Fast Radiosity Solutions For Environments With High Average Reflectance

Gladimir V. Guimarães Baranoski[1] Randall Bramley[1] Peter Shirley[2]

[1] Indiana University, Bloomington, IN, USA
[2] Cornell University, Ithaca, NY. USA

1 Introduction

In radiosity algorithms the average radiance of n Lambertian patches is approximated by solving a linear system with n unknowns. When n is small (i.e. fewer than thousands of patches), general matrix methods like Gauss-Siedel can be used where the explicit $n \times n$ matrix can be pre-computed and stored [5]. When n is large, progressive techniques are used where the matrix rows or elements are recomputed as needed [4]. When n is very large (i.e. hundreds of thousands of patches), stochastic techniques can avoid computing or storing the n^2 elements of the matrix [10].

In applications where n is small enough to store the entire matrix in main memory, general matrix techniques will be faster than progressive techniques[3]. For "massing studies" [8] the lighting can be examined on simple geometric approximations of the environment being designed, and n can be very small. When the color scheme and lighting are being designed, the computationally expensive part (form factors) of the matrix in the linear system can be reused as the material properties are changed. For these applications the fastest possible general matrix solution is desirable.

This paper examines the Chebyshev method for solving linear systems, which for environments with high average reflectance can converge faster than the methods usually used in radiosity problems. We discuss some important characteristics of the linear systems in radiosity applications. We also look for solution methods that converge in small amounts of time, as opposed to a small number of iterations. For this discussion we assume a conventional RISC architecture, where coherent memory access is vital.

1.1 Radiosity System of Linear Equations

For an environment divided into n patches, the total spectral radiant power leaving a patch depends on the spectral radiant power emitted by the patch plus the spectral radiant power that is reflected. The spectral radiant power depends in turn on the total spectral radiant power leaving the other patches in the environment. The following system of equations represents the process of spectral radiant power transfer:

$$\Phi_j = \Phi_j^E + \rho_j \sum_{i=1}^{n} F_{ij}\Phi_i \qquad for \quad each \quad j = 1, 2...n \tag{1}$$

[3] Progressive techniques will initially converge faster because they can begin iterations before computing the entire matrix. If general matrix techniques are modified to gradually construct the matrix during initial iterations, then this advantage of progressive method goes away.

where:
Φ_j = total spectral radiant power leaving patch j (watts/nm).
Φ_j^E = spectral radiant power emitted by patch j (watts/nm).
ρ_j = reflectivity of patch j (dimensionless).
F_{ij} = fraction of energy leaving patch i hitting patch j (dimensionless).

Applying the mathematical derivation described in [1] we get the classical expression in terms of spectral radiant exitance, M, which holds for each patch in the environment:

$$M_j = E_j + \rho_j \sum_{i=1}^{n} F_{ji} M_i \qquad for \quad each \quad j = 1, 2...n \tag{2}$$

Radiosity, B, is the term used for radiant exitance in the computer graphics literature. Determining the radiant exitance (or radiosity) of each patch involves solving the linear system $GB = E$, given by:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$$

Iterative methods used to solve the radiosity system of linear equations can be divided into general matrix methods, which update all components of the solution vector on each iteration, and radiosity-specific methods [4] such as progressive refinement and overshooting methods, which update a single component on each iteration.

1.2 Eigenvalues Estimates

The Chebyshev method depends on eigenvalues estimates (see Appendix for relationship between eigenvalues and iterative solvers). The characteristic polynomial of a square $(n \times n)$ matrix G is given by $p(\lambda) = det(G - \lambda I)$, where I represents the identity matrix.

The zeros of $p(\lambda)$ are called *eigenvalues* or characteristic values of the matrix G. If $\mathbf{v} \neq 0$ is such that $(G - \lambda I)\mathbf{v} = 0$ holds, then $\mathbf{v}$ is called an *eigenvector* or characteristic vector of G corresponding to the eigenvalue λ.

Usually calculating the eigenvalues of a matrix G requires more computation than that required to solve the corresponding linear system. However we can obtain relatively inexpensive estimates of the eigenvalues using the Gerschgorin Circle Theorem [3]. This theorem says that the eigenvalues of G are contained within the union of n circles $S_i = \{z \in C \mid |z - g_{ii}| \leq \sum_{j \neq i}^{n} |g_{ij}|\}$, where **C** is the complex plane. The union of any k of these circles that do not intersect the remaining (n-k) must contain precisely k (counting multiplicities) of the eigenvalues.

In an environment formed by planar or convex surfaces ($F_{ii} = 0$) the radiosity matrix has all the main diagonal entries equal to one. So the centers of the circles are also have value one. In a closed environment ($\sum_{j=1}^{n} F_{ij} = 1$), the radius of S_i is given by ρ_i.

[4] We use the expression *radiosity-specific methods* to group methods specifically developed to solve the radiosity problem. Although those methods can be considered variations of numerical methods such as Southwell Iteration [7] or SOR [9], they have been particularly adjusted and finetuned to the radiosity case.

2 The Chebyshev Method

Gauss-Seidel [7] is a linear stationary method, which implicitly updates solutions by $B^{(j+1)} = TB^{(j)} + \tilde{\mathrm{E}}$, where T is an iteration matrix. Nonstationary methods have an implicit iteration matrix T_j which changes on each iteration. The Chebyshev method [12] [11] [2] is a nonstationary method based on residual polynomials (see Appendix). It is directly applicable to nonsymmetric matrices like the radiosity coefficient matrix. However it requires estimates μ and ν of the smallest and largest eigenvalues, $\lambda_{\max}$ and $\lambda_{\min}$, of the corresponding coefficient matrix. The iterative process is characterized by:

$$B^{(j+1)} = B^{(j)} + \Delta D^{(j)} \qquad \Delta D^{(j)} = q_j^{-1}(\Delta B^{(j)} + p_j \Delta D^{(j-1)}) \tag{3}$$

where $\Delta D^{(j)}$ is a correction vector, $\Delta B^{(j)} = E - G * B^{(j)}$ is the residual, and q_j and p_j are coefficients of the residual polynomials.

To obtain fastest reduction in the residual norm a residual polynomial method needs to select polynomials whose ordinates quickly go to zero on the spectrum of the coefficient matrix G as the degree of the polynomial increases. For radiosity problems the eigenvalues are all real and positive, so given a knowledge of an interval $[\mu, \nu]$ containing the spectrum of G we select polynomials P_k that have their maximum absolute value on $[\mu, \nu]$ minimal over all monic polynomials (polynomials with leading coefficient 1) of degree k. In addition to this "minimax" property, Chebyshev polynomials can also be computed using a three term recursion which implies that the iteration can be implemented using only three additional vectors of storage (see Appendix).

Adapting the classical Stiefel iteration [12] [11], the Chebyshev algorithm in radiosity context becomes the following:

1 **for** (each i)
2 $\quad B_i^{(0)} =$ starting guess
3 **compute** $\Delta B^{(0)} = E - G * B^{(0)}$
4 $\alpha = 2/(\nu - \mu)$
4 $\beta = (\nu + \mu)/(\nu - \mu)$
6 $\gamma = \beta/\alpha$
7 **for** (each i)
8 $\quad B_i^{(1)} = B_i^{(1)} + \gamma \Delta B_i$
9 **compute** $\Delta B^{(1)} = E - G * B^{(1)}$
10 $\omega = 4/(\nu + \mu)$.
11 $j = 1$
12 **while** (not converged)
13 $\quad \omega = (\gamma - \frac{1}{4\alpha^2}\omega)^{-1}$
14 $\quad$ **for** (each i)
15 $\quad\quad \Delta D_i = \omega * \Delta B_i^{(j)} + (\gamma\omega - 1)\Delta D_i$
16 $\quad\quad B_i^{(j+1)} = B_i^{(j)} + \Delta D_i$
17 $\quad$ **compute** $\Delta B^{(j+1)} = E - G * B^{(j)}$
18 $\quad j = j + 1$

In the above algorithm $\lambda_{\max}$ and $\lambda_{\min}$ have been replaced by ν and μ, where $0 < \mu \leq \lambda_{\min} < \lambda_{\max} \leq \nu$. The rate of convergence of Chebyshev is maximal when $\mu = \lambda_{\min}$ and $\nu = \lambda_{\max}$, and the method can even diverge if λ_{max} is underestimated by ν. Therefore to implement the Chebyshev method successfully the extremal eigenvalues of the matrix G must be estimated. However, since the Chebyshev polynomials grow

very rapidly if we underestimate the maximal eigenvalues, we will notice very quickly the sudden increase of the error norm. This would allow us to immediately reset the estimates and proceed.

Using the Gerschgorin Disk Theorem the extremal eigenvalues may be approximated by $1.0 \pm \rho_{max}$, where ρ_{max} is the highest reflectivity in the environment, which may correspond to the reflectivity of a single patch or of a group of patches. The increase of the highest reflectivity may not significantly change the overall reflectivity, expressed in terms of ρ_{avg},e.g. if only the reflectivity of few small patches is changed. In that case our experiments show that increasing the highest reflectivity does not significantly change the convergence. On the other hand if we increase the overall reflectivity of the environment, e.g. high albedo scenes, then slow convergence results. Our experiments show that using ρ_{avg} instead of ρ_{max} to estimate the eigenvalues gives better results for all the cases tested. Therefore our Chebyshev parameters are given by $\nu = 1.0 + \rho_{\text{avg}}$ and $\mu = 1.0 - \rho_{\text{avg}}$ in which ρ_{avg} is given by:

$$\rho_{\text{avg}} = \frac{\sum_{i=1}^{n} \rho_i A_i}{\sum_{i=1}^{i=n} A_i}, \tag{4}$$

where A_i is the area of patch i.

In all the cases tested the use of a starting guess which takes into account the ambient component gives better results than a starting guess which uses only the vector of emittances. To use this starting guess we replace line 2 of the above algorithm by $B_i^{(0)} = E_i + \rho_i Ambient$, where the ambient term is computed replacing ΔB_i by E_i in the ambient equation presented in [4].

3 Testing Parameters

We compared five algorithms, using explicitly stored form factors: Gauss-Siedel(GS), Progressive Refinement(PR) [7], Overshooting(FEDA) [6], Conjugate Gradient(CG) [1] and Chebyshev(CHEBY). The starting guesses were chosen in order to obtain the best possible rate of convergence for each tested algorithm. Consequently the initial error norm is not the same for curves on a single graph. The starting guess used for Gauss-Siedel and Conjugate Gradient algorithms was the vector of emittances and for the radiosity-specific methods we used a vector of zeros. For the Chebyshev algorithm we used the starting guess which takes into account the *Ambient* term, i.e. $B_i = E_i + \rho_i Ambient$.

The general matrix methods check the convergence after a complete sweep of the coefficient matrix, i.e. one iteration. The radiosity-specific methods perform this check after one relaxation step, i.e. one step of iteration. To make our measurement uniform we count steps of an iteration. In this context an iteration of a general matrix method corresponds to n steps of an iteration, where n is the order of the coefficient matrix.

To measure the time we start the clock at the beginning of a cycle of k steps of an iteration and stop it after k steps of an iteration. For the general matrix methods tested we use $k = n$. We check for k at each step in order to make the timing overhead the same for all methods. In addition all the error norms are computed outside of the timing cycles. The time measurements are given by elapsed CPU time on a SGI Challenge (20-R4400). All the algorithms were implemented using the same software guidelines to avoid differences that could affect the timing.

We use as our stopping criteria the largest unshot energy, i.e. the L_∞ norm of the vector with components $r_i A_i$, in which r_i represents the residual and A_i the area of patch i, given by:

$$\xi_\infty = \max_{1 \le i \le n} |r_i A_i| \tag{5}$$

If ξ_∞ is smaller than a given tolerance we stop the iterations. The value assigned to the tolerance depends on how visually close to true solution one wants the final image be. In general it is not necessary to use very low tolerance as in most numerical applications. We used a tolerance equal to 10^{-3}, but present the full convergence histories so that the methods can be compared for larger tolerances.

The test model used in our experiments consists of a sphere in the middle of a room. The sphere was divided into 128 patches and the faces of the surrounding cube were divided into 144 patches forming a total of 992 patches. The light source corresponds to 16 patches on the center of the "ceiling" of the cube. Assigning different values for the reflectivities varies ρ_{avg} and changing the sphere radius, $r = 1.0$ and $r = 2.0$, gives different densities (δ) for the coefficient matrix, 70% and 53% respectively.

4 Testing Results

Testing was performed to compare the performance of the five algorithms which are described in detail in [1]. In particular, we examine the effect of the matrix spectrum (which depends on the reflectivities in the scene). Numerical testing is necessary because most theory about the convergence rates of these methods deals with *asymptotic* convergence. For the low accuracy solutions needed in graphics radiosity problems, oftentimes an adequate solution is available before the asymptotic convergence region is approached.

The Gauss-Seidel and CG methods are *parameter-free*, that is, there are no algorithmic parameters which must be set by the user. Chebyshev requires estimates of the smallest and largest eigenvalues, but for this application those can be set automatically as was done here, by using $1 \pm \rho_{avg}$. The Feda method [6] can be fine-tuned by different choices of the overshooting parameter, in the same way that Gauss-Seidel is generalizable to the SOR method [9]. However, like the SOR method the optimal choice of parameters is unknown except for a few special cases. The version tested here automatically selects the overshooting parameter.

4.1 Steps of Iterations *vs* Time

Usually progressive refinement or overshooting converge in fewer steps of an iteration than the other three methods. However, counting steps of an iteration does not account for the differing amounts and types of work performed on each step. So a method that converges in fewer steps of an iteration may in fact require more overall time.

The experiments show that this distinction does occur in radiosity applications. Figure 1_{Left} shows that for test case A with $\rho_{avg} = 0.24$ progressive refinement methods converge in fewer steps of an iteration for $tol = 10^{-3}$. However, Figure 1_{Right}, showing the same convergence history as in Figure 1_{Left} but plotted against elapsed CPU time, shows that Gauss-Seidel and Chebyshev methods converge in less CPU time. When the overall reflectivity of the environment is increased, the difference becomes even more noticeable, as shown in Figure 2 for test case B with $\rho_{avg} = 0.46$.

Counter-intuitively, Gauss-Seidel and Chebyshev methods require more operations than the radiosity-specific methods, but require less CPU time. The main reason is the

differing amounts of pipelining and data locality the algorithms allow. Note that in progressive refinement methods, we search for the patch with largest amount of unshot radiosity, which involves traversing a potentially large amount of data without performing any operations on it that decrease the residual. The general matrix methods, by contrast, simply process each row of the matrix in order. Although this may mean processing rows whose corresponding patch has no unshot radiosity remaining, in practice performance is enhanced. By avoiding the search phase, the computations can be better pipelined by compilers, and all data which is brought into the processor is actually used in improving the solution rather than searching for the next row to handle.

Furthermore, the innermost loop of the progressive refinement and overshooting methods consists of a *saxpy* (vector update of the form $\mathbf{y} = \mathbf{y} + \alpha\mathbf{x}$, where $\mathbf{x}$ and $\mathbf{y}$ are vectors and α is a scalar) operation, which entails $4n$ memory references (n each for reading ρ_j, $\triangle B_j$, F_{ij}, and an additional n for writing $\triangle B_j$) and $3n$ floating point operations (flops). By contrast the Gauss-Seidel, Chebyshev and CG methods have an inner product as the innermost loop. For the last two methods this entails $2n$ memory references and $2n$ floating point operations, because quantities not indexed by the innermost loop are kept in registers and so do not require a memory reference. In particular, the carry-around scalar that the inner product is summed into, and the reflectivity ρ_i, are kept in registers. Hence the ratio of memory references to flops is 4/3 for the radiosity specific solvers, while the ratio is 1 for the general matrix methods. This means the general matrix methods better utilize data locality, getting more flops out of data in the cache or registers before having to read or write new cache lines. Note that the better data re-use of the general matrix methods is not *a priori* evident from examining the algorithms. It is possible that in progressive refinement, only a few patches are selected to shoot out radiosity over and over again. In that case, the data associated with those patches would likely remain in cache, potentially giving better data locality properties. Our experiments show that this is not the case in practice, however. Usually over 90% the patches are selected the same number of times, ± 1. Furthermore, the patches selected most often are only selected a few more times than the average.

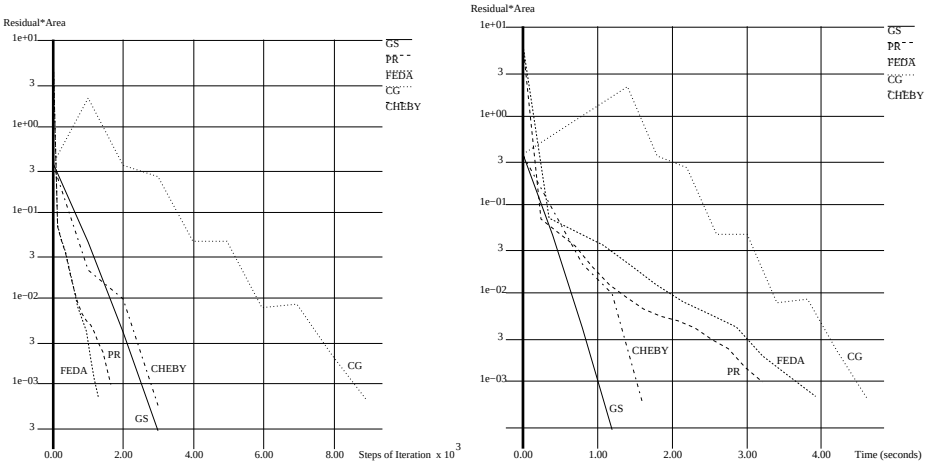

Fig. 1. Case A ($\rho_{avg} = 0.24$, $\delta = 53\%$) Left: steps of an iteration. Right: CPU time.

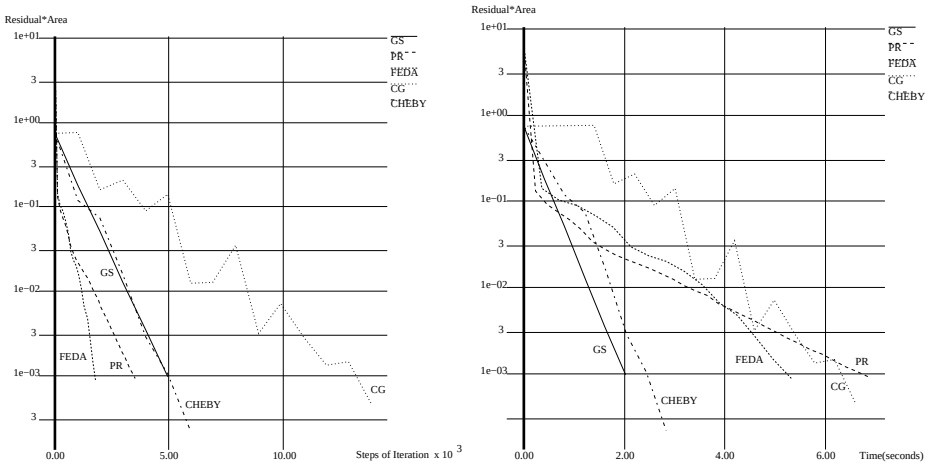

Fig. 2. Case B ($\rho_{avg} = 0.46$, $\delta = 53\%$) Left: steps of an iteration. Right: CPU time.

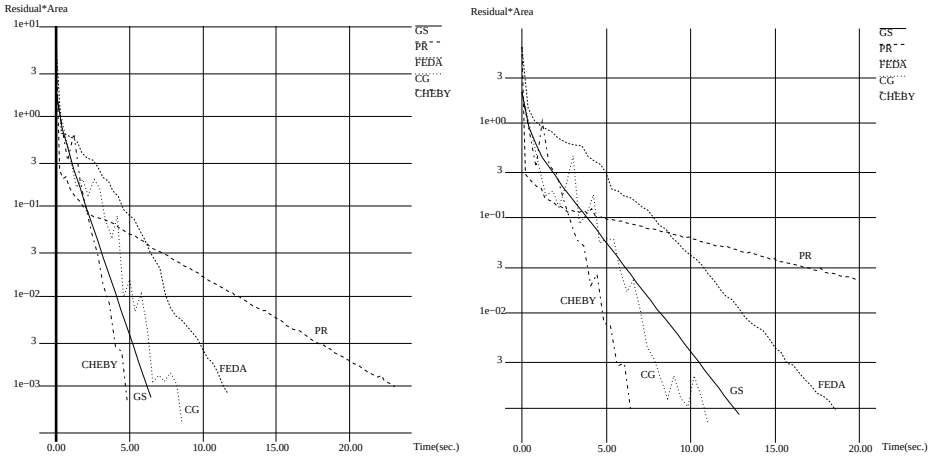

Fig. 3. Left: Case C ($\rho_{avg} = 0.77$, $\delta = 53\%$). Right: Case D ($\rho_{avg} = 0.88$, $\delta = 53\%$).

4.2 Effects of Reflectivity

Figures 2 and 3 show the performance of the various methods as matrix density (occlusion in the environment) is kept fixed at $\delta = 53\%$ and overall reflectivity is increased. Figure 4 does the same for $\delta = 70\%$. High reflectivities cause a larger number of interreflections, causing the eigenvalues to become more spread out which in turn slows

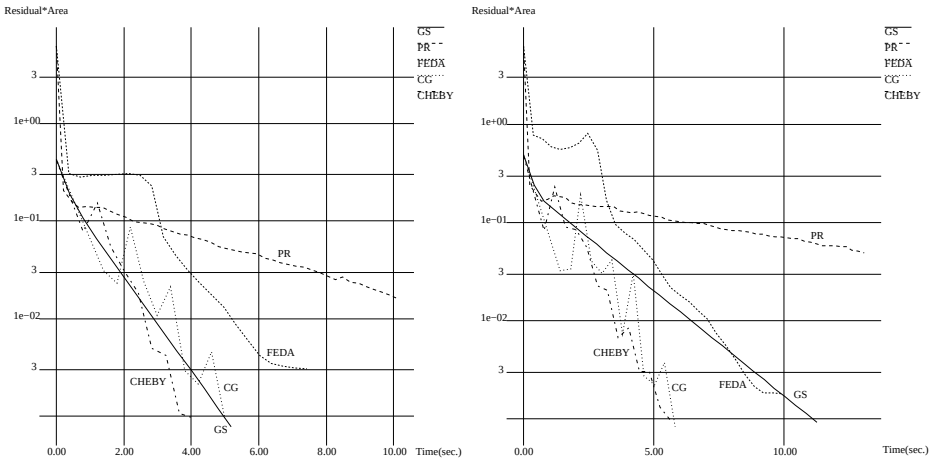

Fig. 4. Left: Case E ($\rho_{avg} = 0.78$, $\delta = 70\%$). Right: Case F ($\rho_{avg} = 0.89$, $\delta = 70\%$).

the convergence. Figure 5 shows the effects of the environment's overall reflectivity increase on the eigenvalue distribution. The increase of reflectivity is especially deleterious for PR. Because PR selects which patch to process on each step, it is a nonstationary method which actually changes its innermost loop depending on the specific data of the problem. This means it is not as amenable to analysis as the Chebyshev or CG methods, which are expected to take more steps of an iteration as the eigenvalues get spread out.

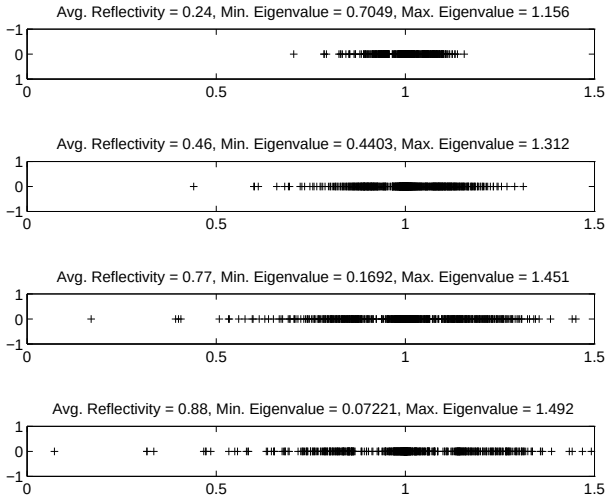

Fig. 5. Eigenvalue distribution as the overall reflectivity increases.

As reflectivity increases the spectral radius of the Gauss-Seidel iteration matrix ap-

proaches 1.0, and its relative performance decreases. The Gauss-Seidel method has a linear convergence rate that directly depends on the spectral radius of the iteration matrix. The Chebyshev method also has worsening performance, but relative to Gauss-Seidel it is not as sensitive to the increase in reflectivity. The adaptivity of the CG method makes its relative performance better as reflectivity increases; however, as Figure 4 shows only for the highest reflectivity and density levels tested did it become competitive with the Chebyshev method.

Table 1 summarizes the results for the test problems, and for each test case the time of the fastest algorithm is presented in boldface. Note that the Feda method implemented failed on test problems E and F, which have high density and reflectivity. In general, the Gauss-Seidel and Chebyshev methods are the fastest overall. These conclusions do not change when using different stopping tests or the other error norms described in [1].

Methods	Test Cases					
	A	B	C	D	E	F
CG	4.68	6.60	8.63	11.02	5.05	5.80
CHEBY	1.61	2.81	**4.81**	**6.45**	**4.05**	**5.64**
FEDA	3.92	5.32	11.73	18.47	7.45^{+}	9.92^{*}
GS	**1.20**	**2.01**	6.43	12.83	5.23	11.26
PR	3.21	6.85	23.03	50.56	22.53	50.43

Table 1. Algorithms performance (total time in seconds). The symbols $*$ and $+$ indicate failure to converge after 2478 and 3304 steps of an iteration respectively.

5 Conclusion and Future Research

Our experiments using explicitly stored form factors have shown that although radiosity-specific methods make rapid initial improvement, faster than any other method for limited tolerances (10^{-1}), they are slower than the general matrix methods for higher tolerances. Radiosity-specific methods require searching for a patch to shoot on each step, which can require traversing a large data structure. The disadvantage of general matrix methods of storing the form factors is compensated by the regularity of the computations which allows good pipelining and data locality.

The performance advantages of the general matrix methods are not as attractive when the form factors are computed on the fly, e.g. when n is large. In that case, the innermost loop consists of computing the form factors, which generally requires more flops than the matrix solving algorithms themselves. Avoiding even a few extra form factor computations by searching through rows for the most unshot radiosity may then give the edge to radiosity-specific methods such as Feda's method.

For environments with high average reflectance, which may occur in several applications, the rate of convergence is slower for all of the iterative methods used. Our experiments have also shown that the CG method and the Chebyshev method, with the estimates of the maximal eigenvalues described previously, represent the fastest approaches to handle those cases.

The experiments also show that selecting the "best" method is delicate, and no single method is superior in all cases. The relative performance depends on architectural

performance features such as pipelining and data locality as well as problem characteristics. Developing practical solution strategies will likely require implementing a variety of linear solvers, with the one actually used chosen at runtime dependent on problem parameters such as reflectivity and occlusion, and figuring out the best parallel implementation for shared memory multiprocessor workstations. It will also be necessary to bring more numerical linear algebra tools to bear on the problem.

Finally we believe that the understanding of the physical meaning of the eigenvalues and eigenvectors in the radiosity context may help us to obtain even faster approximations for the radiosities vector. Our future efforts will be focused on that question.

Appendix - Residual Polynomial Methods and Eigenvalues

A Eigenvalues and Eigenvectors

A.1 Definition and Basic Properties

An *eigenvector* v of a matrix G is a nonzero vector that does not rotate when G is applied to it. In other words, there is some scalar constant λ, an *eigenvalue* of G, such that $Gv = \lambda v$. Every square matrix G of order n has n possibly nondistinct complex eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$. When G is symmetric the eigenvalues are real-valued. The set $\sigma(G)$ of eigenvalues of a matrix is called its spectrum.

A.2 Relationship with Iterative Methods

Eigenvalues determine the convergence of iterative solvers used to solve linear systems such as $GB = E$. For linear stationary methods of the form $B^{(j+1)} = TB^{(j)} + \tilde{E}$, which includes Jacobi, Gauss–Seidel, and SOR, the eigenvalues of the iteration matrix T are the relevant ones. The matrix T is derived from the coefficient matrix G; for example in the Gauss-Seidel iteration $T = (D + L)^{-1}U$ where D, L, and U are the diagonal, strictly lower triangular, and strictly upper triangular parts of G, respectively. However, no connection need hold between the eigenvalues of T and those of G. Linear stationary methods converge if and only if $\rho(T) < 1$, where $\rho(T)$ is the size of the eigenvalue with largest magnitude. Furthermore, convergence is faster for smaller $\rho(T)$.

For nonstationary methods such as conjugate gradients or Chebyshev, the eigenvalues of the coefficient matrix G are the important ones. The Chebyshev method has convergence determined by the convex hull of the spectrum of G, which is determined by the extreme eigenvalues. For a matrix with positive real eigenvalues the largest (λ_{max}) and smallest (λ_{min}) eigenvalues completely determine convergence, which is faster for larger values of $(\lambda_{max} + \lambda_{min})/(\lambda_{max} - \lambda_{min})$. The Conjugate Gradient method's convergence is determined by the overall distribution of eigenvalues, and even for a given $\sigma(G)$ it is impossible to predict the exact number of an iterations CG will require. However, CG generally requires only $s + 1$ iterations when the eigenvalues occur in only $s < n$ clusters, and has faster convergence for larger values of $(\lambda_{max} + \lambda_{min})/(\lambda_{max} - \lambda_{min})$.

B Residual Polynomial Methods

Most iterative solvers for linear systems can be analyzed using *residual polynomials*. Consider the linear system $GB = E$ with an approximate solution $B^{(j)}$. An iterative

method can be seen as choosing a direction of motion $\Delta B^{(j)}$, then choosing a stepsize α to move in that direction:

$$B^{(j+1)} = B^{(j)} + \alpha_j \Delta B^{(j)}, \;\; j = 0, 1, \ldots \tag{6}$$

Because of the residual vector $E - G * B^{(j)}$ corresponds to the direction of steepest descent, it is most commonly used for $\Delta B^{(j)}$. An iterative method can then be constructed by choosing the stepsizes α_j to minimize a measure of the error in $B^{(k)}$, where k is the number of iterations made. Recursively substituting (6) into the definition of residual vector gives [12]:

$$\Delta B^{(k)} = P_k(G) \Delta B^{(0)} \tag{7}$$

$$P_k(\lambda) = (1 - \alpha_{k-1}\lambda)(1 - \alpha_{k-2}\lambda) \cdots (1 - \alpha_0 \lambda) \tag{8}$$

P_k is called a residual polynomial; note that $P_k(0) = 1$ necessarily holds.

If G is diagonalizable then $G = SDS^{-1}$ where S is a nonsingular matrix and D is the diagonal matrix with $\lambda_1, \lambda_2, ..., \lambda_n$ on its diagonal. In this case $P_k(G) = SP_k(D)S^{-1}$, and to obtain fastest reduction in the residual norm a residual polynomial method needs to select polynomials whose ordinates $P_k(\lambda_i)$ quickly go to zero as the degree of the polynomial increases. These ordinates give the reduction in the i^{th} eigencomponent of the residual after the k^{th} step of the iteration. In addition to this optimality property, a good residual polynomial should be computed using short recursions so that only a few of the previous residual vectors need be stored.

C Chebyshev Polynomials

The Chebyshev polynomials τ_k [3] for the interval $[-1, 1]$ are defined by $\tau_k(x) = \cosh[k \cosh^{-1}(x)]$, for each $k \geq 0$. Because they are orthogonal with respect to the weight function $w(x) = (1 - x^2)^{-1/2}$, Chebyshev polynomials can be computed using the three term recursion:

$$\tau_0(x) = 1, \; \tau_1(x) = x, \; \tau_{k+1}(x) = 2x\tau_k - \tau_{k-1}; \;\; k \geq 1.$$

Most importantly, Chebyshev polynomials have a *minimax* property: of all k^{th}–degree polynomials with leading coefficient 1, $2^{1-k}\tau_k$ has the smallest maximum norm in $[-1, 1]$. The value of its maximum norm is 2^{1-k}. Figure 10 shows the graphs of monic Chebyshev polynomials of degree 3,4 and 5.

Given an interval $[\mu, \nu]$ containing the eigenvalues of a matrix G the Chebyshev method uses residual polynomials based on monic Chebyshev polynomials shifted from the interval [-1,1] to the interval $[\mu, \nu]$, and scaled so that $P_k(0) = 1$. This gives:

$$P_k(\lambda) = \frac{\tau_k\left(\frac{\nu+\mu-2\lambda}{\nu-\mu}\right)}{\tau_k\left(\frac{\nu+\mu}{\nu-\mu}\right)}.$$

The three term recursion can similarly be translated to the new variables, giving the Chebyshev algorithm presented in the paper.

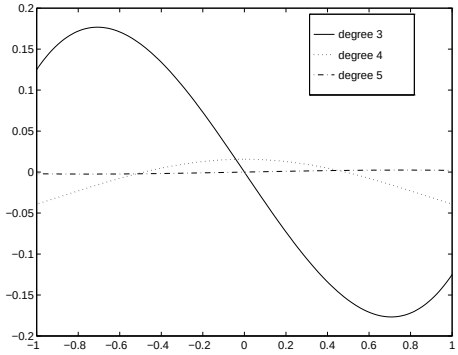

Fig. 6. Monic Chebyshev polynomials.

Acknowledgements

The work reported in this paper was supported in part by the *Conselho Nacional de Pesquisas* (CNPq, Brasil) and by *NSF* (Grant CDA 93-03189, USA).

References

1. G. V. BARANOSKI, R. BRAMLEY, AND P. SHIRLEY, *Iterative methods for fast radiosity solutions*, tech. rep., Indiana University, 1995.
2. R. BARRETT ET AL., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1 ed., 1994.
3. R. BURDEN AND J. FAIRES, *Numerical Analysis*, PWS-KENT Publishing Company, Boston, 5 ed., 1993.
4. M. COHEN, S. CHEN, J. WALLACE, AND D. GREENBERG, *A progressive refinement approach to fast radiosity image generation*, Computer Graphics, 22 (1988), pp. 75–84.
5. M. COHEN AND D. GREENBERG, *The hemi-cube: A radiosity solution for complex environments*, Computer Graphics, 19 (1985), pp. 31–40.
6. M. FEDA AND W. PURGATHOFER, *Accelerating radiosity by overshooting*, in Proc. of the Third Eurographics Rendering Workshop, Consolidation Express, June 1992, pp. 21–32.
7. S. GOERTLER, M. COHEN, AND P. SLUSALLEK, *Radiosity and relaxation methods*, IEEE Computer Graphics and Applications, 14 (1994), pp. 48–58.
8. D. GREENBERG, *Computers and architecture*, Scientific American, 264 (1991), pp. 104–109.
9. L. HAGEMAN AND D. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.
10. L. NEUMANN, *New efficient algorithms with positive definite radiosity matrix*, in Proc. of the Fifth Eurographics Rendering Workshop, June 1994, pp. 219–237.
11. Y. SAAD, A. SAMEH, AND P.SAYLOR, *Solving elliptic difference equations on a linear array of processors*, SIAM Journal of Scientific and Statistical Computing, 6 (1985), pp. 1049–1063.
12. E. STIEFEL, *Kernel polynomials in linear algebra and their numerical application*, in Further Contributions to the Solutions of Simultaneous Linear Equations and the Determination of Eigenvalues, National Bureau of Standards, Applied Mathematical Series - 49, 1958.