convergence rates precisely is more difficult for non-radially symmetric problems, as there are no exact solutions to reference. To counteract this, the same reference solution for the symmetric case is used, but shifted in the positive x direction by some constant. Then, the experimental P_1 solution can be found by translating the initial condition by the same amount, and considering the cross section across the x-axis. While not a precise convergence study, one can still see that the solution approaches the true value. For these problems, the effect of N_s and N_ω must be considered independently of one another.

We begin by fixing $N_\omega = 200$, and varying N_s, as the timestepping is independent of the problem's symmetry. That is to say, the solution in Radon space should be as accurate as the symmetric case.

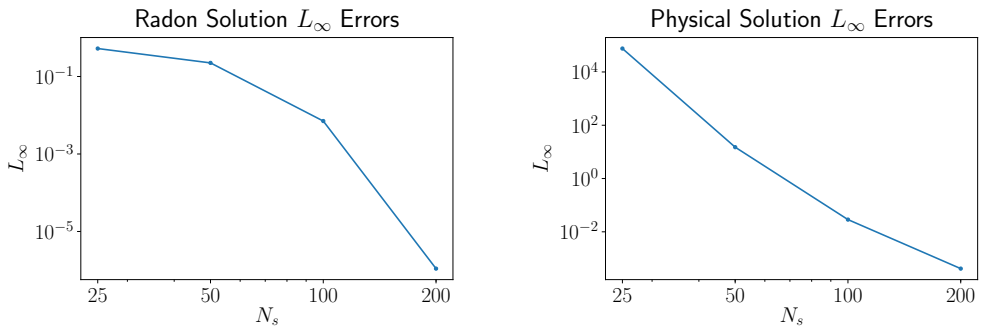


Figure 6: L_∞-Norm for radially symmetric initial conditions

As we see, this is the case. We also observe that now the solution in physical space no longer has the same rate of accuracy, as the solution is limited by N_ω.

We now perform the same test, but fixing N_s and varying N_ω. Across each of these examples, the solution in Radon space is identical, as the timestepping is once again dependent only on the discretization along each diameter. However, the solution in physical space is heavily dependent on the number of N_ω. We expect the solution to be fourth-order accurate, as this is the lowest convergence rate across all the component methods (owing to the radial interpolation across diameters).

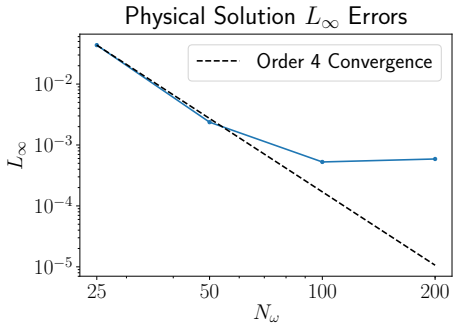


Figure 7: L_∞-Norm for radially symmetric initial conditions

In practice however, we observe that while the error in the solution does decrease, it does not decrease quite as expected. While the error is already relatively low even for low values of N_ω, the desired convergence rate is only observed across the first two experiments. Moreover the reasons for this are largely unknown at this time. It is suspected that, rather than the method simply not being as accurate as hypothesized, there is some hidden source of error that is limiting the efficacy of the inverse Radon transform. Possible sources of this error are a failure to accurately capture the initial condition with $N_s = 200$ or a not sufficiently accurate timestepping method. It is also possible that there is an issue with the reference solution, which is, again, only a high-resolution approximation of the true solution. It is our hope that by addressing some or all of these issues, we can obtain results consistent with our overarching theory. That being said, the potential payoffs of this method are great, and we hope that further investigation will illuminate this.

Summary

To summarize the project, the RT was used to bring the P_N equations[1] into radon space to solve a family of one dimensional PDEs instead of a two or more-dimensional PDE in real space. Then, transport and time-stepping[2] was used to solve the hyperbolic PDEs. Last, the IRT was used to take the solution from radon space back to physical space.

Future Work

For future work, there are a couple of routes that we would suggest. First, adding spatially dependent collision terms to the P_N equations. The scope of this project did not cover those terms and assumed that any collision term was zero. Next, a more sophisticated or higher order time-stepping scheme could be used to reduce any error that may have been encountered. Finally, parallelization could be used to speed up the code that was written. This could be accomplished by parallelizing the RT as well as parallelizing the separate angles.

Acknowledgments

We would like to acknowledge the people and organizations that supported us through this research. These people and organizations include: the NSF and the grant (DMS-1457443) that allowed us to travel to Iowa State University (ISU); ISU for hosting us for our time in Iowa; and most of all, to Dr. Rossmanith, our faculty mentor, and Christine Wiersma, our graduate student mentor, who were crucial to the completion of this research.

References

[1] Brunner, T. and J. Holloway (2005), "Two-dimensional time dependent riemann solvers for neutron transport." *Journal of Computational Physics*, 210, 386–399.

[2] Pieraccini, S. and G. Puppo (2007), "Implicit-explicit schemes for bgk kinetic equations." *Journal of Scientific Computing*, 32.

[3] Press, W. (2006), "Discrete radon transform has an exact, fast inverse and generalizes to operations other than sums along lines." *PNAS*, 103, 19249–19254.

[4] Rim, D. (2018), "Dimensional splitting of hyperbolic partial differential equations using the radon transform." *SIAM Journal of Scientific Computing*, 40, A4184–A4207.

[5] Shin, M. (2019), "Hybrid discrete (h_n^t) approximations to the equation of radiative transfer." Ph.d. thesis, Iowa State Univeristy.

[6] Trefethen, L. (2000), *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics.