# 2-Dimensional Radiative Transfer Equation

The focus of my thesis is studies, parallel implementations, and numerical methods for the Radiative Transfer Equation (RTE).

**Abstract.**
Radiative transfer is the physical occurrence of energy transfer, in the form of electromagnetic radiation or light. The propagation of this light throughout a medium is affected by many things, mainly absorption, scattering, and emission at any internal point. The Radiative Transfer Equation (RTE) describes a mathematical model for these interactions, but algorithms built to solve this equation have proven to be quite complex. We present an overview of two different approaches to solve the RTE.

**Introduction.**

Consider a two-dimensional blob with the following properties:

(1) The material the blob is made out of has uniform consistency and density.
(2) The blob is not moving or changing shape.
(3) As light passes through the blob, it may be absorbed or scattered by the material.

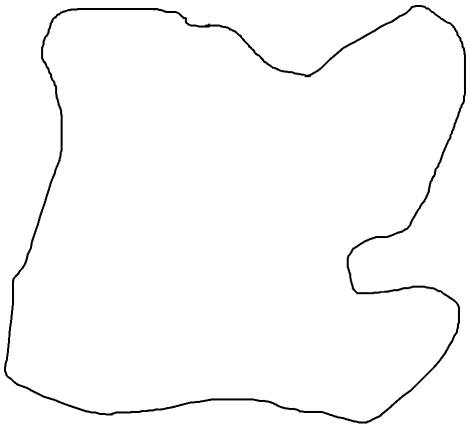


FIGURE 1. A 2-Dimensional Blob

Define the field of radiation / light moving in point $x$ in direction $\omega$ at time $t$, as the **spectral radiance**, represented by $u(x, \omega, t)$.

Suppose that $f(x, \omega)$ describes the spectral radiance of the source, and that $g(x, \omega)$ describes the light impinging on the boundary of the blob.

The radiative transfer equation deals with how this light source will move throughout the blob, at time $t$, position $x$, and direction $\omega$. We can make another assumption - that enough time has passed such that our blob is in a steady-state - although photons will still be moving, the distribution of them throughout our substance is at equilibrium.

Let $X$ be the region representing our physical blob. Assume $u(x, \omega)$ satisfies the RTE:

$$\omega \cdot \nabla u + \mu_t u = \mu_s \int_{S'} N(\omega, \omega') u(x, \omega)' d\omega' + f(x, \omega)$$

at all points $x \in X$, and all $\omega \in S'$.

Furthermore, assume $u$ satisfies the boundary condition

$$u(x, \omega) = g(x, \omega)$$

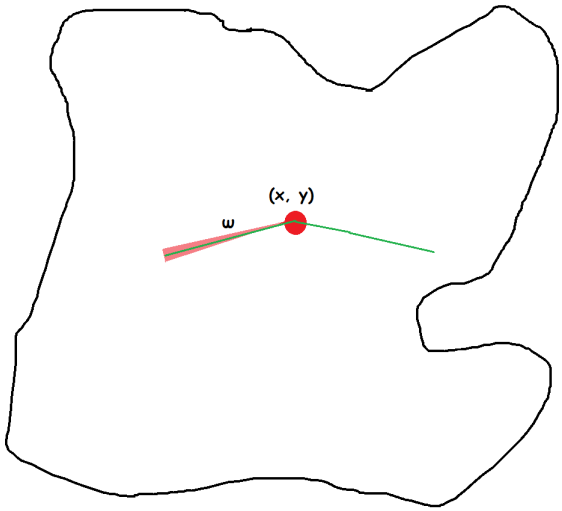
if $(x, \omega) \in \Gamma^-$:

$$\Gamma^- = \{(x, \omega) \,|\, x \in dx, \overrightarrow{n}(x) \cdot \overrightarrow{w} < 0\}$$

and $\overrightarrow{n}(x)$ is the outward pointing normal vector.

In the RTE, $\mu_a$ represents the absorption coefficient, or the likelihood that any particular point inside the blob might absorb photons that are present. Similarly, $\mu_s$ represents the scattering coefficient, or the likelihood that any particular point inside the blob might scatter incoming photons to other directions (or from other directions to itself).

Lastly, $N(\omega, \omega')$ represents the probability that a photon moving in direction $\omega$ scatters to direction $\omega'$.




It should be noted that for this problem, we started out with a generic circle of radius 1, centered around the origin, as the "blob", though we will branch out to more complex shapes later.

## 1. Methods / Implementation

We constructed 2 finite difference methods to approximate a solution to this equation - an implicit method, and an iterative method.

### 1.1. **Implicit Approach.**

To construct a linear system of equations to approximate a solution to the RTE, we replaced $u_x$ and $u_y$ with centered finite difference approximations, the integral term with a quadrature rule, and require that at all $i$,$j$,$k$:

$$\omega_1^k \left( \frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\delta x} \right) + \omega_2^k \left( \frac{u_{i,j+1,k} - u_{i,j-1,k}}{2\delta y} \right) + \mu_t u_{i,j,k} = \mu_s \left( \sum_l W_l \cdot g(\omega_k, \omega_l) \cdot u_{i,j,l} \right) + f(x_i, y_j, \omega_k)$$

We may write the linear system of equations as:

$$D_x \overrightarrow{u} + D_y \overrightarrow{u} + \mu_t \overrightarrow{u} = \mu_s I + f, \tag{1}$$

For the central-difference approximations of $u_x$ and $u_y$ above, it was necessary to perform one step of forward difference at the boundary points, to not "go off" of the grid.

The equation (1), expanded out, is:

$$\begin{bmatrix} \cos(\omega_1) & 0 & 0 & \dots & 0 \\ 0 & \cos(\omega_1) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \cos(\omega_w) & 0 \\ 0 & \dots & \dots & 0 & \cos(\omega_w) \end{bmatrix} \cdot \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & -1 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & \dots 0 & 1 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \dots & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \overrightarrow{u} + \begin{bmatrix} \sin(\omega_1) & 0 & 0 & \dots & 0 \\ 0 & \sin(\omega_1) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \sin(\omega_w) & 0 \\ 0 & \dots & \dots & 0 & \sin(\omega_w) \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 & \dots & -1 & \dots & 0 \\ 0 & -1 & 0 & \dots & -1 & \dots & 0 \\ 0 & 0 & -1 & \dots & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & \dots 1 & 0 & \dots & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \dots & 0 & -1 & \dots & 0 & 0 & 1 \end{bmatrix} \overrightarrow{u} - mu_s \cdot I = f$$

Our forcing function $f(x, y, \omega)$ is a relatively simple term - like our solution $\overrightarrow{u}$, it is an $(MNW \times 1)$ vector.

Of all the terms, the integral term $I$ is the most complicated. We see that for each individual point, we have an integral representation, which we can approximate with a quadrature rule:

$$I = \begin{bmatrix} \mu_s \int u(1,1,\omega') \cdot g(\omega_1, \omega') d\omega' \\ \mu_s \int u(2,1,\omega') \cdot g(\omega_1, \omega') d\omega' \\ \vdots \\ \mu_s \int u(1,1,\omega') \cdot g(\omega_2, \omega') d\omega' \\ \mu_s \int u(2,1,\omega') \cdot g(\omega_2, \omega') d\omega' \\ \vdots \\ \mu_s \int u(m,n,\omega') \cdot g(\omega_n, \omega') d\omega' \end{bmatrix} = \begin{bmatrix} \mu_s \left( \sum_{k=1}^{N_\omega} r_k \cdot u(1,1,\omega_k) g(\omega_1, \omega_k) \right) \\ \mu_s \left( \sum_{k=1}^{N_\omega} r_k \cdot u(2,1,\omega_k) g(\omega_1, \omega_k) \right) \\ \vdots \\ \mu_s \left( \sum_{k=1}^{N_\omega} r_k \cdot u(1,1,\omega_k) \cdot g(\omega_2, \omega_k) \right) \\ \mu_s \left( \sum_{k=1}^{N_\omega} r_k \cdot u(2,1,\omega_k) \cdot g(\omega_2, \omega_k) \right) \\ \vdots \\ \mu_s \left( \sum_{k=1}^{N_\omega} r_k \cdot u(m,n,\omega_k) \cdot g(\omega_{N_\omega}), \omega_k) \right) \end{bmatrix} \tag{2}$$

The integral term for each spatial point is calculated via a quadrature rule, equally weighting all directions.

Let $N_\omega$ be the number of angles we are considering for this problem, and $r_k$ be a specific weight (defaulted to $\frac{1}{N_\omega}$) to put on each summand term. $I$ is transformed from a vector to a sparse diagonal $(MNW \times MNW)$ matrix.

We no consider the boundary conditions. If $(x, \omega)$ was on the inflow boundary (determined by tolerance within spatial distance from $x$), $\overrightarrow{n}(\overrightarrow{x}) \cdot \overrightarrow{\omega} < 0$, we set it equal to our boundary condition. If on the outflow boundary, or the interior of the domain, follow the equation (1).

Below is an example result from this algorithm, where $f(x,\omega) = \sqrt{(x+0.5)^2+(y+0.5)^2}$, $g(x,\omega) = 0$, $mu_a = 0.3$, $mu_s = 0.1$, $nx = ny = 100$, and $nw = 8$.

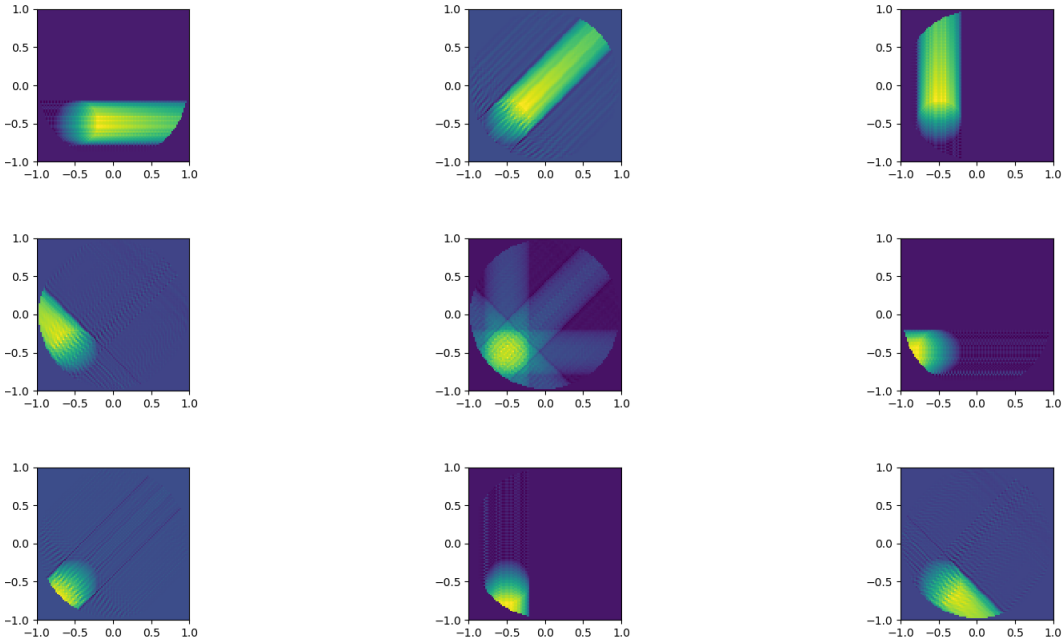
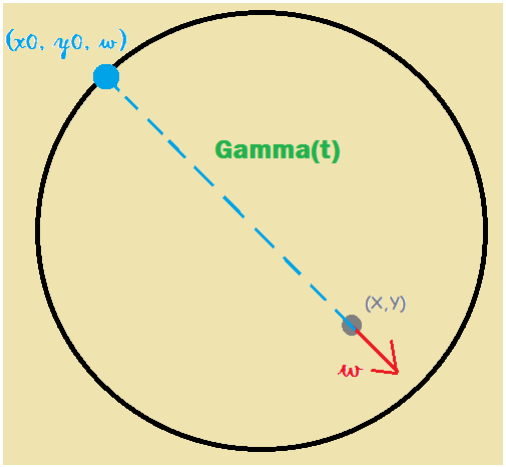


FIGURE 2. Example implementation of implicit method, for $f(x,\omega) = 1$ if $(x+0.5)^2+(y+0.5)^2 <= 0.1$. The middle plot shows all directions summed up, while outer edges each correspond to 1 of 8 directions (0 through $\frac{7\pi}{4}$).

## 2. Iterative Method

For the iterative method, we laid out a grid on a square, covering X, and approximated the solution's value for every $(x, \omega)$ in our grid.

First, we cover the case without scattering:




For a case without scattering, our equation becomes:

$$\omega \cdot \nabla u + \mu_a \cdot u = f$$

Given that a (boundary) light source comes from the inflow boundary, we want to trace back our point $(x, y)$, in direction $-\omega$, in order to find the point on the boundary of the circle $(x_0, y_0)$, travelling in the same direction $\omega$, that would eventually reach our point, as illustrated above.

Let $(x_0, y_0)$ be our traced-back boundary point, and $\Gamma(t) = \overrightarrow{x} + \overrightarrow{\omega} t$ be the paramaterization of the line between $(x_0, y_0)$ and $(x, y)$, travelling from the former toward the latter.

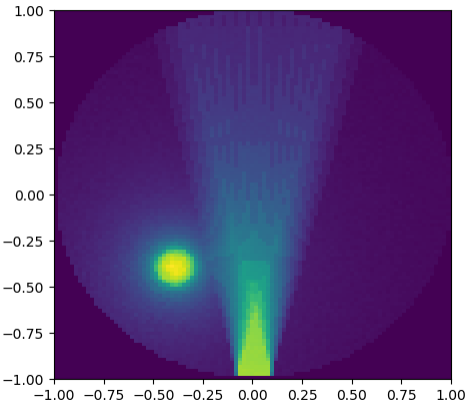
Let $g(t) = u(\Gamma(t), \omega)$. We can see that $\frac{\partial g}{\partial t} = \omega \cdot \nabla u(\Gamma(t), \omega)$, and so:

$$\frac{\partial g}{\partial t}(t) + \mu_a g(t) = f(\Gamma(t), \omega)$$

The solution is (via integrating factor):

$$u(\Gamma(t), \omega) = g(t) = e^{-\mu_a \cdot t} \left( \int_0^t e^{\mu_a \cdot s} f(\Gamma(s), \omega) \partial s \right) + u(x_0, y_0). \tag{3}$$

Where $t$ is the traced-back distance to the boundary of the circle. An example result of this algorithms implementation is shown below, with: $f(x,\omega) = \frac{1}{\sqrt{(x+\frac{1}{3})^2+(y+\frac{1}{3})^2}}$, and $g(x,\omega) = 1$ if $x_2 < 0$ and $|x+1| < 0.1$:




To implement a scattering term, we took our absorption-only method, and introduced the following iterative scheme:

$$u^{(0)}(x,\omega) = 0$$
$$\omega \cdot \nabla u^{(n+1)} + \mu_t u^{(n+1)} = \mu_s \int_{S'} u^{(n)} g(\omega,\omega')\partial\omega' + f \tag{4}$$

Because $u^{(n)}$ is known, we can consider the entire RHS of (4) a fixed forcing function, and solve for $u^{(n+1)}$ using $u^{(n)}$. Once we have the value of $u^{(n+1)}$ at the grid points, we can interpolate to get a value of $u_{(n+1)}(x,\omega)$ at all points $(x,\omega)$. We iterate until $||u^{(n)} - u^{(n-1)}||_\infty < \epsilon$.

## 3. Truncation Error

In order to verify that our finite difference scheme was accurately approximating the solution to the RTE, we estimated the local truncation error of our finite difference scheme as follows[1].

Let $F_{i,j,k}(u) = 0$ represent the difference equation approximating the RTE at the $(i,j,k)th$ mesh point, with exact solution $u$. If we replace $u$ with the exact solution to the RTE, $U$, then $F_{i,j,k}(U)$ measures the amount by which the exact solution does not satisfy our finite difference scheme at point $(i,j,k)$. The value of $F_{i,j,k}(U)$ at point $(i,j,k)$ is known as the local truncation error $T_{i,j,k}$.

$$F_{i,j}(u) = \omega_x \cdot \left(\frac{u_{i+1,j,k} - u_{i-1,j,k}}{2h}\right) + \omega_y \cdot \left(\frac{u_{i,j+1,k} - u_{i,j-1,k}}{2h}\right) + \mu_t u_{i,j,k} - f_{i,j,k} \tag{5}$$

Therefore,

$$T_{i,j,k} = F_{i,j,k}(U) = \omega_x \cdot \left(\frac{U_{i+1,j,k} - U_{i-1,j,k}}{2h}\right) + \omega_y \cdot \left(\frac{U_{i,j+1,k} - U_{i,j-1,k}}{2h}\right) + \mu_t U_{i,j,k} - f_{i,j,k} \tag{6}$$

By Taylor's expansion, for spatial step size $h$,

$$\begin{aligned}
U_{i+1,j,k} &= U(x_i + h, y_j, \omega_k) \\
&= U_{i,j,k} + h\left(\frac{\partial U}{\partial x}\right)_{i,j,k} + \frac{h^2}{2}\left(\frac{\partial^2 U}{\partial x^2}\right)_{i,j,k} + \frac{h^3}{6}\left(\frac{\partial^3 U}{\partial x^3}\right)_{i,j,k} + ... \\
U_{i-1,j,k} &= U(x_i - h, y_j, \omega_k) \\
&= U_{i,j,k} - h\left(\frac{\partial U}{\partial x}\right)_{i,j,k} + \frac{h^2}{2}\left(\frac{\partial^2 U}{\partial x^2}\right)_{i,j,k} - \frac{h^3}{6}\left(\frac{\partial^3 U}{\partial x^3}\right)_{i,j,k} + ... \\
U_{i,j+1,k} &= U(x_i, y_j + h, \omega_k) \\
&= U_{i,j,k} + h\left(\frac{\partial U}{\partial y}\right)_{i,j,k} + \frac{h^2}{2}\left(\frac{\partial^2 U}{\partial y^2}\right)_{i,j,k} + \frac{h^3}{6}\left(\frac{\partial^3 U}{\partial y^3}\right)_{i,j,k} + ... \\
U_{i,j-1,k} &= U(x_i - h, y_j, \omega_k) \\
&= U_{i,j,k} - h\left(\frac{\partial U}{\partial y}\right)_{i,j,k} + \frac{h^2}{2}\left(\frac{\partial^2 U}{\partial y^2}\right)_{i,j,k} - \frac{h^3}{6}\left(\frac{\partial^3 U}{\partial y^3}\right)_{i,j,k} + ...
\end{aligned}$$

[1]Smith, G. D. (1985). Finite difference methods (Third ed., Oxford Applied Mathematics and Computing Science). Clarendon. 38-40. Print.

Substitution into the expression for $T_{i,j,k}$ gives:

(7)

$$T_{i,j} = \frac{\omega_x}{2h}\left(2h\left(\frac{\partial U}{\partial x}\right)_{i,j,k} + \frac{2h^3}{6}\left(\frac{\partial^3 U}{\partial x^3}\right)_{i,j,k} + ...\right) + \frac{\omega_y}{2h}\left(2h\left(\frac{\partial U}{\partial y}\right)_{i,j,k} + \frac{2h^3}{6}\left(\frac{\partial^3 U}{\partial y^3}\right)_{i,j,k} + ...\right) + \mu_t U_{i,j,k} - f_{i,j,k}$$

$$= \omega_x\left(\left(\frac{\partial U}{\partial x}\right)_{i,j,k} + \frac{h^2}{6}\left(\frac{\partial^3 U}{\partial x^3}\right)_{i,j,k} + ...\right) + \omega_y\left(\left(\frac{\partial U}{\partial y}\right)_{i,j,k} + \frac{h^2}{6}\left(\frac{\partial^3 U}{\partial y^3}\right)_{i,j,k} + ...\right) + \mu_t U_{i,j,k} - f_{i,j,k}$$

$$= \left(\omega_x\left(\frac{\partial U}{\partial x}\right)_{i,j,k} + \omega_y\left(\frac{\partial U}{\partial y}\right)_{i,j,k} + \mu_t U_{i,j,k} - f_{i,j,k}\right) + \frac{h^2}{6}\left(\frac{\partial^3 U}{\partial x^3}\right)_{i,j,k} + \frac{h^2}{6}\left(\frac{\partial^3 U}{\partial y^3}\right)_{i,j,k} + ...$$

Since U is the solution of the RTE:

$$\left(\omega_x\left(\frac{\partial U}{\partial x}\right) + \omega_y\left(\frac{\partial U}{\partial y}\right) + \mu_t U - f\right)_{i,j,k} = 0$$

Hence,

$$T_{i,j} = O(h^2).$$

for our given scheme, without the integral term. The truncation error of the integral term:

$$\mu_s \int_{S'} n(\omega, \omega') u(x, \omega') d\omega'$$

relies on the truncation error of the trapezoidal rule used to approximate its value. Given $k$ directional points, the local truncation error of the integral is $O(k^2)$.[2]

There is one final consideration as to truncation error - for the implicit approach, a central difference approximation could not be performed at every mesh point, since we cannot roll off the boundary of the mesh. A single finite difference step was used to approximate these points, which makes $T_{i,j} = O(h)$, although the majority of points were calculated with a central difference method.

$O(h)$ truncation error is not the most appealing order of error to have, but it does provide conformation that our finite difference scheme is consistent- that as $\lim_{h,k\to 0}$, that $T_{i,j} \to 0$.

[2]Atkinson, Kendall E. (1989), An Introduction to Numerical Analysis (2nd ed.), New York: John Wiley Sons, ISBN 978-0-471-50023-0

## 4. Multigrid Method

For the iterative approach outlined above, with scattering, we focused on finding a better initial guess to the fixed point approximation than $\overrightarrow{0}$. One solution is the multigrid approach.

Instead of starting on the original mesh size with an initial guess of $\overrightarrow{0}$, we approximated the solution on a much coarser mesh. We interpolated that solution up to a finer mesh size, used it so approximate the solution to the finer mesh, repeatedly until we reached the desired mesh size.

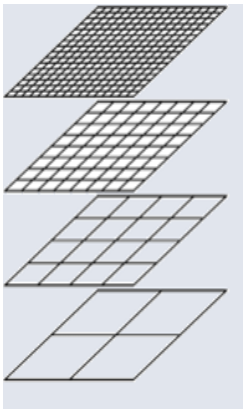


Figure 3. Example of coarse mesh scaling

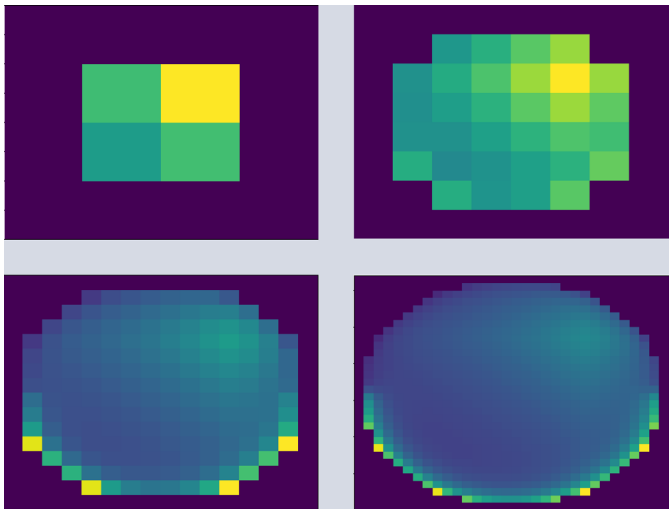


Figure 4. Example of Coarse Mesh Interpolating

Figure 3 shows the thought process behind scaling up a coarse grid. Figure 4 shows the intermediate steps of this algorithm on a $4 \times 4$, $8 \times 8$, $16 \times 16$, and $32 \times 32$ grid.

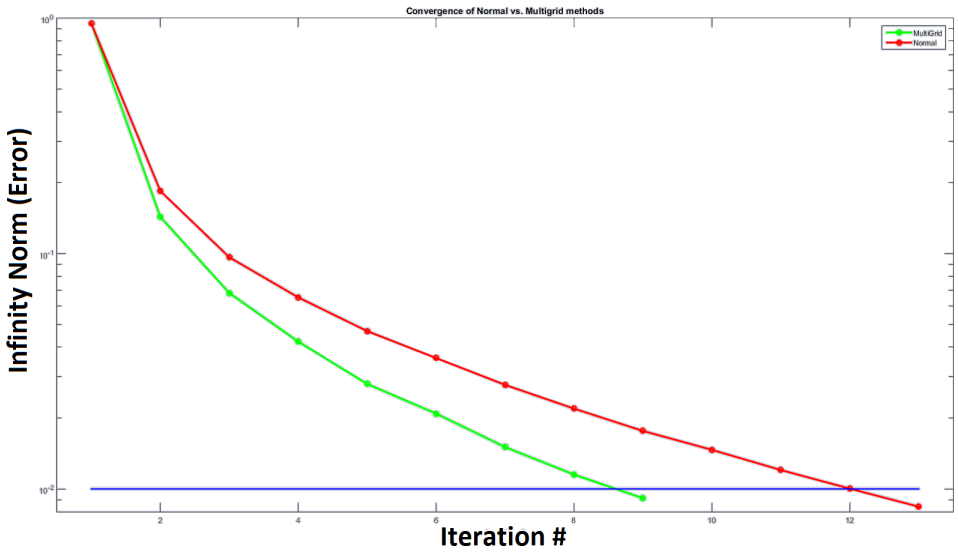



FIGURE 5. Convergence of multigrid vs. original methods. Red line shows error at each step for initial guess of $\overrightarrow{0}$, green line shows error at each step for initial guess generated using multigrid method.

Figure 4 shows the number of steps for convergence, for a problem solved using the iterative method, where the green line represents an initial guess generated by our multigrid method, and the red line represents an initial guess of 0. The boundary forcing functions were $h(x, \omega) = 1$, and $f(x, \omega) = 0$.

The multigrid approach shows a promising reduction in the number of iterations required until convergence. Experimenting with other forcing/boundary functions and parameters, we found that the higher $\frac{\mu_s}{\mu_a}$ becomes, the more iterations are required until convergence, and the greater disparity between the multigrid intial guess and $\overrightarrow{0}$.

## 5. Parallelization

One of the major focuses of this project was parallel implementations of the aforementioned methods, which was one of the reasons Julia was selected as the programming language of choice. All parallel tests were done on a server consisting of 44 hyperthreaded processors.

The iterative method for approximating the RTE lended itself much better to manual parallelization, given that every point could be calculated independently of the other points in the solution, and each processor could just get a roughly equal number of points to compute. The implicit approach was much harder to *manually* parallelize, given that Julia's linear solver is parallel by default, and that the time bottleneck was the solving of the system, not the creation of matrices used in the systems.

The following results come from an approximation using the iterative method ($g(x,\omega) = 1$, $f(x,\omega) = 0$, $nx = ny = 1024$, $nw = 32$). When approximating the solution with $p$ processors, either the $X$, $Y$, or $W$ input vectors were 'split up' into $\frac{nx}{p}$, $\frac{ny}{p}$, or $\frac{nw}{p}$ sized chunks, and each processor was spawned a task consisting of that split up vector, as well as all points for the other 2 vectors (e.g. If $W$ was split up, each task would do $\frac{nw}{p}$ directions, and all corresponding $nx \times ny$ spatial nodes for each direction). No significant difference was noticed when choosing to split by $X$, $Y$, or $W$ vectors.

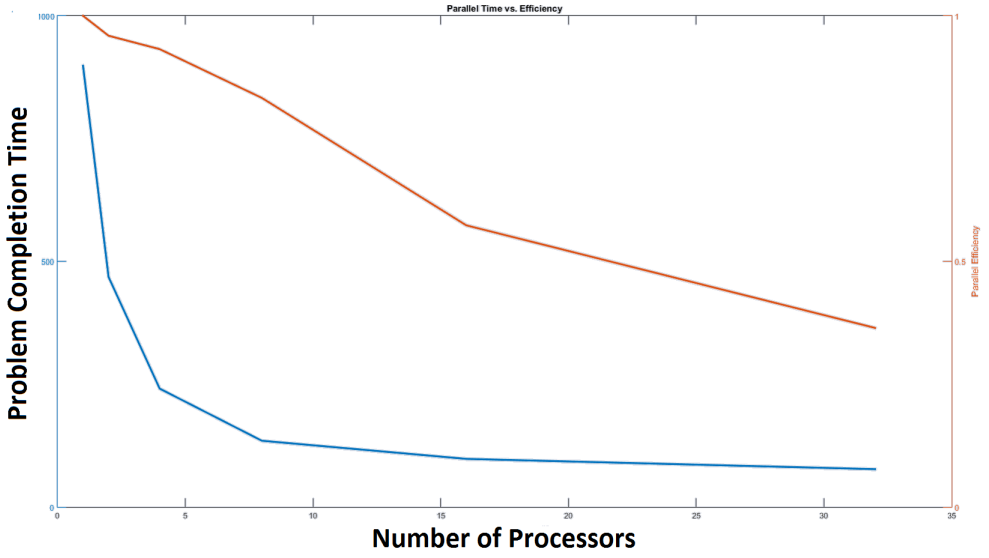



Figure 6. Example of Coarse Mesh Interpolating

Some interesting things to note are that giving each processor more directions to check, as opposed to more spatial resolution, seems to give better parallel efficiency, when looking at a similar number of mesh points. Also, when going from 32 to 64 processors (in row 1), we see a very significant drop in efficiency (though still a slight running time improvement). This is likely due to our parallel server only having 44 physical cores, a threshold which was crossed.

## 6. Conclusion

In conclusion, both the iterative and implicit methods appeared to accurately approximate a solution to the RTE, though each solution had their tradeoffs. While the implicit approach ran faster in serial, the iterative approach scaled much more nicely in parallel. The implicit method also has a much larger spatial complexity ($O(n^6)$) and time complexity ($O(n^9)$ when scaling to much larger step sizes.

The multigrid method was very effective in reducing computation time for the iterative method with scattering , taking around 40% - 60% of the time as starting with an initial guess of $\overrightarrow{0}$. Looking at other ways of scaling up (rather than simply doubling all dimensions) might lead to more efficient solutions.

There are many avenues for expanding our work, such as allowing the boundary of our medium to take any shape, as opposed to a simple circle centered around the origin, or moving from 2 spatial dimensions to 3. It might be rewarding to improve upon the efficiency of Julia's built-in parallel linear solver, to try improving time efficiency for the implicit approach. Finally, looking at different, more complicated parallel schemes in general could yield more efficient processor usage and see a great reduction in problem time, especially were we to move to 3 spatial dimensions.