Radiosity and Relaxation Methods

Progressive Refinement is Southwell Relaxation

Steven Gortler and Michael F. Cohen
Department of Computer Science
Princeton University

Philipp Slusallek
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

February 3, 1993

Abstract

The radiosity method for realistic image synthesis has been described in the computer graphics literature since 1984. This paper discusses the various algorithms which have been developed for solving the radiosity problem and places them in the context of the literature on solving systems of linear equations. The progressive radiosity method developed in 1988 is shown to be equivalent to a numerical technique known as Southwell iteration. A proof of convergence for this method when used for the radiosity problem is presented in the appendix. A new overshooting (similar to over relaxation) method is developed as a means of accelerating the convergence of the iterative radiosity methods.

1 Introduction

In 1984, Goral et al. introduced the radiosity method as a way of obtaining an approximate solution to the global illumination problem of image synthesis [6]. The radiosity solution is obtained by solving a system of linear equations resulting from a discrete approximation of the illumination across surfaces in an environment.

The original paper used a Gaussian elimination scheme to solve the linear system. In [3] Cohen and Greenberg introduced the *hemicube* algorithm for computing interaction coefficients, or *form factors*, in environments with occluded surfaces. They also recognized that the matrix was *diagonally dominant*, and suggested the use of Gauss-Seidel (GS) iteration to obtain the solution of the linear system of equations. In 1988, Cohen et al. introduced the progressive refinement (PR) approach to obtaining a radiosity solution, presenting a different method for solving the same linear system [2]. The PR method has the advantages of quickly converging to an accurate image, and displaying an approximate image while the computation proceeds. A modification to this method uses *overshooting* to more rapidly approach an accurate solution [4].

To date, there has been some confusion in the computer graphics community about where the Progressive Radiosity method sits in relation to the numerical methods literature on solutions of linear systems of equations. In this paper we show that PR is actually equivalent to a numerical analysis technique know as Southwell relaxation. In section 2 we discuss GS, PR, and overshooting methods from the point of view of radiosity. We also develop a new overshooting method which has faster convergence than PR for radiosity problems. In section 3 we discuss GS, Southwell, Jacobi iteration, and over relaxation from the point of view of linear systems in general. In section 4 we show that PR is actually an implementation of Southwell relaxation followed by a Jacobi sweep. In section 5 and in the appendices we rigourously show that Southwell's method converges for radiosity problems. In section 6 we present experimental results comparing the available algorithms on a variety of test cases.

2 Radiosity Solutions with Gauss-Seidel and Progressive Refinement Radiosity

2.1 Gathering and Shooting

The radiosity formulation results in the system of n linear equations (where n is the number of discrete patches) given by,

$$B_i = E_i + \rho_i \sum_j B_j F_{i,j} \tag{1}$$

or in matrix form:

$$\begin{bmatrix} 1-\rho_1 F_{1,1} & -\rho_1 F_{1,2} & -\rho_1 F_{1,3} & . & . & -\rho_1 F_{1,n} \\ -\rho_2 F_{2,1} & 1-\rho_2 F_{2,2} & -\rho_2 F_{2,3} & . & . & -\rho_2 F_{2,n} \\ . & & & & & . \\ . & & & & & . \\ -\rho_{n-1} F_{n-1,1} & & & & & . \\ -\rho_n F_{n,1} & . & . & . & . & 1-\rho_n F_{n,n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ B_{n-1} \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_{n-1} \\ E_n \end{bmatrix}$$

where:

B_i = the radiosity of the i^{th} patch
ρ_i = the reflectivity of the i^{th} patch
E_i = the emission of the i^{th} patch
$F_{i,j}$ = the form factor from patch i to patch j
= the fraction of energy leaving patch i arriving directly at patch j
A_i the area of the i^{th} patch (appears later in the paper)

We will now briefly review and describe the PR and GS methods for solving radiosity problems. Let us begin with the GS method.

```
1  for all i
2     B_i = E_i
3  while not converged
4     for each i in turn
5        B_i = E_i + ρ_i Σ_{j≠i} B_j F_ij
6  display the image using B_i as the intensity of patch i.
```

There is a simple physical interpretation for this algorithm. In line 5 we obtain a new estimate for the radiosity of patch i by adding its emitted radiosity, and all the radiosity that this patch reflects from incoming radiosity. We estimate this incoming radiosity by "gathering" the radiosity from the other patches, using the most recent estimates (B_j) as the radiosities of all the other patches.

Each gathering step (line 5) updates the radiosity of only one patch, gathering for the patches i in order. A gathering step takes $O(n)$ operations and can be viewed as the dot product of the vector $\mathbf{B}$, with the appropriate row of the radiosity matrix. For all patches to have gathered some radiosity, all rows must be processed. In fact, the solution converges after some number of complete passes through the matrix.

Let us contrast this with the PR algorithm.

```
1  for all i
2     B_i = E_i
3     ΔB_i = E_i
4  while not converged
5     pick i, such that ΔB_i * A_i is largest
6     for every patch j
7        Δrad = ΔB_i * ρ_j F_ji
8        ΔB_j = ΔB_j + Δrad
9        B_j = B_j + Δrad
10    ΔB_i = 0
11    display the image using B_i as the intensity of patch i.
```

The above algorithm has the following physical interpretation. All patches i have a value B_i which is the radiosity calculated so far for that patch, and ΔB_i which is the portion of that patch's radiosity which has yet to be "shot". During one iteration, the patch with the most unshot radiosity is chosen and its radiosity is shot through the environment. As a result of the shooting, the other patches j, may receive some new radiosity, Δrad. This Δrad is added to B_j. This Δrad is also added to ΔB_j since this newly received radiosity is unshot. As a result of the shooting, patch i has no unshot radiosity so $\Delta B_i = 0$.

In this algorithm one shooting step (lines 6–10) updates all the other patches. We shoot from the patch that currently has the most unshot radiosity. One shooting step takes O(n) operations, and can be viewed as as multiplying the scalar B_i, by a column of the form factor matrix. Cohen et al.[2] showed that in many cases only a small fraction of n shooting steps is required to closely approximate a solution.

At first glance these two algorithms seem to be very distinct. One gathers, the other shoots. One updates a single patch, the other updates all of them. One uses rows of the matrix, the other uses columns. In this paper we will show that these two methods are quite related.

2.2 Overshooting

In the PR algorithm, as each patch shoots its unshot radiosity, all other patches may receive some portion of that radiosity, of which some is reflected back into the environment and some absorbed. Some of that reflected radiosity will return to the shooting patch. In addition, some energy will arrive at the shooter from other unshot radiosity sources in subsequent steps. This may result in the need to shoot radiosity from the same patch multiple times. An alternative would be to shoot the current unshot radiosity plus an estimate of future reflected radiosity. This modification to the PR algorithm has been discussed by Feda [4].

2.2.1 Overshooting Using The Ambient Term

In [2], an *ambient* term estimated from the total unshot radiosity in the environment was added *for display purposes only*; this term was not used as part of the itererative solution method. Feda used this ambient term to do *overshooting*. If we call the overshooting amount $\Delta\hat{B}_i$, then the PR algorithm becomes,

1 for all i
2 $\quad B_i = E_i$
3 $\quad \Delta B_i = E_i$
4 while not converged
5 $\quad$ pick i, such that $(\Delta B_i + \Delta\hat{B}_i) * A_i$ is largest
6 $\quad$ for each patch j
7 $\quad\quad \Delta rad = (\Delta B_i + \Delta\hat{B}_i) * \rho_j F_{ji}$
8 $\quad\quad \Delta B_j = \Delta B_j + \Delta rad$
9 $\quad\quad B_j = B_j + \Delta rad$
10 $\quad \Delta B_i = -\Delta\hat{B}_i$
11 $\quad$ display the image using B_i as the intensity of patch i.

Note that after shooting, the unshot radiosity of patch i is the negative of the overshooting amount. The hope is that as other patches shoot their radiosity, this value will tend back towards zero. It may, however, be necessary to shoot a *negative* amount of radiosity back into the environment if

the radiosity to overshoot was overestimated.

One would like an estimate of the radiosity which will arrive at a patch in the future, to determine the best value for overshooting. Feda used the ambient term described in [2] defined as the area weighted average unshot radiosity, $\Delta\bar{B}$, increased by the geometric series of the average reflectivity, $\bar{\rho}$, (to account for multiple reflections),

$$Ambient \;=\; \Delta\bar{B} \;*\; (1 + \bar{\rho} + \bar{\rho}^2 + \bar{\rho}^3 + ...) \;=\; \Delta\bar{B} \;*\; \frac{1}{1-\bar{\rho}} \tag{2}$$

This estimate may become too high, particularly if $\bar{\rho}$ is close to unity. The estimate also ignores form factor information available just before shooting.

2.2.2 Overshooting Using Known Information

In this section we present an alternative method which can account in advance for some of the radiosity that will return due to interaction with the environment but only uses known information, and does not rely on any estimations. Since computing the form factors is the most expensive part of a shooting step, it makes sense to exploit these calculations as much as we can.

When a patch i is chosen for shooting and its form factors are computed, we can obtain both a row and a column of the form factor matrix using the reciprocity relationship.

$$F_{ij}A_i = F_{ji}A_j. \tag{3}$$

With this information we can shoot all of i's unshot radiosity into the environment, compute how much radiosity is shot from all other patches, j, to the patch i (gathering), how much of *that* radiosity is shot back into the environment, how much of *that* radiosity is returned directly to the chosen patch, and so on ad infinitem. In other words we can shoot, then gather, then shoot, etc. Let us call this step involving an infinite series, a *SuperShootGather*, or simply (SG). See figure 1.

To compute the (SG) we must solve the following radiosity subproblem:

$$\begin{bmatrix} 1 & 0 & -\rho_1 F_{1,i} & & & \\ 0 & 1 & -\rho_2 F_{2,i} & 0 & & \\ & & \cdot & & & \\ -\rho_i F_{i,1} & -\rho_i F_{i,2} & \cdot\;\; 1 \;\;\cdot & -\rho_i F_{i,n-1} & -\rho_i F_{i,n} & \\ & & \cdot & & & \\ & 0 & -\rho_{n-1} F_{n-1,i} & 1 & & \\ & & -\rho_n F_{n,i} & & & 1 \end{bmatrix} \begin{bmatrix} (SG)_1 \\ (SG)_2 \\ \cdot \\ (SG)_i \\ \cdot \\ (SG)_{n-1} \\ (SG)_n \end{bmatrix} = \begin{bmatrix} \Delta B_1 \\ \Delta B_2 \\ \cdot \\ \Delta B_i \\ \cdot \\ \Delta B_{n-1} \\ \Delta B_n \end{bmatrix}$$

In this radiosity subproblem our selected patch i can interact with all the other patches and vice versa, but the other patches cannot interact with each other. The unshot radiosity replaces the "emissions". This system has the following closed form solution (where i is the chosen patch);

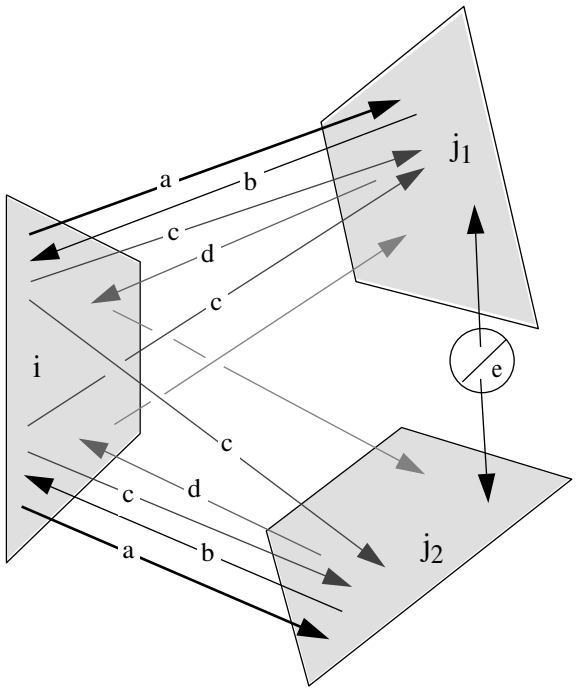

Figure 1: Super_Shoot_Gather: a) shoot unshot energy from i to all j. b) gather unshot energy to i from all j. c) shoot this energy. d) re-gather, etc. e) NO energy transfer between patches j.

$$(SG)_i = \frac{\Delta B_i + \sum_{j \neq i} \rho_i F_{ij} * \Delta B_j}{1 - \sum_{j \neq i} \rho_i F_{ij} \rho_j F_{ji}} \tag{4}$$

$$(SG)_{j \neq i} = \Delta B_j + \rho_j F_{ji} * (SG)_i \tag{5}$$

$(SG)_i$ is the radiosity of a patch that results from the unshot radiosity bouncing around our restricted environment. As a result of this interaction, patch j's radiosity will increased by $(SG)_j - \Delta B_j$. ΔB_i is now 0 since we have just shot from patch i into the environment, and no unshot radiosity remains. In addition, the other patches have no more radiosity to shoot to patch i.

However, we cannot set the ΔB_j to zero, since they now have $(SG)_j - \Delta B_j$ more unshot radiosity to shoot towards each other. Thus, there is no single value of "unshot radiosity" for each patch. We must keep track of an unshot radiosity *matrix* where ΔB_{jk} is the amount of radiosity unshot from patch j to patch k. Row j of this matrix indicates how much unshot radiosity patch j has to shoot to every other patch, and column j represents how much unshot radiosity each other patch has to shoot towards patch j.

Fortunately, we do not have to maintain a full matrix of unshot radiosity, requiring quadratic storage, and quadratic time to update during each step. Instead, we explicitly store ∇B_{jk}, the amount of radiosity already shot from j to k. We can compute the unshot radiosity from patch j to patch k as $\Delta B_{jk} = B_j - \nabla B_{jk}$. By storing the *shot* radiosity we only have to update a linear number of matrix entries after each step.

Also, the actual stored matrix will only have non-zero entries in the rows and columns of patches which have previously been selected. Thus, after shooting from a small number of patches (as is usually the case in PR) the matrix will not require exorbitant storage.

When computing the (SG) radiosity subproblem, we replace the "emissions" of patch j with ΔB_{ji}, (patch i is the only patch that j can interact with). Patch i however interacts with all the patches and has a different amount of radiosity unshot to each of them. We thus first do a shooting operation from patch i (shooting different amounts to each patch), and then compute (SG) (with patch i having no unshot radiosity.)

Here is the complete algorithm (recall that $\Delta B_{jk} = B_j - \nabla B_{jk}$):

```
1  for all j                                  10  /*compute (SG) */
2     B_j = E_j                               11  (SG)_i = (Σ_{j≠i} ρ_i F_ij * ΔB_ji) / (1 - Σ_{j≠i} ρ_i F_ij ρ_j F_ji)
3  while not converged                        12  B_i = B_i + (SG)_i
4     pick a patch i                          13  for every other patch j
5     /* do a shoot */                        14     (SG)_j = ΔB_ji + ρ_j F_ji * (SG)_i
6     for every other patch j                 15     Δrad_j = (SG)_j - ΔB_ji
7        Δrad_j = ΔB_ij * ρ_j F_ji            16     B_j = B_j + Δrad_j
8        B_j = B_j + Δrad_j                   17     ∇B_ji = B_j
9        ∇B_ij = B_i                          18     ∇B_ij = B_i
                                              19  display the image using B_i as the intensity of patch i.
```

In line 4, we should choose the patch whose row and column of unshot radiosity sum to the greatest number (possibly weighted by area).

This implementation of the overshooting algorithm requires linear time for each step.

3 Relaxation

3.1 Gauss-Seidel and Southwell

In this section we will briefly review the concept of Relaxation as it applies to solving linear systems. We will discuss two related methods, GS iteration, and Southwell relaxation. For a more complete discussion see [8][5].

We wish to solve the linear system

$$\mathbf{M}\,\mathbf{x} = \mathbf{b} \tag{6}$$

where $\mathbf{M}$ is an n by n matrix. Given the approximate solution at the k^{th} step of the algorithm, $\mathbf{x^{(k)}}$ we define the k^{th} error as

$$\mathbf{e^{(k)}} = \mathbf{x} - \mathbf{x^{(k)}}, \tag{7}$$

and we define the k^{th} residual as

$$\mathbf{r^{(k)}} = \mathbf{b} - \mathbf{M}\,\mathbf{x^{(k)}}. \tag{8}$$

Notice

$$\mathbf{r^{(k)}} = \mathbf{M}\,\mathbf{e^{(k)}}. \tag{9}$$

We would like some method of moving from an approximate solution $\mathbf{x^{(k)}}$ to an approximation $\mathbf{x^{(k+1)}}$ which is closer to the correct solution $\mathbf{x}$. If, when using this method, the residuals $\mathbf{r^{(k)}}$ converge to zero, then we have converged to the correct solution. One method that attempts to find a solution this way is relaxation. Given the approximation $\mathbf{x}$, we pick one of the $x_i^{(k)}$ to change in such a way that $r_i^{(k+1)} = 0$. Of course the other $r_j^{(k)}$ may increase, but we hope that we have made an improvement on the whole.

A little algebra shows that if we wish to relax x_i, we should set

$$x_i^{(k+1)} = (b_i - \sum_{j \neq i} M_{ij} * x_j^{(k)})/M_{ii} \tag{10}$$

Alternatively, since

$$r_i^{(k)} = b_i - \sum_{j} M_{ij} * x_j^{(k)} \tag{11}$$

we can set

$$x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)}/M_{ii}. \tag{12}$$

This step takes $O(n)$ operations. It involves taking the dot product of $\mathbf{x}$ with a row of the matrix. If we relax the i's in order, we obtain the following algorithm.

```
1  for all i
2     x_i = 0
3  while not converged
4     for each i in turn
5        x_i = (b_i - sum_{j≠i} x_j M_ij)/M_ii
6  output x
```

This is the GS iteration algorithm. It is easy to see that this is the same as the gathering algorithm presented above. The x_i here corresponds to the radiosities, the b_i here corresponds to the emittances, and the matrix $\mathbf{M}$ corresponds to the radiosity matrix defined above.

Suppose that instead of sweeping the i's in order, we decide to relax the i with the greatest residual r_i. This ordering is called Southwell iteration [5]. At first you might think that we would have to spend $O(n^2)$ operations to compute all the r_i's before picking the greatest one. (The computation of each r_i above involves computing the dot product of $\mathbf{x}$ with the row M_i).

Fortunately, there is a better way. If we know, at some step k, $\mathbf{r^{(k)}}$ for a given $\mathbf{x^{(k)}}$, we can express our next approximation as:

$$\mathbf{x^{(k+1)}} = \mathbf{x^{(k)}} + \mathbf{\Delta x^{(k)}} \tag{13}$$

and we can compute the updated residual as:

$$\mathbf{r^{(k+1)}} = \mathbf{b} - \mathbf{M}(\mathbf{x^{(k)}} + \mathbf{\Delta x^{(k)}}) = \mathbf{r^{(k)}} - \mathbf{M}\ \mathbf{\Delta x^{(k)}} \tag{14}$$

since

$$\mathbf{r^{(k)}} = \mathbf{b} - \mathbf{M x^{(k)}}. \tag{15}$$

In our case $\mathbf{\Delta x^{(k)}}$ is a vector with zeros everywhere except for the i^{th} component which is $r_i^{(k)}/M_{ii}$. Thus,

$$r_j^{(k+1)} = r_j^{(k)} - \frac{M_{ji}}{M_{ii}} * r_i^{(k)}. \tag{16}$$

Updating $\mathbf{r}$ takes only $O(n)$ steps. This step involves multiplying a scalar by a column of the matrix.

The only thing we must still show is that we are able to compute $\mathbf{r^{(0)}}$ easily at the start of the algorithm. This is simple. If we choose $\mathbf{x^{(0)}}$ to be $\mathbf{0}$ (the zero vector), then

$$\mathbf{r^{(0)}} = \mathbf{b} - \mathbf{M x^{(0)}} = \mathbf{b} - \mathbf{M} * \mathbf{0} = \mathbf{b}. \tag{17}$$

We can write the Southwell relaxation method as follow.

```
1  for all i
2     x_i = 0
3     r_i = b_i
4  while not converged
5     pick i, such that r_i is largest
6     x_i = x_i + r_i/M_ii
7     temp = r_i
8     for all j
9        r_j = r_j - M_ji/M_ii * temp
10 output x
```

To summarize, we have presented two methods which solve linear systems by relaxing one variable at a time. The only difference between the two methods is how the variable to be relaxed is chosen. One chooses variables in order. The other chooses the variable with the largest corresponding residual. Both methods require $O(n)$ operations per relaxation step. One performs a row operation, the other performs a column operation.

3.2 Jacobi Iteration

There is another iteration algorithm known as Jacobi iteration. It differs from GS in the following way. In Jacobi iteration we keep two copies of all the variables x_i, one old and the other new. When we update a variable using $x_i^{(k+1)} = (b_i - \sum_{j \neq i} M_{ij} * x_j^{(k)})/M_{ii}$ we update only the new copy of x_i, and we continue using the old copy in all further computation. Only after we have updated

all n variables, do we begin using the new copies. We then use these fixed values for the next *sweep* through all the variables.

We can also express Jacobi iteration as updating variables using $x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)}/M_{ii}$. But unlike GS where we compute new values for all the r_j after we update one variable, in Jacobi iteration we add *all* the residuals to their variables before we compute any new r_j.

3.3 Over Relaxation

Over relaxation techniques are similar to the relaxation methods described above with one exception. When relaxing a particular residual, the change in the solution vector is increased by a factor ω (or equivalently the residual is considered to have been larger than it actually is). Equation 13 becomes,

$$x^{(k+1)} = x^{(k)} + \omega \Delta x^{(k)} \tag{18}$$

and the i^{th} residual being relaxed is now not set to zero, but rather,

$$r_i^{(k+1)} = (1 - \omega) * r_i^{(k)} \tag{19}$$

Over relaxation involves a value for ω greater than 1, while under relaxation involves a value between 0 and 1. In cases where GS methods converge, over relaxation will often increase the convergence rate. This has been the experience with radiosity algorithms [1]. The overshooting methods of section 2.2 fall into this class of algorithm.

4 Transforming Progressive Radiosity to Southwell

The PR algorithm is similar to Southwell in that both operate with one column of the matrix during one step. The algorithms are different in that PR appears to update all of the variables in one step, whereas Southwell updates only one of the variables per step.

However, we can make the following transformation. Define a new variable ∇B_i where $\nabla B_i = B_i - \Delta B_i$. ∇B_i is the amount of "shot" radiosity while ΔB_i is the amount of "unshot" radiosity. Naturally $\nabla B_i + \Delta B_i = B_i$. Here is the rewritten algorithm:

```
1  for all i
2     ∇B_i = 0
3     ΔB_i = E_i
4  while not converged
5     pick i, such that ΔB_i * A_i is largest
6     ∇B_i = ∇B_i + ΔB_i
7     for every other patch j
8        ΔB_j = ΔB_j + ρ_j F_ji * ΔB_i
9     ΔB_i = 0
10    display the image using ∇B_i + ΔB_i as the intensity of patch i.
```

This algorithm is equivalent to the first algorithm, and will display the same sequence of images. But in this form it is clear that we are actually implementing Southwell relaxation. ΔB is simply the residual $\mathbf{r}$, and ∇B is the vector of variables (unknowns) that we are solving for $\mathbf{x}$. The matrix $\mathbf{M}$ is equivalent to the original matrix given at the beginning of the paper.

The only difference between Southwell and our PR algorithm is that at the end of PR, instead of outputting ∇B ,which is the variable vector we are solving for, we output $\nabla B + \Delta B$, which is the variables added to their residuals. This makes sense within our physical interpretation. When the algorithm is finished, we have the "unshot" radiosities stored in ΔB, so adding them to our image should give us a more correct image. This also makes sense from a numeric point of view. By outputting the residuals added to the variables, we are performing one complete sweep of Jacobi iteration. In other words, performing m shooting operations is the same as performing m Southwell relaxation steps followed by one complete Jacobi sweep. The numerical significance of these steps will be discussed in the next section. (There is one other minor difference. In Southwell the variable with the largest residual is chosen. In PR the variable with the largest area weighted residual is chosen. This will also be adressed in the next section).

5 Discussion of Convergence

5.1 Southwell Converges for Radiosity Problems

In this section, we will briefly discuss the convergence properties of Southwell and use those properties to show that PR converges for radiosity problems. We will also discuss the significance of the final Jacobi sweep. A full discussion of the necessary and sufficient conditions for the convergence of Southwell's relaxation is also beyond the scope of this paper. Much of the literature discussing Southwell restricts itself to symmetric positive definite matrices [5].

In appendix A, we show that Southwell relaxation converges to the correct solution of $\mathbf{Mx} = \mathbf{b}$, and that at each step, the error decreases, for certain row diagonally dominant matrices which include radiosity matrices. (A matrix is strictly column diagonally dominant (CDD) if for all

j, $\mid M_{jj} \mid > \sum_{i \neq j} \mid M_{ij} \mid$, and a matrix is strictly row diagonally dominant (RDD) if for all i, $\mid M_{ii} \mid > \sum_{j \neq i} \mid M_{ij} \mid$.) As noted in [3] the matrix in the radiosity is strictly row diagonally dominant. This is true since the reflectivity terms, ρ_i are always less than unity, and the sum of the form factors in any row is by definition equal to unity in a closed environment, (or less than unity in an open environment).

Since Southwell's method converges for our system, ∇B (the variable) converges to the correct solution and ΔB (the residual) converges to 0, and therefore $\nabla B + \Delta B$ (the output of the PR algorithm) converges to the correct solution. (The proof in the appendix includes PR's area weighted patch choice).

In the appendix B, we show that Southwell relaxation converges to the correct solution of $\mathbf{M}\ \mathbf{x} = \mathbf{b}$, and that at each step, the amount of residual decreases, if the matrix is column diagonally dominant. Radiosity matrices are not necessarily CDD. If the radiosity matrix has a column that sums to more than 1, and the corresponding variable is relaxed, the total amount of residual will actually increase. This means that after a shooting operation, it is possible for the total amount of unshot *radiosity* to increase.

However non-intuitive this may seem, it does not contradict the fact that our system is physically dissipative, since after each shooting step, the amount of unshot *energy* does decrease. This can be shown as follows. Instead of solving the system:

$$B_i = E_i + \rho_i \sum_j B_j F_{ij} \tag{20}$$

solve the equivalent system:

$$B_i A_i = E_i A_i + \rho_i \sum_j B_j A_j F_{ji} \tag{21}$$

These systems are equivalent due to the reciprocity relationship. Ignoring time, the first system expresses the equation with variables B_i having units of radiosity (energy/area). The second system expresses the exact same equation with variables $B_i A_i$ having units of energy. The matrix from the first system is RDD, while the matrix from the second system is CDD. So if we apply Southwell's method to the second system, by the arguments in the appendix it will converge. And after each step, the amount of residual, which in the second system is the total amount of unshot energy, will decrease. It is easy to see that relaxing a variable in the second system is computationally equivalent to relaxing a variable in the first system. The only difference is that Southwell's method applied on the second system will pick the patch with the most unshot energy. This explains the patch choice made in the PR algorithm above, where the patch with the greatest amount of unshot *energy* was selected.

What is the significance of the Jacobi sweep?

Jacobi iteration is guaranteed to converge to the correct solution if the matrix is CDD or RDD [8]. Extending arguments similar to those of the appendix, we can show that in radiosity problems, one Jacobi sweep brings us closer to a correct solution, reducing the error for each variable, and and

reducing the total amount of residual.

One full Jacobi sweep is n "under-relaxation" steps. (Jacobi steps are not full relaxation steps as defined above, since by using old values in its computation, the Jacobi "relaxation" will not set the current residual of a variable to 0.) In PR we obtain this Jacobi sweep at no extra cost, although generally a full sweep should cost us $O(n^2)$. If we were already doing some large number of relaxation steps, then n free steps may not be very significant, But in PR, we typically do some relatively small number of relaxation steps so this free Jacobi sweep is a relatively significant advantage. This free sweep is what allows us to update all the variables and arrive at a radiosity solution without even going through n full relaxation steps.

In appendix C we outline the correspondence between the overshooting method of section 2.2.2 and the application of block relaxation techniques to an extended linear system.

5.2 Comparison of Shooting and Gathering

Gathering is an implementation of GS, while shooting is an implementation of Southwell followed by one Jacobi sweep. Why is shooting better?

GS and Southwell are both sequences of relaxations. The algorithms differ only in the method they use to choose variables. Southwell has the advantage of using a greedy heuristic when choosing variables. It chooses the largest residual, in hopes of reducing the total residual by a large amount. Also since $M_{ii} = 1$ and variables are relaxed by $x_i = x_i + r_i/M_{ii}$, Southwell's choice of largest residual is also the choice that changes any variable by the largest amount possible. And since in radiosity problems the x_i are always increasing, that is we never overshoot the correct answer, moving it by the most possible, is also removing the most error possible in any step. It is important to note that this does not imply that Southwell must always do better than GS in the long run. One can show cases where making a "worse" choice now would allow us to remove more error in later steps.

Southwell's method is not well known or extensively used in numerical analysis practice. This is perhaps due in part to the extra overhead needed in Southwell to pick the maximum, and the fact that as the problem moves towards exact convergence Southwell may not do any better than GS.

In radiosity problems, the advantages of Southwell are more clear since the initial residual is very concentrated at the light sources, i.e., the emission vector, E, usually has only a few non-zero terms. As the problem continues, much of the radiosity is supplied by a few bright reflecting surfaces. At these early stages many of the patches have no or little residual, so GS spends a lot of time updating variables that don't change by too much. This advantage is accentuated by the fact that the cost of the form factor computation is generally much more significant than the matrix solution, thus Southwell provides an approximate solution without having to precompute the full matrix. Southwell concentrates its effort on variables where a lot of change will occur. Once the radiosity becomes more distributed through the environment (and the advantages of picking the max each time is not as significant) we stop the method and add on an approximation of the ambient radiosity.

The final free Jacobi sweep is the other advantage of shooting over gathering. We are in possession

of all the current residuals and so we can add them on to obtain a Jacobi sweep. The Jacobi sweep is n "relaxation" steps, which is quite significant since our hope is to avoid doing many GS or Southwell steps.

6 Experimental Results

The methods described above were run on a number of test cases and their performance was compared.

After each iteration, the output of each method was compared to a converged solution. We report the error as

$$Error^{(k)} = \frac{\sum_i (B_i^* - B_i^{(k)})}{\sum_i (B_i^* - E_i)} \tag{22}$$

(where B_i^* is the correct radiosity of patch i). There is no need to use the root mean square formula, since all errors are always positive. In the denominator we subtract all the emitted radiosity. With this definition of error, we measure the percentage of "reflected" radiosity that each method accounts for. If a method has not yet accounted for all of the emitted radiosity, then its error is greater than 1.

We compared the following 6 algorithms:

Gauss-Seidel-0 The initial guess for all variables is 0. The variables are then relaxed in order.

Gauss-Seidel+Jacobi The initial guess is 0. The variables are relaxed in order. The output is defined as the variables added to their residuals. (This is equivalent to shooting in order).

Southwell This is just like Gauss-Seidel-0, except that the variable with the largest residual at any time is relaxed.

Southwell+Jacobi This is like Southwell except the output is defined as the variables added to their residuals. (This is equivalent to Progressive Radiosity).

Gauss-Seidel-E The initial guess for all variables is set to be the emissivity of that patch. The variables are then relaxed in order. (This is equivalent to gathering).

Overshooting This is an implementation of the overshooting method described in section 2.2.2.

The algorithms were run on the following cases:

- A matrix of form factors was derived from the geometric description of an office environment. (See figure 2 for a rendered image of the environment). (This system had 270 variables).
- A matrix of form factors was derived from the geometric description of the interior of an empty cube with a few emitting polygons on the "ceiling" of the cube. (This system had 390 variables). The ρ_i were first set to be around 0.3, giving us a dim cube. The ρ_i were then set to be around 0.8, giving us a bright cube.

- A random sparse 390 by 390 row diagonally dominant matrix was generated. We first (randomly) chose a small number of variables to emit random amounts of radiosity. (This is a realistic assumption for many radiosity problems, since only a small number of the polygons usually emit radiosity). We later set all of the variables to emit random amounts of radiosity..

See figures 3-8 for the results of the experiments.

The different methods behaved similarly across many of the test cases. The worst behaved solution was Gauss-Seidel-0. It often did not account for all the emitted radiosity until nearly a full sweep through the matrix.

Gauss-Seidel+Jacobi, Gauss-Seidel-E, and Southwell behaved similarly to each other.

Southwell+Jacobi (progressive radiosity) faired much better than the above algorithms. As expected, this method outperformed the above methods by the greatest amounts in the early iterations. Its advantages were less pronounced when the polygons were all emitters.

The new overshooting algorithm showed the best performance in all cases. This was particularly the case in environments with bright surfaces.

We measured performance in error/iteration. This is the correct metric if the form factor computations dominate the cost of an iteration. But since overshooting does more (although still a linear amount of) arithmetic operations per iteration than the other methods, it may or may not fair better than the other methods measured in error/cpu-time if the form factors are already computed.

7 Conclusion

This paper has attempted to put the various algorithms which have been developed for solving the radiosity problem into the context of the literature on solving systems of linear equations. We have shown the equivalence of the PR method with a numerical technique known as Southwell iteration and have presented a proof of convergence for this method. Overshooting (over relaxation) methods have also been discussed as a means of accelerating the convergence of this iterative method.

Further study should be devoted to placing the hierarchical methods which have recently been developed into a traditional context as well [7]. A more complete study of over relaxation factors is also worth investigation. The authors hope this paper can answer some of the questions which have surrounded the development of radiosity algorithms.

8 Acknowledgements

Pat Hanrahan, Peter Schroeder and David Ohsie gave us many useful comments on the paper. The office model was created by Larry Aupperle. The form factors and rendered image were computed with Larry's program and the help of S. V. Krishnan.

Figure 2: Office environment from Hanrahan et al. Computer Graphics 1991

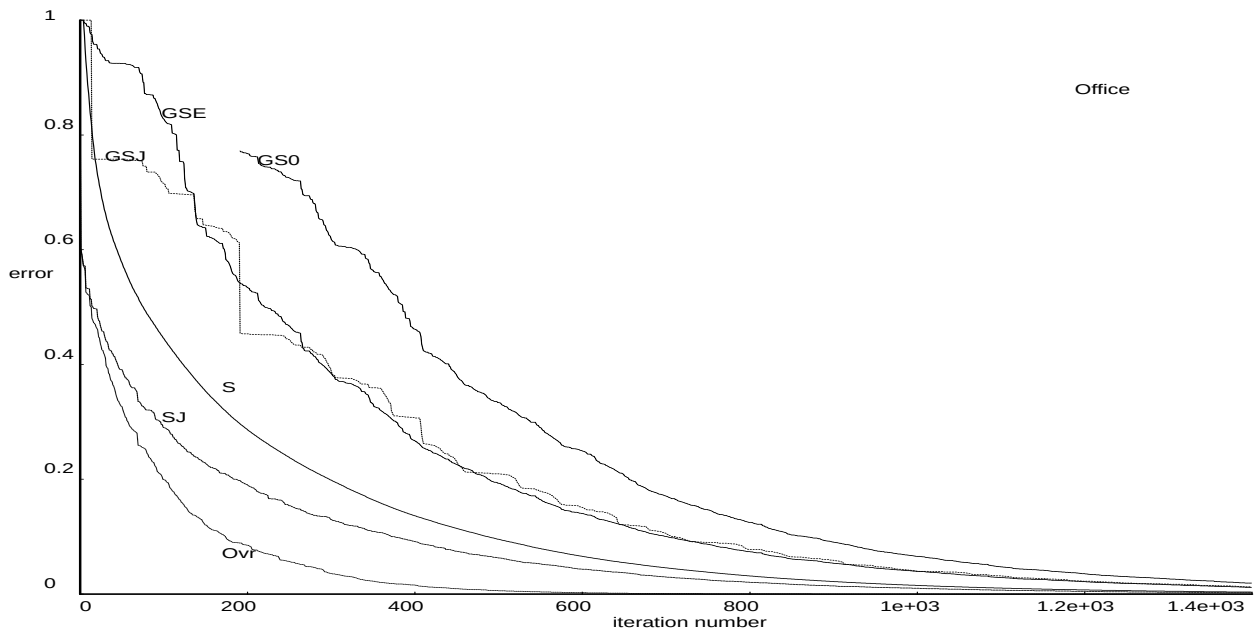

Figure 3: Error at each iteration, office environment

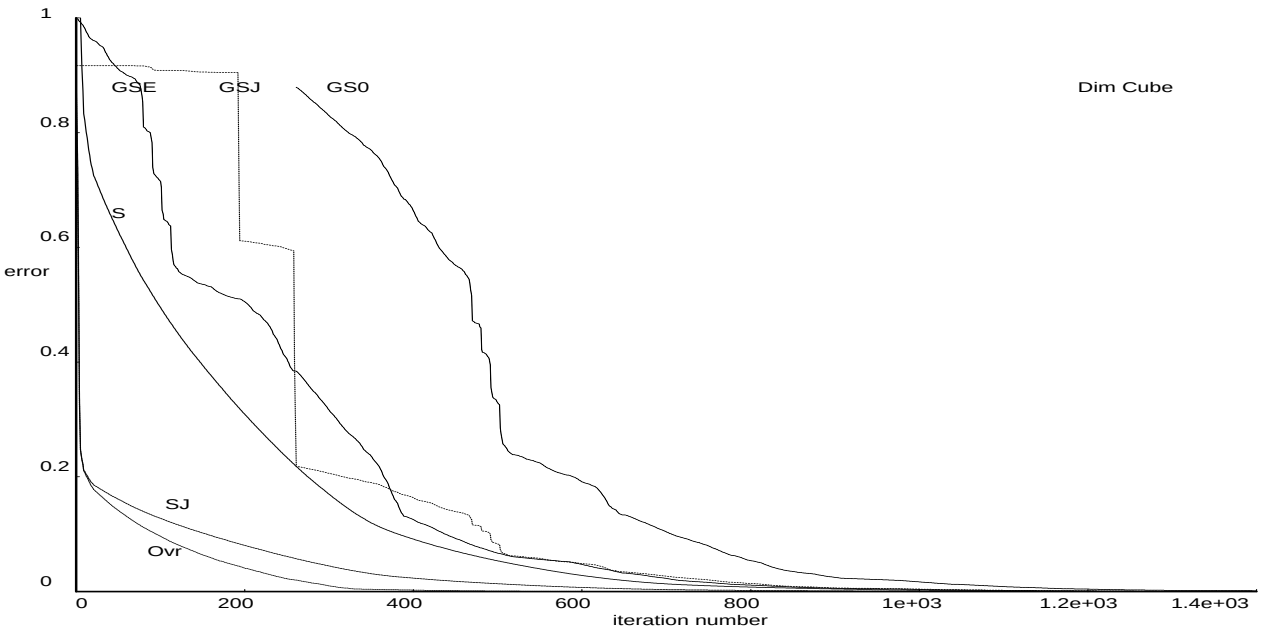

Figure 4: Error at each iteration, dim cube environment

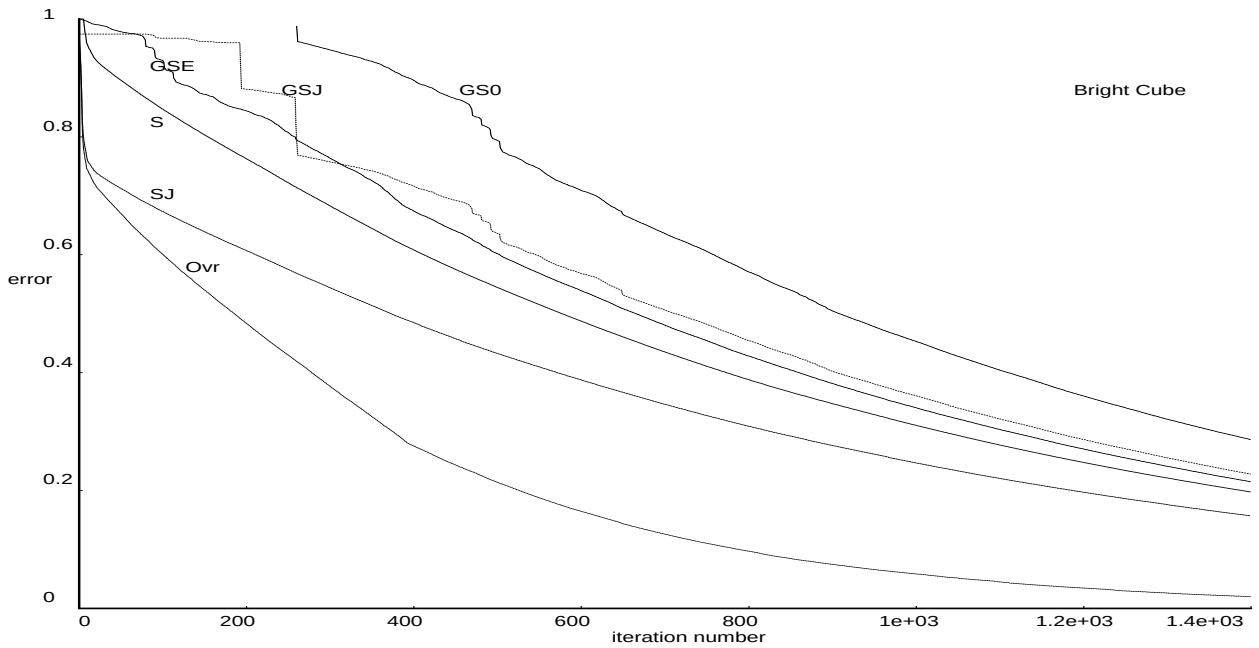

Figure 5: Error at each iteration, bright cube environment

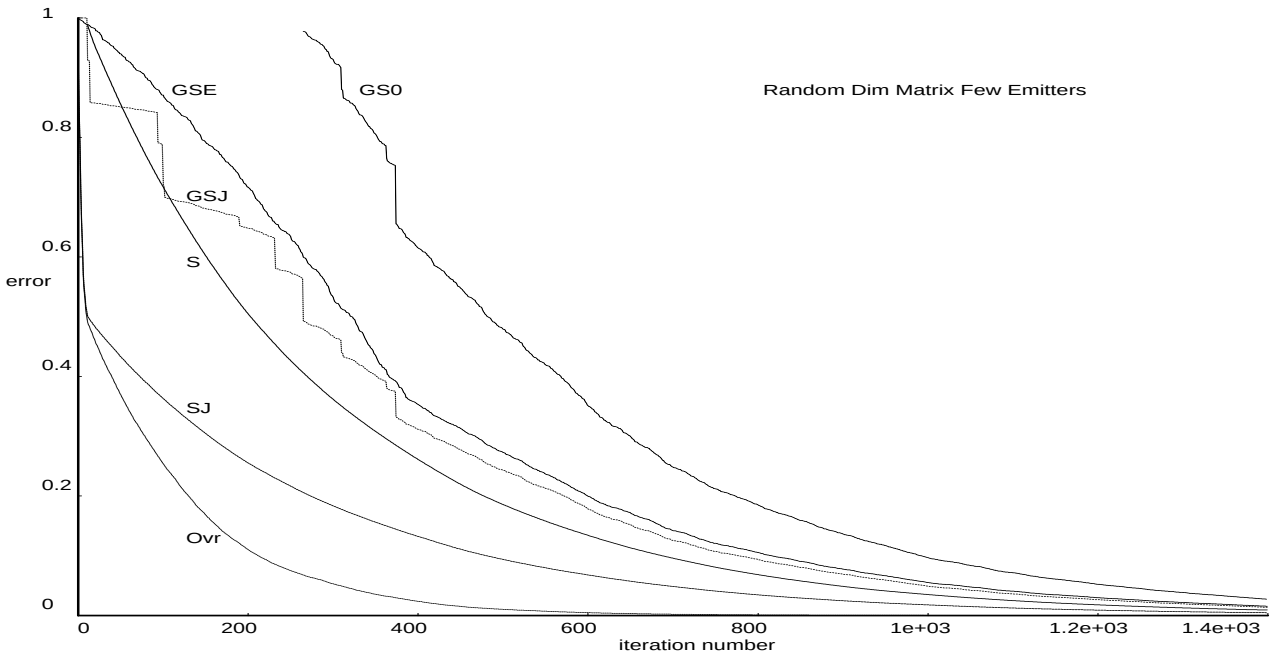

Figure 6: Error at each iteration, random dim matrix with few emmiters

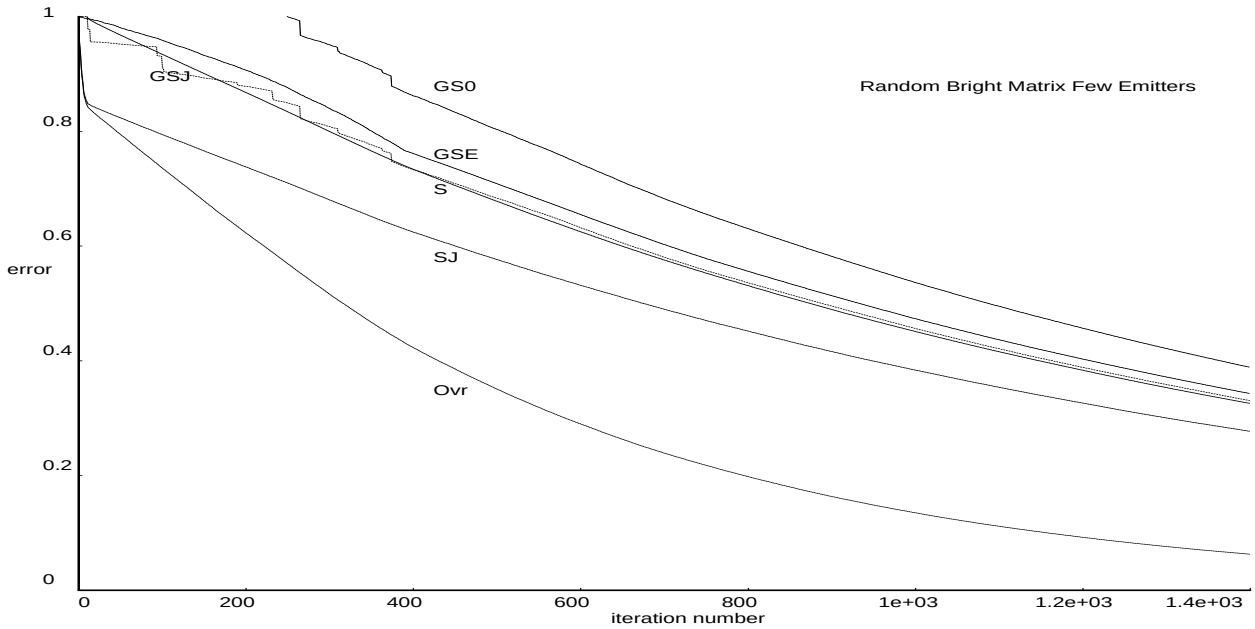

Figure 7: Error at each iteration, random bright matrix with few emmiters

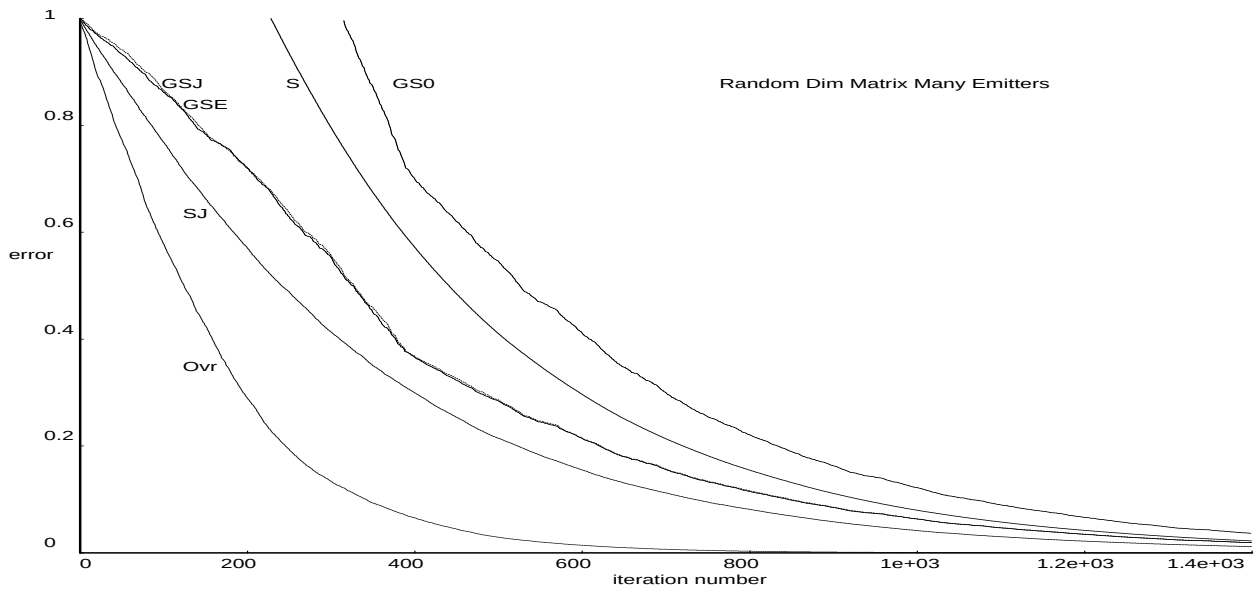

Figure 8: Error at each iteration, random dim matrix with many emmiters

References

[1] Cohen, M. *Masters Thesis: The Radiosity Method for the Realistic Image Synthesis of Complex Diffuse Environments.* Cornell University.

[2] Cohen, M., Chen, S. E., Wallace, J., and Greenberg, D. A Progressive Refinement Approach to Fast Radiosity Image Generation. *Computer Graphics 22*, 4 (July 1988), 75–84.

[3] Cohen, M., and Greenberg, D. The Hemi-cube: A Radiosity Solution for Complex Environments. *Computer Graphics 19*, 3 (July 1985), 31–40.

[4] Feda, M. Accelerating Radiosity by Overshooting. *1992 Eurographics Rendering Workshop* (June 1992).

[5] Gastinel, N. *Linear Numerical Analysis.* Academic Press, 1970.

[6] Goral, C., Torrance, K., Greenberg, D., and Battaile, B. Illumination for Computer-Generated Pictures. *Computer Graphics 18*, 3 (July 1984), 31–40.

[7] Hanrahan, P., Salzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics 22*, 4 (July 1991), 197–206.

[8] Stoer, J., and Bulirsch, R. *Introduction to Numerical Analysis.* Springer Verlag, 1972.

A Proof of Convergence for Certain Row Diagonally Dominant Matrices

In this section we prove that Southwell converges when solving the system

$$\mathbf{M} * \mathbf{x} = \mathbf{b}$$

if the following assumptions are true.

- The matrix $\mathbf{M}$ is strictly row diagonally dominant (RDD).
- The diagonal elements of the matrix are positive.
  The off-diagonal elements are non-positive.
- The vector $\mathbf{b}$ has all non-negative elements.

All of these assumptions are valid in radiosity problems.

Since $x^{(0)} = 0$ and $b^{(0)} \geq 0$.

$$r^{(0)} = b - Mx^{(0)} \geq 0; \tag{23}$$

the initial residuals are all non-negative. After a relaxation step in Southwell, we update all the residuals by

$$r_j^{(k+1)} = r_j^{(k)} - M_{ji}/M_{ii} * r_i^{(k)} \tag{24}$$

For j = i , $r_j^{(k+1)} = 0$, and so it is non-negative. For $j \neq i$, $-M_{ji}/M_{ii}$ is positive and $r_i^{(k)}$ is non-negative, so $r_j^{(k+1)}$ is also non-negative. So for all k and for all i

$$r_i^{(k)} \geq 0. \tag{25}$$

During a relaxation step, we update one x_i by

$$x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)}/M_{ii}, \tag{26}$$

and since $r_i^{(k)}$ is non-negative, and M_{ii} is positive, the $x_i^{(k+1)}$ never get smaller as we proceed from step k to step $k+1$. Now let us look at the error at each step,

$$e^{(k)} \equiv x - x^{(k)}. \tag{27}$$

If $\mathbf{x}$ is the solution to our linear system, then for all i:

$$x_i = (b_i - \sum_{j \neq i} M_{ij} * x_j)/M_{ii}. \tag{28}$$

We can express this as

$$x_i^{(k+1)} + e_i^{(k+1)} = (b_i - \sum_{j \neq i} M_{ij} * (x_j^{(k)} + e_i^{(k)}))/M_{ii} \tag{29}$$

But by our method of relaxation

$$x_i^{(k+1)} = (b_i - \sum_{j \neq i} M_{ij} * x_j^{(k)})/M_{ii} \tag{30}$$

So

$$e_i^{(k+1)} = (-\sum_{j \neq i} M_{ij} * e_j^{(k)})/M_{ii} \tag{31}$$

If $emax^{(k)}$ is the element of the vector $\mathbf{e^{(k)}}$ with largest absolute value, then

$$\mid e_i^{(k+1)} \mid \leq \sum_{j \neq i} \mid M_{ij}/M_{ii} \mid * \mid emax^{(k)} \mid \tag{32}$$

and since the matrix is RDD

$$\mid e_i^{(k+1)} \mid < \mid emax^{(k)} \mid \tag{33}$$

This implies that all elements of $\mathbf{e^{(k+1)}}$ are not-greater in magnitude than the largest element of $\mathbf{e^{(k)}}$. Using this, we can show by induction that all the elements of $\mathbf{e^{(m)}}$ for any m are all not-greater in magnitude than the largest element of $\mathbf{e^{(0)}}$. From this we can conclude that the values of all the elements of the error vector are *bounded* between $-emax^{(0)}$ and $emax^{(0)}$.

We showed earlier that $x_i^{(k)}$ are never decreasing as we proceed from step k to step $k+1$. so by definition, the $\mathbf{e_i^{(k)}}$ never increase. Therefore, the $\mathbf{e_i^{(k)}}$ forms a monotonically decreasing sequence in k, and is bounded, so it must converge to some number (not necessarily 0) as k goes to infinity. If $\mathbf{e^{(k)}}$ converges, then by definition $\mathbf{x^{(k)}}$ also converges.

Now if $\mathbf{x^{(k)}}$ converges, that means that $\mathbf{x^{(k)} - x^{(k-1)}}$ converges to 0. This means that the $r_i^{(k-1)}/M_{ii}$ (the number we add to obtain $\mathbf{x^{(k)}}$ from $\mathbf{x^{(k-1)}}$) converges to zero. But in Southwell relaxation we are always picking the largest $r_i^{(k-1)}$, so if it converges to zero, the entire vector $\mathbf{r^{(k)}}$ must converge to 0. (This argument is valid as long as our method always picks a residual with a magnitude that is within some constant fraction of the largest residual. This allows us to choose the larges area weighted residual, if we wish.) If the residual has converged to 0 then we have obtained a correct solution. □

B Proof of Convergence for Column Diagonally Dominant Matrices

In this section we show that Southwell converges to the correct solution of a linear system if the matrix is strictly column diagonally dominant. This is the case for the modified system (21).

Here is an outline of the proof: At any step, we measure the size of the residual using the ℓ_1 norm.

$$\| \mathbf{r} \| \equiv \sum_i | r_i |$$

Whenever we relax any variable x_i it's residual r_i goes to zero. But it is possible that some of the other r_j increase. We show (using the fact that the matrix is column diagonally dominant) that this total residual increase (from all the other r_j) is less (by some constant factor) than the the decrease in r_i, so $\| \mathbf{r} \|$ decreases. Since $\| \mathbf{r} \|$ decreases each step, and must be non-negative, it must converge to some number.

We can show that $\| \mathbf{r} \|$ converges to $\mathbf{0}$ if we choose the largest r_i as in the Southwell method. Each relaxation step reduces $\| \mathbf{r} \|$ by some constant factor times r_i, and r_i is at least some constant fraction of $\| \mathbf{r} \|$, so $\| \mathbf{r} \|$ is reduced by some constant fraction of itself during each iteration. (The requirement that we pick the largest residual is actually more restrictive than we need to be. We could show convergence as long as the residual we pick is larger than some constant fraction of the total residual.) The formal proof follows.

Suppose that at step k , we have the residual $\mathbf{r^{(k)}}$. During step k we relax variable x_i. After step k, we have

$$r_i^{(k+1)} = 0. \tag{34}$$

We have set the other residuals to

$$r_j^{(k+1)} = r_j^{(k)} - \frac{M_{ji}}{M_{ii}} * r_i^{(k)}. \tag{35}$$

so

$$\| \mathbf{r}^{(k+1)} \| \leq \| \mathbf{r}^{(k)} \| - | r_i^{(k)} | + \sum_{j \neq i} | \frac{M_{ji}}{M_{ii}} * r_i^{(k)} | \tag{36}$$

$$\| \mathbf{r}^{(k+1)} \| \leq \| \mathbf{r}^{(k)} \| - | r_i^{(k)} | + | r_i^{(k)} | * \sum_{j \neq i} | \frac{M_{ji}}{M_{ii}} | \tag{37}$$

Since M is strictly column diagonally dominant,

$$\rho = \max_i \sum_{j \neq i} \mid \frac{M_{ji}}{M_{ii}} \mid < 1 \tag{38}$$

and

$$\| \mathbf{r}^{(k+1)} \| \leq \| \mathbf{r}^{(k)} \| - (1-\rho) \mid r_i^{(k)} \mid \tag{39}$$

Now, since Southwell picks the r_i which is largest, it must be true that

$$\frac{\| \mathbf{r}^{(k)} \|}{n} \leq \mid r_i^{(k)} \mid \tag{40}$$

so

$$\| \mathbf{r}^{(k+1)} \| \leq \| \mathbf{r}^{(k)} \| - \frac{(1-\rho)}{n} * \| \mathbf{r}^{(k)} \| \tag{41}$$

$$\| \mathbf{r}^{(k+1)} \| \leq q * \| \mathbf{r}^{(k)} \| \tag{42}$$

where

$$q = 1 - \frac{(1-\rho)}{n} < 1 \tag{43}$$

and thus after m steps

$$\| \mathbf{r}^{(m)} \| \leq q^m * \| \mathbf{r}^{(0)} \| \tag{44}$$

since q^m converges to 0, so does $\| \mathbf{r}^{(m)} \|$. □

C Overshooting

In section 2.2.2 we introduced an overshooting method based on computing the amount of radiosity a patch shoots, then receives, then reshoots etc. This method can also be understood (and analyzed more rigorously) from a numerical analysis perspective,

Given the linear radiosity system with n unknowns,

$$B_i = E_i + \rho_i \sum_j B_j F_{ij} \tag{45}$$

we can create a new linear system with n^2 unknowns. We replace each variable B_i with n variables B_{ik}. We replace the E_i with n identical numbers E_{ik}. And we solve the system:

$$B_{ik} = E_{ik} + \rho_i \sum_j B_{ji} F_{ij}. \tag{46}$$

The matrix of this new system has $n^2 * n^2$ elements. It is made up of n^2 blocks, each having n^2 elements. In general, the ij^{th} element of the IJ^{th} block is $-\rho_I F_{IJ}$. If $j = J$ add 1 to the diagonal entries. The remaining entries are 0. For example, if the original matrix had 3 entries, the new matrix would be:

$$\begin{bmatrix}
1-\rho_1 F_{1,1} & & & -\rho_1 F_{1,2} & & & -\rho_1 F_{1,3} & & \\
-\rho_1 F_{1,1} & 1 & & -\rho_1 F_{1,2} & & & -\rho_1 F_{1,3} & & \\
-\rho_1 F_{1,1} & & 1 & -\rho_1 F_{1,2} & & & -\rho_1 F_{1,3} & & \\
 & -\rho_2 F_{2,1} & & 1 & -\rho_2 F_{2,2} & & & -\rho_2 F_{2,3} & \\
 & -\rho_2 F_{2,1} & & & 1-\rho_2 F_{2,2} & & & -\rho_2 F_{2,3} & \\
 & -\rho_2 F_{2,1} & & & -\rho_2 F_{2,2} & 1 & & -\rho_2 F_{2,3} & \\
 & & -\rho_3 F_{3,1} & & & -\rho_3 F_{3,2} & 1 & & -\rho_3 F_{3,3} \\
 & & -\rho_3 F_{3,1} & & & -\rho_3 F_{3,2} & & 1 & -\rho_3 F_{3,3} \\
 & & -\rho_3 F_{3,1} & & & -\rho_3 F_{3,2} & & & 1-\rho_3 F_{3,3}
\end{bmatrix}$$

This new system has the same solution as the original system, but we can use this system to derive the overshooting algorithm. The n^2 variables of this system correspond to the n^2 values of shot radiosity ∇B_{jk}. The n^2 residuals of this system correspond to the n^2 values of unshot radiosity ΔB_{jk}.

If we relax one variable in this system, we are effectively "shooting" from a single patch to a single other patch. Instead of relaxing a single variable, we can relax a block of variables. Relaxing m variables entails changing those m variables simultaneously such that their m residuals all go to zero [8]. If we choose patch i and simultaneously relax the $2n-1$ variables $\{B_{ij}, B_{ji}\}$ we find that we can solve for this relaxing step in O(n) time. The computation done in this simultaneous relaxation step is the same as the computation done in an overshooting step.

If the original matrix is RDD, the extended matrix will also be RDD; relaxation and block relaxation will converge to the correct solution.