Contents

PART ONE

IMAGE FORMATION AND IMAGE MODELS

RENDERING

CHAPTER 1

Rendering a Diffuse World with Finite Elements

1.1 SOLUTIONS FOR RADIOSITY IN A WORLD OF DIFFUSE AREA SOURCES

We have a world of diffuse surfaces, illuminated by diffuse luminaires. We will construct some set of coordinates so that every point on every surface in the world is identified by its own coordinate (this is much easier to say than to do, but right now we need it only for formal reasons). At each point $\boldsymbol{x}$, there is some exitance $E(\boldsymbol{x})$ — which is zero, unless the point lies on a luminaire — and some radiosity $B(\boldsymbol{x})$. A luminaire typically has both exitance and radiosity, because it can both generate light internally and reflect light. At each point there is also an albedo, $\rho(\boldsymbol{x})$.

1.1.1 Setting up the Problem

The radiosity at each point is the sum of the exitance and the radiosity reflected at the point and caused by illumination from other points. From section **??**, we have the terminology of Figure **??** and

$$\text{visible}(\boldsymbol{x}, \boldsymbol{u}) = \begin{cases} 1 \text{ if } \boldsymbol{x} \text{ can see } \boldsymbol{u}, \\ 0 \text{ if } \boldsymbol{x} \text{ cannot see } \boldsymbol{u}. \end{cases}$$

Then we can write

$$\begin{aligned} B(\boldsymbol{x}) &= E(\boldsymbol{x}) + \rho(\boldsymbol{x}) \int_{\text{world}} \text{visible}(\boldsymbol{x}, \boldsymbol{u}) B(\boldsymbol{u}) \frac{\cos\theta_{\boldsymbol{x}} \cos\theta_{\boldsymbol{u}}}{\pi d_{\boldsymbol{u}\boldsymbol{x}}^2} dA_{\boldsymbol{u}} \\ &= E(\boldsymbol{x}) + \rho(\boldsymbol{x}) \int_{\text{world}} \Phi(\boldsymbol{x}, \boldsymbol{u}) B(\boldsymbol{u}) dA_{\boldsymbol{u}} \end{aligned}$$

Now we introduce a **linear operator** $\mathcal{K}$, which is a map that takes functions to functions. If we apply $\mathcal{K}$ to the function f — whose argument we suppress — we obtain the function $\mathcal{K}(f)$. When it seems helpful, we write the value of this function at the point $\boldsymbol{x}$ as $\mathcal{K}(f)(\boldsymbol{x})$ We define $\mathcal{K}$ by

$$\mathcal{K}(f)(\boldsymbol{x}) = \rho(\boldsymbol{x}) \int_{\text{world}} \Phi(\boldsymbol{x}, \boldsymbol{u}) f(\boldsymbol{u}) dA_{\boldsymbol{u}}$$

This operator is linear, because it takes the zero function to the zero function, and scaling its input scales its output proportionally. For our purposes, it is attractive because we can write

$$B(\boldsymbol{x}) = E(\boldsymbol{x}) + \mathcal{K}(B)(\boldsymbol{x})$$

or, equivalently,

$$B(\boldsymbol{x}) - \mathcal{K}(B)(\boldsymbol{x}) = E(\boldsymbol{x})$$

If we introduce another linear operator, $\mathcal{I}$ which takes each function to itself, we can write this expression as

$$(\mathcal{I} - \mathcal{K})B = E$$

1.1.2 The Neuman Series as an Inverse

Functions can often be thought of as souped-up vectors (vectors with more entries than is conventional, but otherwise familiar), and it is helpful to think of a linear operator as a (souped-up) matrix. Thus, for example, we can interpret $\mathcal{K}^2$ as the operator that takes f to $\mathcal{K}(\mathcal{K}(f))$. At present, we should like to know the inverse of the operator $\mathcal{I} - \mathcal{K}$, because we can then apply it to E to obtain a solution. Now assume that $\mathcal{K}$ is small (in some appropriate sense — we'll get to this). Then we can use the analogy with the series

$$(1 - x)^{-1} = 1 + x + x^2 + x^3 + \ldots$$

to propose an inverse. In particular, note that

$$(\mathcal{I} - \mathcal{K})(\mathcal{I} + \mathcal{K} + \mathcal{K}^2 + \mathcal{K}^3 + \ldots = (\mathcal{I} + \mathcal{K} + \mathcal{K}^2 + \mathcal{K}^3 + \ldots) - (\mathcal{K} + \mathcal{K}^2 + \mathcal{K}^3 + \mathcal{K}^4 + \ldots)$$

Now assume that

$$\lim_{k \to \infty} \mathcal{K}^k = 0$$

(for an operator to be the zero operator means that it takes every function to zero). Then we have that

$$(\mathcal{I} - \mathcal{K})(\mathcal{I} + \mathcal{K} + \mathcal{K}^2 + \mathcal{K}^3 + \ldots \mathcal{K}^k) = (\mathcal{I} - \mathcal{K}^k)$$

meaning that

$$\lim_{k \to \infty} (\mathcal{I} - \mathcal{K})(\mathcal{I} + \mathcal{K} + \mathcal{K}^2 + \mathcal{K}^3 + \ldots \mathcal{K}^k) = \mathcal{I}$$

and so we can write

$$(\mathcal{I} - \mathcal{K})^{-1} = \lim_{k \to \infty} (\mathcal{I} + \mathcal{K} + \mathcal{K}^2 + \mathcal{K}^3 + \ldots \mathcal{K}^k)$$

This yields a formal solution to our problem, which is known as the **Neumann series**. In particular, we have

$$\begin{aligned} B &= (\mathcal{I} + \mathcal{K} + \mathcal{K}^2 + \mathcal{K}^3 + \ldots)E \\ &= E + \mathcal{K}(E) + \mathcal{K}^2(E) + \mathcal{K}^3(E) + \ldots \end{aligned}$$

This series has a natural interpretation. You should think of the kernel as a device that takes illumination through "one bounce". The first term is the exitance; the second is the component of radiosity due to incoming *exitance* — i.e. the result of a local shading model; the third term is the radiosity obtained by computing a local shading model with the second term as the source; the fourth uses the third as a source; etc. This is usually thought of as the source term plus the direct illumination plus the "one bounce" term plus the "two bounce" term and so on.

1.1.3 The Smallness of $\mathcal{K}$

The operator $\mathcal{K}$ has a natural physical intepretation — it gives "one bounce". This means that, if we have an exitance $E(\boldsymbol{x})$, then $\mathcal{K}(E)$ is the **direct illumination radiosity** — the radiosity at each surface resulting from direct contributions from the exitance of the luminaire, and nothing else. Another way to think about it is that it is the prediction of radiosity that a local shading model would make.

This leads to a very useful bound. The total power radiated by luminaires in a world with exitance $E(\boldsymbol{x})$ is

$$\int_{\text{world}} E(\boldsymbol{x}) dA_{\boldsymbol{x}}$$

(we're assuming that E is non-negative as it must be in physically real situations). Now this power may be absorbed when it is redistributed, but extra power can't be created. This means that

$$\int_{\text{world}} \mathcal{K}(E)(\boldsymbol{x}) dA_{\boldsymbol{x}} \leq \int_{\text{world}} E(\boldsymbol{x}) dA_{\boldsymbol{x}}$$

In almost every world equality is not attained, because some power is either absorbed or radiated into free space. This gives some quite strong properties. For any function $f(\boldsymbol{x})$ we take its absolute value at every point to get the function $|f|(\boldsymbol{x})$. The **L1 norm** of the function is

$$|f|_1 = \int_{\text{world}} |f|(\boldsymbol{x}) dA_{\boldsymbol{x}}$$

Now we have that

$$|\mathcal{K}(f)|_1 = \int_{\text{world}} |\mathcal{K}(f)(\boldsymbol{x})| \, dA_{\boldsymbol{x}} \leq \int_{\text{world}} \mathcal{K}(|f|)(\boldsymbol{x}) dA_{\boldsymbol{x}} \leq |f|_1$$

Now because we will never deal with worlds where the albedo is one at every point, we are guaranteed that some power is lost and that, in turn

$$\frac{|\mathcal{K}(f)|_1}{|f|_1} \leq \lambda \leq 1$$

for some λ and *for any function* f. The smallest λ for which this expression is true for all functions is known as the **L1 norm of the operator** $\mathcal{K}$ and is written as $|\mathcal{K}|$. Because this norm is less than one, $\mathcal{K}$ makes functions "smaller" (by at least a factor of λ). In particular, we have that

$$|\mathcal{K}^{n+1}(f)|_1 \leq \lambda^{n+1} |f|_1$$

meaning that this shrinkage can be very fast if λ is small. In particular, if λ is small, the Neumann series must converge quickly. We consider truncated estimates of the series

$$B^{(n)}(\boldsymbol{x}) = (\mathcal{I} + \mathcal{K} + \ldots \mathcal{K}^n) E(\boldsymbol{x})$$

and find that

$$|B^{(n+1)} - B^{(n)}|_1 \leq \lambda^{n+1} |E|_1$$

meaning that if λ is small, a very small number of iterations of the Neumann series give an very good approximation to the solution, because the contribution of the next term is exponentially small. It is an experimental fact that for most worlds of interest to computer graphics, λ is small (0.2 or less is a good guess).

The Norm of the Radiosity.

You should notice that the norm of $\mathcal{K}$ being less than one does not mean that the norm of B is smaller than the norm of E. This should look puzzling to you (or at least, most people find it puzzling at first sight!). In fact, the only bound we have on the norm of B is given by

$$\begin{aligned} |B| &\leq |1 + \alpha + \alpha^2 + \ldots||E| \\ &= \frac{1}{1-\alpha}|E| \end{aligned}$$

Why should this look puzzling? the implication is that, somehow, the radiosity solution "contains" more power than the original exitance. In fact, this effect has to do with careless thinking about what the radiosity solution represents. It contains all photons that leave the source plus photons that have left the source and bounced once plus photons that have left the source and bounced twice, etc. This means that the norm of the solution counts power multiple times — which is why the solution can appear to have more power than the exitance.

1.1.4 Methods for Constructing Approximate Solutions

The Neuman series isn't really an algorithm, because the integrals are mostly too difficult to do in closed form. We need to build an approximate representation of the problem that is manageable. The main difficulty is that the solution is a function (or, equivalently, an infinite-dimensional vector). For most practical problems, the solution can be represented only approximately.

There are three significant decisions to make.

- **What kind of problem do we wish to solve?** we could be trying to produce a single rendered image, or a sequence of frames. The two cases are very different indeed. If we wish to produce a sequence of frames, it is usually a good idea to determine a solution for radiosity at every point in the model world. If we wish to produce only a single frame, significant savings are possible, because some light does not affect the image (the camera might be in a cupboard on the fifth floor; do we really need to know the distribution of light for every surface on every floor to render the image?). We discuss these ideas in section **??**, once we have dealt with basic algorithms.

- **How do we set up a finite-dimensional approximation to the radiosity problem?** it is fairly natural to want to approximate an infinite dimensional linear problem with a finite dimensional approximation (it's easier to store, for one thing). There are several different choices that can be made here, each with rather specific consequences in computational cost. We review these choices next (in section **??**).

- **How do we solve the approximate problem?** almost always, the approximation results in a very large, very sparse linear system. If one is clever with systems like this, it is possible to obtain solutions much more efficiently than, say, Gauss elimination. We review methods in section **??**.

1.2 FINITE-DIMENSIONAL APPROXIMATIONS TO THE RADIOSITY PROBLEM

We wish to represent a solution as an element of a finite-dimensional space. Let us write the basis as $\{\phi_1(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \ldots, \phi_n(\boldsymbol{x})\}$. Now our representation of the radiosity is

$$B(\boldsymbol{x}) \approx \sum_{j=1}^{n} b_j \phi_j(\boldsymbol{x}) = \hat{B}$$

where the b_i are coefficients that we must adjust to obtain the best representation.

The problem we have is of the form

$$B - \mathcal{K}(B) = E$$

where $\mathcal{K}$ is a linear operator. The expression $B - \mathcal{K}(\hat{B}) - E$ is known as the **residual** (or, sometimes, **residual error**). This is a *function*, which we can write as $R(\boldsymbol{x})$. If we replace B with a sum of basis functions, we cannot expect that there is a choice of b_j — of which there is a finite number — which gives a residual that is zero for every one of the infinite number of values of $\boldsymbol{x}$. The best we can hope for is that the residual approximates the zero function — but in what sense? Different choices lead to different linear systems.

1.2.1 Error Criteria

Collocation.

One way a function could approximate the zero function is that it could be zero at some set of isolated points. We choose n points $\boldsymbol{x}_i$, and require that $R(\boldsymbol{x}_i)$ be zero for each of these points. Of course, this doesn't mean that the residual is zero at other points. Requiring that the residual vanish at some fixed set of points is known as **collocation**. Applying this criterion to our problem, we have that for each i

$$\begin{aligned}
\hat{B}(\boldsymbol{x}_i) - \mathcal{K}(\hat{B})(\boldsymbol{x}_i) - E(\boldsymbol{x}_i) &= \sum_j b_j \left(\phi_j(\boldsymbol{x}_i) - \mathcal{K}(\phi_j)(\boldsymbol{x}_i)\right) - E(\boldsymbol{x}_i) \\
&= \sum_j \left(b_j \left\{ (\phi_j(\boldsymbol{x}_i) - \rho(\boldsymbol{x}_i) \int_{\text{world}} \Phi(\boldsymbol{x}_i, \boldsymbol{u}) \phi_j(\boldsymbol{u}) dA_{\boldsymbol{u}} \right\} \right) - E(\boldsymbol{x}_i) \\
&= 0
\end{aligned}$$

Now this is a system of linear equations in the unknowns b_i.

Galerkin Methods.

The space spanned by the basis functions $\phi_j(\boldsymbol{x})$ is a finite-dimensional linear subspace of an infinite-dimensional vector space of functions. The residual lives in

this infinite-dimensional vector space, too. The residual could approximate zero by being zero *in the space spanned by our basis functions.*

What this means is that $\int R(\boldsymbol{x})\phi_j(\boldsymbol{x})d\boldsymbol{x}$ is zero for any choice of j. Since $\hat{B} = \sum_j b_j\phi_j(\boldsymbol{x})$, we have for any choice of i,

$$\begin{aligned}\left(\int \left\{\hat{B}(\boldsymbol{x}) - \mathcal{K}(\hat{B})(\boldsymbol{x}) - E\right\}\phi_i(\boldsymbol{x})d\boldsymbol{x}\right) &= \sum_j \left(\int \phi_j(\boldsymbol{x})\phi_i(\boldsymbol{x}) - \mathcal{K}(\phi_j)(\boldsymbol{x})\phi_i(\boldsymbol{x})dA_{\boldsymbol{x}}\right) b_j - \int E(\boldsymbol{x})\phi_i(\boldsymbol{x})d\boldsymbol{x} \\ &= 0\end{aligned}$$

Now this, too, is a linear system in the unknowns b_i. We give two important forms in section **??**

1.2.2 What is the Basis to be?

Radiosity is an example of a general class of problem called a **Fredholm equation of the second kind**. However, it has quite special features. The most important is scale — we usually require solutions on large, complex domains. We might need millions of basis elements, which means that any efficiency is attractive. Notice that the linear systems that arise both from collocation and in Galerkin methods involve integrals of the basis functions. If the basis function has broad **support** (the region of the domain over which the function is not zero), estimating these integrals could be expensive. Furthermore, the domain of the basis function is typically modelled as a set of meshes of polygons or surface patches. Any basis that didn't take this into account might also yield very difficult numerical integration problems.

All this suggests building basis functions by dividing up the domain into small pieces, and then using functions that are zero on all but a small number of these pieces (which we shall call **elements**). These functions are particularly easy to manage if the pieces of domain overlap only on boundaries. Such functions are often known as **finite elements**.

There are many different finite elements, often distinguished by their continuity. It is usually easy to smooth a radiosity solution (see section **??**), so this doesn't much matter for our purposes. This suggests using a function that is non-zero on only one domain element, and zero elsewhere. By far the most useful such function is constant — one is a good choice — within a domain element, and zero elsewhere. For definiteness, we assume that our domain is divided into elements that overlap only on boundaries, and that there is one basis function corresponding to each element. This basis function will be one over the element, and zero elsewhere. The elements are not required to be flat.

1.2.3 The Linear System for a Galerkin Method

We write the i'th element as D_i. The i'th basis function $\phi_i(\boldsymbol{x})$ is one for $\boldsymbol{x} \in D_i$ and zero otherwise. Our equations are

$$\begin{aligned}
\int \left(\hat{B}(\boldsymbol{x}) - \mathcal{K}(\hat{B})(\boldsymbol{x})\right) \phi_i(\boldsymbol{x}) d\boldsymbol{x} &= \sum_j \left(\int \phi_j(\boldsymbol{x})\phi_i(\boldsymbol{x}) - \mathcal{K}(\phi_j)(\boldsymbol{x})\phi_i(\boldsymbol{x}) dA_{\boldsymbol{x}} \right) b_j \\
&= \sum_j \left(A_j \delta_{ij} - \int_{D_i} \rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}} \right) b_j \\
&= \int E(\boldsymbol{x})\phi_i(\boldsymbol{x}) d\boldsymbol{x} \\
&= \int_{D_i} E(\boldsymbol{x}) dA_{\boldsymbol{x}}
\end{aligned}$$

where A_i is the area of the i'th element, and δ_i^j is one if $i = j$ and zero otherwise. In this case, there are two important possibilities: We can work in terms of radiosity, or we can work in terms of radiant power.

Working with Radiosity.

We can interpret b_j as representing the average radiosity of the j'th element, because $\hat{B}(\boldsymbol{x}) = \sum_j b_j \phi_j(\boldsymbol{x})$. This suggests introducing another set of coefficients, e_j, representing the average exitance of each element. If we do this, the i'th equation of the linear system can be written as

$$\sum_j \left(\delta_{ij} A_j - \int_{D_i} \rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}} \right) b_j = e_i A_i$$

Now write

$$S_{ij} = \int_{D_i} \rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}}$$

Our equation has become

$$\sum_j \left(\delta_{ij} A_j - S_{ij} \right) b_j = e_i A_i$$

which is recognisably a linear system. However, the area terms are inconvenient; it is easy to divide through and get

$$\sum_j \left(\delta_{ij} - \frac{1}{A_i} S_{ij} \right) b_j = e_i$$

(if you're puzzled about the fate of the A_j in the first term, remember the definition of δ_{ij}). Now we form a matrix $\mathcal{K}^{(r)}$ whose i, j'th element is

$$\frac{1}{A_i} S_{ij}$$

(watch that area term!) and can then write the equation in matrix vector form using $\mathcal{I}$ for the identity matrix, $\boldsymbol{B}$ for the vector whose j'th component is b_j, etc. and get

$$(\mathcal{I} - \mathcal{K}^{(r)})\boldsymbol{B} = \boldsymbol{E}$$

The Linear System for Radiant Power with a Galerkin Method. We wrote the j'th equation of the linear system for a Galerkin method as

$$\sum_j (\delta_{ij} A_j - S_{ij})\, b_j = e_i A_i$$

Instead of dividing out the area term, we could work in terms of radiant power. If b_j is the radiosity of the j'th element, then $b_j A_j$ is its radiant power (i.e. the power transferred by radiosity) and $e_j A_j$ is its exitant power (i.e. the power transferred by exitance). Write $\beta_j = b_j A_j$ and $\epsilon_j = e_j A_j$, and we get

$$\sum_j \left(\delta_{ij} - (\frac{1}{A_j}) S_{ij}) \right) \beta_j = \epsilon_j$$

Now we form a matrix $\mathcal{K}^{(p)}$ whose i, j'th element is

$$\frac{1}{A_j} S_{ij}$$

(notice that we are dividing by a different A here, A_j rather than the A_i we used above). We can then write the radiant power form as

$$(\mathcal{I} - \mathcal{K}^{(p)})\boldsymbol{\beta} = \boldsymbol{\epsilon}$$

Properties of the Kernel matrix.

There are strong constraints on both $\mathcal{K}^{(p)}$ and $\mathcal{K}^{(r)}$. The easiest way to see these constraints is to separate terms due to the geometry of the elements and those due to the reflectance. We defined

$$S_{ij} = \int_{D_i} \rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}}$$

and formed our matrices out of these terms. Now the term

$$F_{ij} = \int_{D_i} \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}}$$

encapsulates the geometric relationship of the two elements i and j. For any pair of elements, we must have

$$0 \leq S_{ij} \leq F_{ij}$$

because the terms inside the integral are always non-negative, and because $\rho(\boldsymbol{x}) \leq 1$. This means that we should be able to turn bounds on F_{ij} into bounds on S_{ij}, too.

Assume that element j has constant radiosity b_j. The radiosity at point $\boldsymbol{x}$ on element i due to this is

$$\rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} b_j$$

and the total power radiated by element i and caused by this radiosity is

$$P_{\text{leaving } i \text{ due to } j} = \int_{D_i} \left(\rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} b_j \right) dA_{\boldsymbol{x}} = S_{ij} b_j$$

The total power on element j is $b_j A_j$. All this means that the fraction of power leaving j that then proceeds to leave i is

$$\frac{P_{\text{leaving } i \text{ due to } j}}{b_j A_j} = \frac{1}{A_j} S_{ij}$$

Now if i had a uniform albedo of one, it would radiate all power that arrived. This means that the fraction of power leaving j that arrives at i is

$$\frac{1}{A_j} F_{ij}$$

Now the fraction of power leaving j that arrives at something is

$$\sum_i \frac{1}{A_j} F_{ij}$$

and this must be less than one, because power isn't created. In turn, we have

$$\sum_i \frac{1}{A_j} S_{ij} \leq \sum_i \frac{1}{A_j} F_{ij} \leq 1$$

Now this has implications for both $\mathcal{K}^{(p)}$ and for $\mathcal{K}^{(r)}$. If we write the i, j'th element of $\mathcal{K}^{(p)}$ as $k_{ij}^{(p)}$ and of $\mathcal{K}^{(r)}$ as $k_{ij}^{(r)}$, we have

$$\sum_i k_{ij}^{(p)} = \sum_i \frac{1}{A_j} S_{ij} \leq \sum_i \frac{1}{A_j} F_{ij} \leq 1$$

and also

$$\sum_j k_{ij}^{(r)} = \sum_j \frac{1}{A_i} S_{ij} \leq \sum_j \frac{1}{A_i} F_{ij} = \sum_j \frac{1}{A_i} F_{ji} \leq 1$$

This means that, for every column of $\mathcal{K}^{(p)}$, the sum of column elements must be less than one. For every row of $\mathcal{K}^{(r)}$, the sum of row elements must be less than one.

1.2.4 The Linear System for Collocation

The linear system for collocation admits bounds as well, but less easily so. We write the i'th element as D_i. The i'th basis function $\phi_i(\boldsymbol{x})$ is one for $\boldsymbol{x} \in D_i$ and zero otherwise. For the collocation method, we must also choose points at which the residual must vanish. We choose one per element, and use the notation $\boldsymbol{x}_i$ for the point lying inside the i'th element. For this choice of basis, the collocation method yields the linear system

$$\begin{aligned} E(\boldsymbol{x}_i) &= \hat{B}(\boldsymbol{x}_i) - \rho(\boldsymbol{x}_i) \int \Phi(\boldsymbol{x}_i, \boldsymbol{u}) \hat{B}(\boldsymbol{u}) dA_{\boldsymbol{u}} \\ &= \sum_j \left(\delta_{ij} - \rho(\boldsymbol{x}_i) \int_{D_j} \Phi(\boldsymbol{x}_i, \boldsymbol{u}) dA_{\boldsymbol{u}} \right) b_j \end{aligned}$$

which we can write as

$$(\mathcal{I} - \mathcal{K}^{(c)})\boldsymbol{B} = \boldsymbol{E}$$

where the i'th element of $\boldsymbol{E}$ is $E(\boldsymbol{x}_i)$. We can bound the row sums of $\mathcal{K}^{(c)}$. For any choice of i, we have

$$\sum_j k_{ij}^{(c)} = \sum_{j \in \text{elements}} \left(\rho(\boldsymbol{x}_i) \int_{D_j} \Phi(\boldsymbol{x}_i, \boldsymbol{u}) dA_{\boldsymbol{u}} \right) \leq \rho(\boldsymbol{x}_i) \int_{\Omega} \Phi(\boldsymbol{x}_i, \boldsymbol{u}) dA_{\boldsymbol{u}}$$

(because no ray leaving the point $\boldsymbol{x}_i$ sees more than one object). But this latter integral is easy to bound.

$$\begin{aligned} \rho(\boldsymbol{x}_i) \int_{\Omega} \Phi(\boldsymbol{x}_i, \boldsymbol{u}) dA_{\boldsymbol{u}} &\leq \rho(\boldsymbol{x}_i) \frac{1}{\pi} \int_{\Omega} \cos\theta d\omega \\ &= \rho(\boldsymbol{x}_i) \end{aligned}$$

so for any row of $\mathcal{K}^{(c)}$, the row sum is less than one.

1.3 COMPUTING THE KERNEL MATRIX

Depending on the method we adopt, the kernel matrix will contain a set of integrals that need to be computed. There are two important integrals: We must be able to compute either

$$C_{ij} = \int_{D_j} \Phi(\boldsymbol{x}_i, \boldsymbol{u}) dA_{\boldsymbol{u}}$$

(for the collocation method) or

$$S_{ij} = \int_{D_i} \rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}}$$

(for the Galerkin method). Notice the distinctive property of the Galerkin method, that more integration is required — this is not usually commented on in writings on finite element methods for radiosity.

These integrals tend to be difficult, either in closed form or numerically. For a very small number of these integrals, the answer is known in closed form. Some

examples are given in figure **??**; others are known (see (**?**, **?**)). Generally, these integrals are hard to do without a symmetry or some other simplifying property of the domain. The numerical difficulties generally come from occlusion. Occluding objects break up the domain (which is equivalent to regions of Φ that are zero), and the presence of these objects is hard to predict — we usually have to search for them.

1.3.1 Breaking the Domain into Boxes

There is a big difference between integration over a two-dimensional domain and integration over a four-dimensional domain. In the case of a 2D domain, it is conceivable that one might break up the domain into individual boxes, and then estimate the integral by summing the values in the boxes.

The simplest method for estimating C_{ij} breaks up the domain D_j of the integral into some set of pieces B_k, each with area A_k and containing a point $\boldsymbol{p}_k$, and then estimates

$$\begin{aligned} C_{ij} &= \int_{D_j} \Phi(\boldsymbol{x}_i, \boldsymbol{u}) dA_{\boldsymbol{u}} \\ &\approx \sum_k \Phi(\boldsymbol{x}_i, \boldsymbol{p}_k) A_k \end{aligned}$$

This is easiest to do in practice if the domain is a polygon — triangles or quadrilaterals are particularly easy, see figure **??**. You should think of samples as rays from $\boldsymbol{x}_i$ to $\boldsymbol{p}_k$, and occlusion is detected (or Φ is zero, equivalently) when one of these rays strikes some intervening object. It should be clear that, if we are unlucky, a very poor estimate of the integral can result (think about a view through an occluding fencing mask — we could have every sample hit the mask, resulting in a zero estimate, or every sample pass through the holes in the mask, resulting in a massive overestimate).

The process allows a form of error control if we can produce increasingly fine subdivisions easily. In particular, we compute an estimate of C_{ij} for a subdivision, then compute it for a finer subdivision. If these two estimates are close and if the finer subdivision has elements no larger than some threshold, then we accept the estimate for the finer subdivision and stop, otherwise we subdivide again. This process doesn't guarantee a good estimate (again, think about the fencing mask — the subdivision procedure could result in rays that still hit the mask, etc.).

You should notice that this technique is equivalent to the simplest method for estimating a 1D integral — replace the original function with an approximation consisting of strips of constant height, and estimate the integral of the approximation. Other 1D methods — for example, the trapezoidal rule or Simpsons method — approximate the original function with more complicated interpolates (linear and quadratic, respectively), and integrate the approximation. This isn't usually done for integrals in higher dimensions, because constructing the approximations is finicky.

A technique known as **Richardson extrapolation** can be used to polish these estimates slightly, though it isn't much used in practice, probably because it isn't usually necessary to have the highest available accuracy in estimates of

the integral. Imagine plotting the estimate of the integral against some parameter representing the subdivision. This curve should have an assymptote — the correct value — as the subdivision gets finer and finer. In Richardson extrapolation, we construct an extrapolate that estimates that assymptotic value.

Estimating S_{ij} by breaking up the domain is usually less successful. This is because the domain has higher dimension. The problem is most easily seen by assuming that the two components of the domain, D_i and D_j, are quadrilaterals. Our subdivision is going to involve splitting each element into four children (rather like a quadtree). We shall use the notation B_{ik} for components of D_i, B_{jl} for components of D_j, $\boldsymbol{p}_{ik}$ (resp. $\boldsymbol{p}_{jl}$) for the sample points inside the components, and A_{ik} (resp. A_{jl}) for the area of the components. Then the estimate is

$$\begin{aligned} S_{ij} &= \int_{D_i} \rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}} \\ &\approx \sum_k \sum_l \rho(\boldsymbol{p}_{ik}) \Phi(\boldsymbol{p}_{ik}, \boldsymbol{p}_{jl}) A_{ik} A_{jl} \end{aligned}$$

This involves computing the value of Φ for each pair consisting of a point from one side and a point from the other (figure **??**). If we subdivide, we typically have to subdivide both sides (because we typically cannot tell whether it is better to subdivide one or the other) and so the number of elements in the sum goes up by a factor of 16 and quickly becomes unmanageable. Again, you can think of the samples as rays joining $\boldsymbol{p}_{ik}$ and $\boldsymbol{p}_{jl}$; if there is an occlusion, one of these rays hits an intervening object. It should be clear that this process doesn't guarantee a good estimate either — use the fencing mask example again.

The Hemi-Cube.

Recall that in the discussion of foreshortening, we said that a receiver couldn't distinguish between two sources that appeared exactly the same when viewed from the receiver. This is what makes it possible to change the variable in the integral as we did in chapter **??**, section **??**, where we started with an integral over $\boldsymbol{x}_i$'s incoming hemisphere and turned it into an integral over D_j. Changing variables in the integral leads to a very efficient trick for computing C_{ij}.

If we change variable to the top half of a cube, centered on $\boldsymbol{x}_i$, computing an estimate of the integral is quite a lot like rendering. We have to obtain a set of points lying on a projection of the element to the cube; we will then weight these points, and add them up. Obtaining the points is the hard part of the integral, because it corresponds to determining which parts of D_j are visible.

This gives an estimate of S_{ij} as

$$C_{ij} \approx \sum_{\text{cube faces}} \left(\sum_{k \in \text{pixels seeing } j} \omega_k \right)$$

where ω_k is the solid angle subtended at j by the k'th pixel (which is different for different pixels in the cube).

What is important here is that we could do many different elements at the same time — in effect, we could compute C_{ij} for every j efficiently. We do this by

setting up five different cameras, each with a focal point at $\boldsymbol{x}_i$ and with clipping regions in their image planes set up so as to form a cube (figure **??**). We then render all elements potentially visible from $\boldsymbol{x}_i$ with these cameras. We flat shade the elements, using as a shading value a pointer to the elements. This means that we end up with five buffers (one per view), each of which contains at every pixel a pointer to the element visible from that pixel — call these the **item buffers** (because the trick of rendering pointers into buffers is known as **item buffering**). We keep corresponding buffers containing the product of the solid angle subtended by each pixel times the direction cosine (all of this is independent of $\boldsymbol{x}_i$, so can be computed once and kept). Call these buffers the **constant buffers**. To obtain an estimate of the integrals C_{ij} for all j elements, we set the value for each integral to be zero. We then walk all five buffers: We look up the polygon seen by the working pixel in the item buffer, and then add the solid angle subtended by the working pixel to the integral estimate for that polygon (algorithm **??**).

1.3.2 Integration using Random or Pseudo-Random Methods

Dividing a domain into near-constant patches is a poor way to estimate an integral in a high-dimensional space. One can usually get away with this method in one or two dimensions, but anything higher is hard, because the number of elements required is going to go up exponentially with the dimension. What this means is that estimates of C_{ij} obtained like this are either poor, or expensive (or possibly both!).

The standard method for estimating an integral in a high dimensional space involves taking random samples in that space and using the samples to estimate the integral. Remarkably, the accuracy of this method, properly implemented, does not depend on the dimension of the space. This means that the method is very widely used for graphics problems that involve integrals (pretty much all of rendering). We shall see the method again in this chapter (section **??**) and chapter **??** describes a variety of rendering strategies vested in the method.

We have a domain D and a function f. Assume we have a probability distribution on D. We will assume that this distribution can be represented by a probability density function $p(\boldsymbol{x})$ (this isn't essential, but by doing so we avoid having to do finicky analysis). Now assume that we have a set of N points $\boldsymbol{x}_m$, which are independent samples from $p(\boldsymbol{x})$. We write this property

$$\boldsymbol{x}_m \; p(\boldsymbol{x})$$

The weak law of large numbers guarantees that

$$\frac{1}{N}\sum_m f(\boldsymbol{x}_m) \to \int_D f(\boldsymbol{x})p(\boldsymbol{x})dx$$

as N gets large. A formal statement of this property is that the probability that the sum is different from the integral by more than a small amount gets very small as N gets sufficiently large; for details, see (). Another, very useful, form of this expression is

$$\frac{1}{N}\sum_m \frac{f(\boldsymbol{x}_m)}{p(\boldsymbol{x})} \to \int_D f(\boldsymbol{x})dx$$

We can use this observation to estimate S_{ij}. The domain for this integral is $D_i \times D_j$. For the moment, let us use a uniform distribution for $p(\boldsymbol{x})$; samples from this distribution consist of pairs of points $(\boldsymbol{x}_{i,m}, \boldsymbol{x}_{j,m})$ where $\boldsymbol{x}_{i,m}$ is uniformly distributed in D_i and $\boldsymbol{x}_{j,m}$ is uniformly distributed in D_j. If we write the area of D_i as A_i, then

$$p(\boldsymbol{x}) = \frac{1}{A_i A_j}$$

This means that we have

$$\begin{aligned} S_{ij} &= \int_{D_i} \rho(\boldsymbol{x}) \int_{D_j} \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} dA_{\boldsymbol{x}} \\ &\approx \frac{A_i A_j}{N} \sum_m \rho(\boldsymbol{x}_{i,m}) \Phi(\boldsymbol{x}_{i,m}, \boldsymbol{x}_{j,m}) \end{aligned}$$

This is equivalent to casting a set of random rays between D_i and D_j and evaluating the transfer down each ray. The estimate incorporates the effect of occlusion because Φ is zero for occluded rays. Notice that there are errors in this estimate — we might cast all our rays through the holes in the fencing mask again — but that one needs to be "unlucky" to get reliably bad estimates — or, equivalently, there is a very low probability that we will be able to get randomly constructed rays to go through the holes in the fencing mask.

The Variance of Estimates.

An important property of this integration process is that estimating S_{ij} twice will yield two different estimates, unless we use the same set of random numbers each time. This property is important for two reasons. Firstly, it means that we will not obtain large, patterned errors in our estimates of S_{ij}. The fencing mask is a good example here, too. Assume we wish to render a scene with two small luminaires inside a fencing mask. If we compute the S_{ij} terms linking the luminaires with the outside world using a fixed grid of samples (as in section **??**) on each luminaire, there are several nasty possibilities. One poor choice of grid will have every ray leaving one luminaire strike the mask, and every ray leaving the other pass through the holes. In this case, we exaggerate the effect of one luminaire and underestimate the effect of the other. This is a particularly nasty example of **aliasing**. It has the typical property of aliasing, that high frequencies (the holes in the grid) end up looking like low frequencies (the constant zero term). In particular, estimates of S_{ij} for *all* elements on a luminaire are poor simultaneously.

If, instead, we choose rays randomly, then estimates of S_{ij} for some elements on each luminaire might be poor, but the average estimate is good. This is because it is relatively easy to build a large fixed grid with the "unlucky" property that every ray passes through a hole in the mask, but it is very unlikely that a series of small random grids (one for each element) will have this property. In particular, instead of having high frequencies look like low frequencies, each frequency will have a small error, because each estimate has a small but not patterned error.

Secondly, it means that we can talk about the **variance** of the estimate of S_{ij}. The estimate is itself a random variable (because it is a weighted sum of N random variables) and so has a variance. It turns out that this variance is

independent of the dimension over which we're integrating — which is good news — but behaves assymptotically like $1/N$ — which is bad news. However, we can use various strategies to change the constant in this assymptotic behaviour; some are described below, and more in section **??**.

Pseudo-random Estimates.

A remarkable fact about random points uniformly distributed within some plane region is that they don't look particularly random (figure **??**). In particular, one sees "clumps" of points. This phenomenon is usually understood by thinking of a set of points as the result of a **point process** (which is something that produces a set of random points). In the case of uniformly distributed points — otherwise known as a **Poisson process** — points are put down independent of the position of the other points. This means that some points can get very close together. The distribution of points is uniform if there are enough points, but for practical point sets we expect some pairs to be quite close.

`more will come here: DAF`

1.4 SOLVING THE LINEAR SYSTEM

For the moment, assume that we possess the linear system of section **??** (we have deferred the details of computing form factors, etc. to section **??**). Gauss elimination — the standard method for solving some linear systems — is an inappropriate choice of solution algorithm, because our system is very large and very sparse. Gauss elimination doesn't take advantage of the sparseness, and takes time cubic in the size of the linear system. It is natural to want to use an iterative method.

1.4.1 Simple Iterative Solutions

Recursive Approximation of the Neumann Series.

Whether we do collocation or a Galerkin method (either radiosity or radiant power), we get a system that looks like

$$(\mathcal{I} - \mathcal{K})\boldsymbol{B} = \boldsymbol{E}$$

(where we suppress the superscripts for the moment.) Using the reasoning of section **??**, we have that

$$\boldsymbol{B} = (I + \sum_{i=1}^{\infty} \mathcal{K}^i)\boldsymbol{E}$$

Now we could define an iterative solution by

$$\boldsymbol{B}^{(0)} = \boldsymbol{E}$$

$$\boldsymbol{B}^{(k)} = \boldsymbol{E} + \mathcal{K}\boldsymbol{B}^{(k-1)}$$

It is easily verified that

$$\boldsymbol{B}^{(k)} = (I + \sum_{i=1}^{k} \mathcal{K}^i)\boldsymbol{E}$$

We need to verify that this iteration is going to converge; doing so involves a little thrashing around in linear algebra which is confined to appendix **??**.

In fact, the algorithm is not terribly practical in the current form for two reasons. Firstly, for each iteration we need to multiply by $\mathcal{K}$. For most problems we deal with, this is a very big matrix indeed, and the multiplication is expensive. Secondly, we need to store the whole of $\mathcal{K}$ — or recompute it on each iteration — and this is likely to be difficult for a very big problem. In fact, if we adopt a compressed representation of $\mathcal{K}$, both of these problems can be overcome, and the algorithm becomes practical (section **??**).

Jacobi and Gauss-Seidel Iteration.

Our matrix equation can be written as $(\mathcal{I} - \mathcal{K})\boldsymbol{B} = \boldsymbol{E}$. Repeated evaluation of the Neumann series is equivalent to interpreting this equation as

$$\boldsymbol{B}^{(n+1)} = \boldsymbol{E} + \mathcal{K}\boldsymbol{B}^{(n)}$$

where the superscripts indicate the iteration. We have shown that this iteration converges. We can be more general, by writing

$$\mathcal{K} = \mathcal{D} + \mathcal{L} + \mathcal{U}$$

where $\mathcal{D}$ is diagonal, $\mathcal{L}$ contains only elements below the diagonal (all the rest are zero) and $\mathcal{U}$ contains only elements above the diagonal. In this case, the iterative method given by

$$(\mathcal{I} - \mathcal{D})\boldsymbol{B}^{(n+1)} = \boldsymbol{E} + (\mathcal{L} + \mathcal{U})\boldsymbol{B}^{(n)}$$

is known as **Jacobi iteration**. The equations are easy to solve, because $(\mathcal{I} - \mathcal{D})$ is diagonal.

In practice, this iteration involves sweeping through the elements of $\boldsymbol{B}$, computing new values for each element. But once we have a new value, we could use it. This gives

$$(\mathcal{I} - \mathcal{D})\boldsymbol{B}^{(n+1)} = \boldsymbol{E} + \mathcal{L}\boldsymbol{B}^{(n+1)} + \mathcal{U}\boldsymbol{B}^{(n)}$$

(assuming that we are sweeping from the first to the last element). This works, because we need elements of $\boldsymbol{B}^{(n+1)}$ on the right hand side only after we have computed their value on the left. This strategy is known as **Gauss-Seidel** iteration.

1.4.2 Southwell Iteration

Neither Gauss-Seidel nor Jacobi iteration are methods of choice for radiosity problems because one needs to multiply by the whole of $\mathcal{K}$ at each step, which is expensive. However, they give us a hint of a way to proceed. Recall that the residual is $\boldsymbol{R} = (\mathcal{I} - \mathcal{K})\boldsymbol{B} - \boldsymbol{E}$. In Jacobi iteration, we are visiting each component of the residual in turn and zeroing that component by adjusting the corresponding component of $\boldsymbol{B}$ (of course, the value doesn't stay zero because the other adjustments change it; that's why we iterate).

A small change to Jacobi iteration yields a really powerful method, usually known as **Southwell iteration**. Instead of visiting each element of the residual in turn and zeroing it, we go to the largest value of the residual and zero it by

modifying the corresponding component of $\boldsymbol{B}$. In particular, instead of visiting every component of the residual in turn, we will visit the largest at each step — this means that we may visit some components of $\boldsymbol{B}$ many times before we adjust others.

Assume we have an estimate $\boldsymbol{B}^{(m)}$. We form the residual for this estimate, $\boldsymbol{R}^{(m)} = (\mathcal{I} - \mathcal{K})\boldsymbol{B}^{(m)} - \boldsymbol{E}$. Assume that the i'th component of the residual has the largest value. We obtain a new estimate of b_i by adjusting the value so that the i'th component of the residual is zero. This is most easily written in terms of components (rather than matrices and vectors), where we get

$$b_i^{(m+1)} = \frac{1}{1 - k_{ii}} \left(e_i + \sum_{j, j \neq i} k_{ij} b_j^{(m)} \right)$$

Now we have that $r_i^{(m)} = b_i^{(m)} - \sum_j k_{ij} b_j^{(m)} - e_i$, which means that

$$\left(e_i + \sum_{j, j \neq i} k_{ij} b_j^{(m)} \right) = b_i^{(m)} - r_i^{(m)} - k_{ii} b_i^{(m)}$$

Substituting this back in to the update equation, we get

$$b_i^{(m+1)} = b_i^{(m)} - \frac{r_i^{(m)}}{1 - k_{ii}}$$

Now we know how to update the radiosity, but what has happened to the residual? We can do this in terms of matrices and vectors, to get

$$\begin{aligned} \boldsymbol{R}^{(m+1)} &= (\mathcal{I} - \mathcal{K})\boldsymbol{B}^{(m+1)} - \boldsymbol{E} \\ &= \boldsymbol{R}^{(m)} + (\mathcal{I} - \mathcal{K})(\boldsymbol{B}^{(m+1)} - \boldsymbol{B}^{(m)}) \end{aligned}$$

Only one entry in the vector $\boldsymbol{B}$ has changed, so we expect $(\boldsymbol{B}^{(m+1)} - \boldsymbol{B}^{(m)})$ to be zero in all but one component. This will be the i'th component. We can use the notation $\boldsymbol{e}_i$ for the vector which has all components zero, except the i'th, which is 1.

$$(\boldsymbol{B}^{(m+1)} - \boldsymbol{B}^{(m)}) = -(\frac{r_i^{(m)}}{1 - k_{ii}})\boldsymbol{e}_i$$

Now the new residual becomes

$$\boldsymbol{R}^{(m+1)} = \boldsymbol{R}^{(m)} - (\frac{r_i^{(m)}}{1 - k_{ii}})(\mathcal{I} - \mathcal{K})\boldsymbol{e}_i$$

We now have an algorithm, given in algorithm **??**. There are three steps at each iteration:

1. Find the residual element with largest magnitude.

2. Update the corresponding element of the radiosity.
3. Update the residual.

The first step is linear in the number of elements; the second is trivial; and the third has the crucial feature is that each update involves only one column of $\mathcal{K}$

Southwell iteration can be shown to converge only in the case of a Galerkin method where one works in terms of radiant power.

Progressive Radiosity: A Qualitative Physical Interpretation of Southwell Iteration.

Southwell iteration is usually referred to as **progressive radiosity**, because it allows an attractive physical interpretation. The exact details of the interpretation depend slightly on whether one has used a collocation method, a Galerkin method, or some other method to form the finite-dimensional linear system. We shall avoid these quibbles, and sketch a general form.

Assume that the basis consists of functions that have local support (as opposed to, say, sinusoids). Then we can interpret the components of the vector $\boldsymbol{B}^{(m)}$ as giving the radiosity at each element. Now the residual vector, $\boldsymbol{R}^{(m)} = (\mathcal{K}\boldsymbol{B}^{(m)} + \boldsymbol{E}) - \boldsymbol{B}^{(m)}$ (we changed signs here — this doesn't matter because the residual just contains errors), can be intepreted as

(current estimate of radiosity arriving plus exitance)−(current estimate of radiosity)

or, equivalently, radiosity that has arrived from other elements but hasn't yet been accounted for in computing the value of this element. This is sometimes called **unshot radiosity** or **unshot energy**, depending upon what method is used to construct the linear system.

Now each iteration of the algorithm becomes:

1. Find the element with the largest unshot radiosity.
2. Update its radiosity to take this into account.
3. Use this radiosity to update the unshot radiosity (the residual) at every other element.

This interpretation gives some insight into why the method is efficient and popular.

Acceleration using Ambient Illumination.

1.5 HIERARCHICAL RADIOSITY AND ITERATIVE METHODS

There are other iterative methods that one can use to solve linear systems, the method of choice currently being the conjugate gradient method. These methods share a difficulty that Southwell iteration avoids: the need to compute and store the whole of $\mathcal{K}$. This can present significant difficulties, because there might be millions of elements. However, the physics of radiation transfer mean that we can adopt very highly compressed representations of $\mathcal{K}$, which could be small, and could allow extremely fast multiplications.

The underlying reason that this is possible is a relatively simple phenomenon; the significance of a radiator to a particular receiver depends on the solid angle it subtends *at that receiver*. This means that very large radiators may have very small effects at a particular receiver. Multiplying by $\boldsymbol{K}$ involves computing the effects at each receiver of a particular pattern of radiation. Now if a set of radiators taken together still subtend a small solid angle at a particular receiver, there is no particular reason to consider each one individually when computing the effect *on that receiver*. This means that we can cluster radiators.

We can also cluster receivers. Assume that we have some receivers that are near one another and facing in about the same direction. They are looking at a distant radiator. If the receivers are close, and point in about the same way, then each receiver's view of the radiator is about the same, meaning that the radiator has about the same effect on each receiver.

1.5.1 The Simplest Hierarchical Radiosity

We will look at hierarchical radiosity in its simplest form to get a sense of the approach, and then generalize the idea. Assume that we are in a world all of whose elements are quadrilaterals. Each element will be a function that is one on some quadrilateral, and zero elsewhere. We will build this world out of relatively large quadrilaterals at first. Each quadrilateral can be seen as the root of a quad-tree. Each element can be seen as the root of a tree of elements, where its children are one on one child of the quad-tree rooted at the quadrilateral, and zero elsewhere (figure **??**).

Now consider the radiation exchange between two quadrilaterals. It could well be the case that these two quadrilaterals are a good representation for this exchange. For this to be the case, we would want the radiosity on the receiver due to the source to be roughly constant (so the constant element represents it well) and the source to look small enough to the receiver that any variation in radiosity on the source doesn't affect the receiver particularly. We expect this to be possible, because of the smoothing effects of interreflections (recall section **??**). In this case, we should be able to link these two elements.

Another possibility would be that the source is quite well represented by a single element, but its effect on the receiver varies sufficiently over the receiver that we need a finer scale representation; this would argue for splitting the receiver. A third possibility would be that the effect on the receiver may be represented well with a single element, but there is substantial spatial variation in the radiosity on the source; this would argue for splitting the source.

Assume, for the moment, we have an oracle that can tell us whether elements can be linked. We now have the core of an algorithm for building links, though much of the details are missing. For each pair of roots, we call algorithm **??**.

We need to associate two variables with each element in the hierarchy dependent from each root. Firstly, each element gets its radiosity. Secondly, each element gets its exitance. Now let us focus on a particular receiver (root). Our representation of radiosity on this receiver will be given by the values at the leaves of the hierarchy — the other elements are there for convenience. This receiver wishes to gather radiosity from some source. There will be a series of links between source

elements and receiver elements. Each of these links will gather radiosity. We need to inform the leaves of the hierarchy about radiosity that has arrived at elements higher up in the hierarchy. Once this is done, we need to adjust the elements higher up in the hierarchy so they correctly represent the leaves.

Assume that radiosity arrives at some element. Each of its children must then also receive radiosity; in particular, because radiosity is radiant power per unit area, each child must receive the same amount of radiosity the parent obtained. We can write this as

$$B_{\text{incoming, child}} = B_{\text{gathered at child}} + B_{\text{incoming, parent}}$$

This means that we can **push** radiosity down the hierarchy. We start at the root, and add to the value arriving at each of its children any radiosity that has arrived at the root. We now recur; each of the children take the radiosity that arrived, and add it to the radiosity that arrived at all of its children.

We now have the correct value of gathered radiosity at the leaves. We must ensure that the rest of the hierarchy represents the leaves correctly. Because radiosity is power per unit area, we have

$$B_{\text{parent}} = \frac{1}{A_{\text{parent}}} \left(\sum_{\text{children}} B_{\text{child}} A_{\text{child}} \right)$$

This gives an algorithm of the following form:

- **Gather:** each element in the receiver's hierarchy gathers radiosity along its own links.
- **Push:** each element informs its children (recursively, down the tree) how much radiosity has arrived.
- **Pull:** each element informs its parent (recursively, up the tree) how much radiosity has arrived.

All this has been for a single receiver. One way to build a radiosity algorithm out of this would be for each receiver to gather in turn, using this algorithm. As we shall see, this corresponds rather precisely to either Jacobi iteration, or Gauss-Seidel iteration, depending on how we set up the details.

1.5.2 Hierarchical Radiosity as Linear Algebra

As we said, the underlying representation of radiosity on each element is the representation at the leaves of the hierarchy. All the rest is for efficiency. In particular, for each root, we know linear expressions giving the radiosity at each element of the hierarchy in terms of the leaves. Some — but not all — of these elements are linked to other elements in other hierarchies. What happens when a particular root gathers from all other roots?

We can arrange the radiosity values at each leaf in the whole system into a vector, $\boldsymbol{B}^{(leaves)}$. When we gather at a particular root, we are computing an approximation to some components of

$$\mathcal{K}\boldsymbol{B}^{(leaves)}$$

(in particular, the components that represent the leaves associated with our root). We are doing this by:

- Computing a representation of the radiosity at linked elements of the hierarchy (this is the pull stage). This representation is a linear function of the radiosity at the leaves.
- Transferring radiosity down links (the gather stage). Generally, there are significantly fewer links than there are pairs of leaves, because the links are (hopefully) often between elements higher up the hierarchy. This is also a linear function, but now is a function of the radiosity at the linked elements.
- Taking the gathered radiosity and using it to update the leaves (the push stage). This is also a linear function of the radiosity gathered at the elements.

All this means that we are approximating $\mathcal{K}$ by

$$\mathcal{P}_{\text{ush}}\mathcal{K}_{\text{reduced}}\mathcal{P}_{\text{ull}}$$

If we iterate the gather-push-pull algorithms as given, then this corresponds to iterating the Neumann series, as in section **??**. Depending on whether we use new values immediately, or store them and update at the end of an iteration, we are engaged in either Gauss-Seidel or Jacobi iteration, *using an approximation to* $\mathcal{K}$ (check section **??** if you're not sure).

Of course, once it is clear that what we're doing is approximating multiplication by the kernel matrix, other possibilities become available — one might wish to engage in Southwell iteration, for example.

1.5.3 The Oracle

Whether we have a good or a bad approximation depends rather specifically on the oracle we use to determine whether elements should be linked. There are three major kinds of oracle; we will discuss two here, and defer the third until after section **??**.

Linking on the Form Factor.

The first possible oracle is to estimate the form-factor between the two elements. If this is low, then the elements are linked. It would be unwise to use an accurate estimate of the form-factor, because this is expensive to compute (section **??**). Instead, it is usual to use an approximation based on an element to point form-factor for a disk.

It is an exercise to show that, for a point $\boldsymbol{x}$ on the axis of a circular disk D with a normal pointing directly to the center of the disk (figure **??**), we have

$$\int_D \Phi(\boldsymbol{x}, \boldsymbol{u}) dA_{\boldsymbol{u}} = \frac{R^2}{R^2 + d^2}$$

This gives a rough-and-ready estimate for a form factor between two elements. Choose a point in the center of each element, labelled $\boldsymbol{p}$; choose an approximating

disk for one; now we approximate with

$$\frac{1}{A_i}\int_{D_i}\int_{D_j}\Phi(\boldsymbol{x},\boldsymbol{u})dA_{\boldsymbol{u}}dA_{\boldsymbol{x}} \approx \frac{R^2}{R^2+d^2}\cos\theta_{\boldsymbol{p}_i}\cos\theta_{\boldsymbol{p}_j}$$

(the notation is given in figure **??**). It is not a good idea to use this expression to compute actual links, but it does work as an oracle. Of course, other approximations may be used as well.

Linking on Radiosity Transferred.

Linking on the form factor essentially corresponds to constructing an approximation to the matrix $\mathcal{K}$ that approximates the matrix well. In fact, we are not going to multiply this matrix by many different vectors, so we could look for an approximation that represents the product $\mathcal{K}\boldsymbol{B}$ well. As stated, this sounds absurd, because we would need to know the answer to the radiosity problem and to the multiplication to tell whether the approximation was good. But there is a way out. We can start by constructing an extremely coarse hierarchy and solving the problem on this coarse hierarchy. If some links carry very little power in this solution, there is no particular need to refine them; otherwise, we refine the hierarchy and solve again on this refined hierarchy. We could do this more than once, too.

Refining the hierarchy is straightforward. When we refine, we are going to be inspecting links between leaves to see if they need to be subdivided further. This means we need to know an estimate of the radiosity transferred down the link — which is an estimate of the component of $\mathcal{K}\boldsymbol{B}$ represented by the receiving element. If this estimate is large, then our representation of $\mathcal{K}\boldsymbol{B}$ is probably poor, as the value is unlikely to be both large and constant over the element, so we need to split. We assume that radiosity is constant over leaves of the coarse scale solution, and get an approximation to the radiosity travelling down the link by multiplying the radiosity at the element by an estimate of the form-factor.

We are using the coarse scale solution as an oracle; we can actually use it as a start point, too. Typically, one computes a solution on a coarse hierarchy (established using a form-factor linking strategy) and then uses this solution as an oracle to establish a hierarchy for a slightly finer solution. We then transfer the coarser scale solution to the finer hierarchy. The transfer is easy; elements in the fine hierarchy corresponding to elements of the coarse hierarchy simply get the value of the corresponding elements, and elements that don't get the value of the first ancestor that does (figure **??**).

1.6 IMPORTANCE

1.7 RENDERING A SOLUTION WITH A FINAL GATHER

1.8 DISCONTINUITIES, MOTION AND MESHES