7 Algebraic Reconstruction Algorithms

An entirely different approach for tomographic imaging consists of assuming that the cross section consists of an array of unknowns, and then setting up algebraic equations for the unknowns in terms of the measured projection data. Although conceptually this approach is much simpler than the transform-based methods discussed in previous sections, for medical applications it lacks the accuracy and the speed of implementation. However, there are situations where it is not possible to measure a large number of projections, or the projections are not uniformly distributed over 180 or 360°, both these conditions being necessary requirements for the transform-based techniques to produce results with the accuracy desired in medical imaging. An example of such a situation is earth resources imaging using cross-borehole measurements discussed in Chapter 4. Problems of this type are sometimes more amenable to solution by algebraic techniques. Algebraic techniques are also useful when the energy propagation paths between the source and receiver positions are subject to ray bending on account of refraction, or when the energy propagation undergoes attenuation along ray paths as in emission CT. [Unfortunately, many imaging problems where refraction is encountered also suffer from diffraction effects (see Chap. 4).] As will be obvious from the discussion to follow, in algebraic methods it is essential to know ray paths that connect the corresponding transmitter and receiver positions. When refraction and diffraction effects are substantial (medium inhomogeneities exceed 10% of the average background value and the correlation length of these inhomogeneities is comparable to a wavelength), it becomes impossible to predict these ray paths. If algebraic techniques are applied under these conditions, we often obtain meaningless results.

If the refraction and diffraction effects are small (medium inhomogeneities are less than 2 to 3% of the average background value and the correlation width of these inhomogeneities is much greater than a wavelength), in some cases it is possible to combine algebraic techniques with digital ray tracing techniques [And82], [And84a], [And84b] and devise iterative procedures in which we first construct an image ignoring refraction, then trace rays connecting the corresponding transmitter and receiver locations through this distribution, and finally use these rays to construct a more accurate set of

algebraic equations. Experimental verification of this iterative procedure for weakly refracting objects has been obtained [And84b].

Space limitations prevent us from discussing here the combined ray tracing and algebraic reconstruction algorithms. Our aim in this section is to merely introduce the reader to the algebraic approach for image reconstruction. First we will show how we may construct a set of linear equations whose unknowns are elements of the object cross section. The Kaczmarz method for solving these equations will then be presented. This will be followed by the various approximations that are used in this method to speed up its computer implementation.

7.1 Image and Projection Representation

In Fig. 7.1 we have superimposed a square grid on the image $f(x, y)$; we will assume that in each cell the function $f(x, y)$ is constant. Let f_j denote this constant value in the jth cell, and let N be the total number of cells. For algebraic techniques a ray is defined somewhat differently. A ray is now a "fat" line running through the (x, y)-plane. To illustrate this we have shaded the ith ray in Fig. 7.1, where each ray is of width τ. In most cases the ray width is approximately equal to the image cell width. A line integral will now be called a *ray-sum*.

Like the image, the projections will also be given a one-index representa-

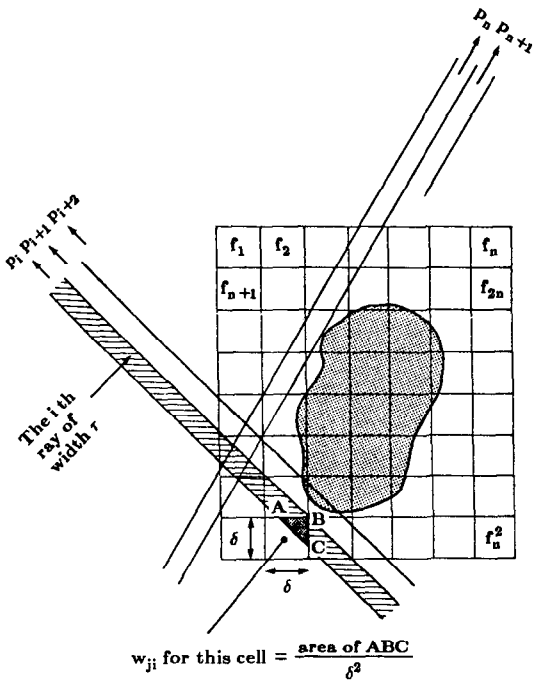

Fig. 7.1: *In algebraic methods a square grid is superimposed over the unknown image. Image values are assumed to be constant within each cell of the grid. (From [Ros82].)*

tion. Let p_i be the ray-sum measured with the ith ray as shown in Fig. 7.1. The relationship between the f_j's and p_i's may be expressed as

$$\sum_{j=1}^{N} w_{ij} f_j = p_i, \qquad i = 1, 2, \cdots, M \tag{1}$$

where M is the total number of rays (in all the projections) and w_{ij} is the weighting factor that represents the contribution of the jth cell to the ith ray integral. The factor w_{ij} is equal to the fractional area of the jth image cell intercepted by the ith ray as shown for one of the cells in Fig. 7.1. Note that most of the w_{ij}'s are zero since only a small number of cells contribute to any given ray-sum.

If M and N were small, we could use conventional matrix theory methods to invert the system of equations in (1). However, in practice N may be as large as 65,000 (for 256 × 256 images), and, in most cases for images of this size, M will also have the same magnitude. For these values of M and N the size of the matrix $[w_{ij}]$ in (1) is 65,000 × 65,000 which precludes any possibility of direct matrix inversion. Of course, when noise is present in the measurement data and when $M < N$, even for small N it is not possible to use direct matrix inversion, and some least squares method may have to be used. When both M and N are large, such methods are also computationally impractical.

For large values of M and N there exist very attractive iterative methods for solving (1). These are based on the "method of projections" as first proposed by Kaczmarz [Kac37], and later elucidated further by Tanabe [Tan71]. To explain the computational steps involved in these methods, we first write (1) in an expanded form:

$$\begin{aligned}
&w_{11} f_1 + w_{12} f_2 + w_{13} f_3 + \cdots + w_{1N} f_N = p_1 \\
&w_{21} f_1 + w_{22} f_2 + \qquad + \cdots + w_{2N} f_N = p_2 \\
&\vdots \\
&w_{M1} f_1 + w_{M2} f_2 + \qquad + \cdots + w_{MN} f_N = p_M .
\end{aligned} \tag{2}$$

A grid representation with N cells gives an image N degrees of freedom. Therefore, an image, represented by $(f_1, f_2, \cdots, f_N)$, may be considered to be a single point in an N-dimensional space. In this space each of the above equations represents a hyperplane. When a unique solution to these equations exists, the intersection of all these hyperplanes is a single point giving that solution. This concept is further illustrated in Fig. 7.2 where, for the purpose of display, we have considered the case of only two variables f_1 and f_2 satisfying the following equations:

$$\begin{aligned}
w_{11} f_1 + w_{12} f_2 &= p_1 \\
w_{21} f_1 + w_{22} f_2 &= p_2 .
\end{aligned} \tag{3}$$

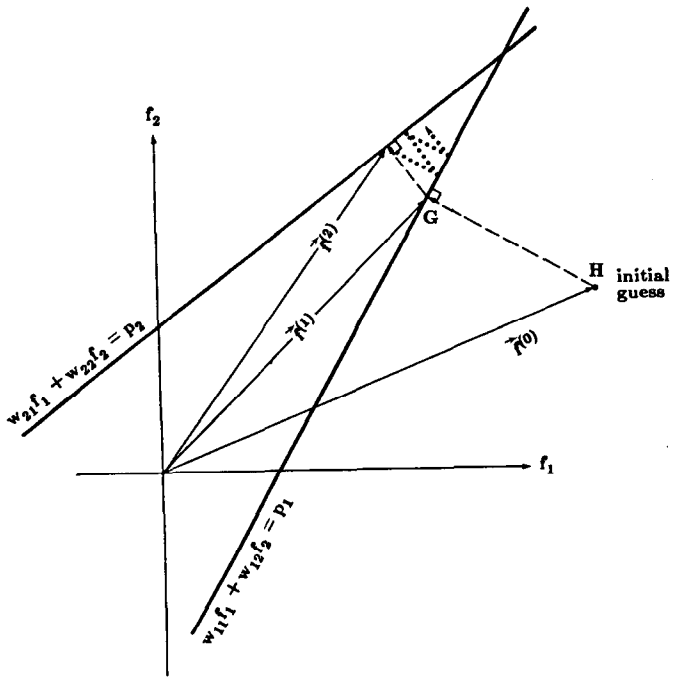

Fig. 7.2: *The Kaczmarz method of solving algebraic equations is illustrated for the case of two unknowns. One starts with some arbitrary initial guess and then projects onto the line corresponding to the first equation. The resulting point is now projected onto the line representing the second equation. If there are only two equations, this process is continued back and forth, as illustrated by the dots in the figure, until convergence is achieved. (From [Ros82].)*

The computational procedure for locating the solution in Fig. 7.2 consists of first starting with an initial guess, projecting this initial guess on the first line, reprojecting the resulting point on the second line, and then projecting back onto the first line, and so forth. If a unique solution exists, the iterations will always converge to that point.

For the computer implementation of this method, we first make an initial guess at the solution. This guess, denoted by $f_1^{(0)}, f_2^{(0)}, \cdots, f_N^{(0)}$, is represented vectorially by $\vec{f}^{(0)}$ in the N-dimensional space. In most cases, we simply assign a value of zero to all the f_i's. This initial guess is projected on the hyperplane represented by the first equation in (2) giving $\vec{f}^{(1)}$, as illustrated in Fig. 7.2 for the two-dimensional case. $\vec{f}^{(1)}$ is projected on the hyperplane represented by the second equation in (2) to yield $\vec{f}^{(2)}$ and so on. When $\vec{f}^{(i-1)}$ is projected on the hyperplane represented by the ith equation to yield $\vec{f}^{(i)}$, the process can be mathematically described by

$$\vec{f}^{(i)} = \vec{f}^{(i-1)} - \frac{(\vec{f}^{(i-1)} \cdot \vec{w}_i - p_i)}{\vec{w}_i \cdot \vec{w}_i} \vec{w}_i \tag{4}$$

where $\vec{w}_i = (w_{i1}, w_{i2}, \cdots, w_{iN})$, and $\vec{w}_i \cdot \vec{w}_i$ is the dot product of $\vec{w}_i$ with itself. To see how (4) comes about we first write the first equation of (2) as

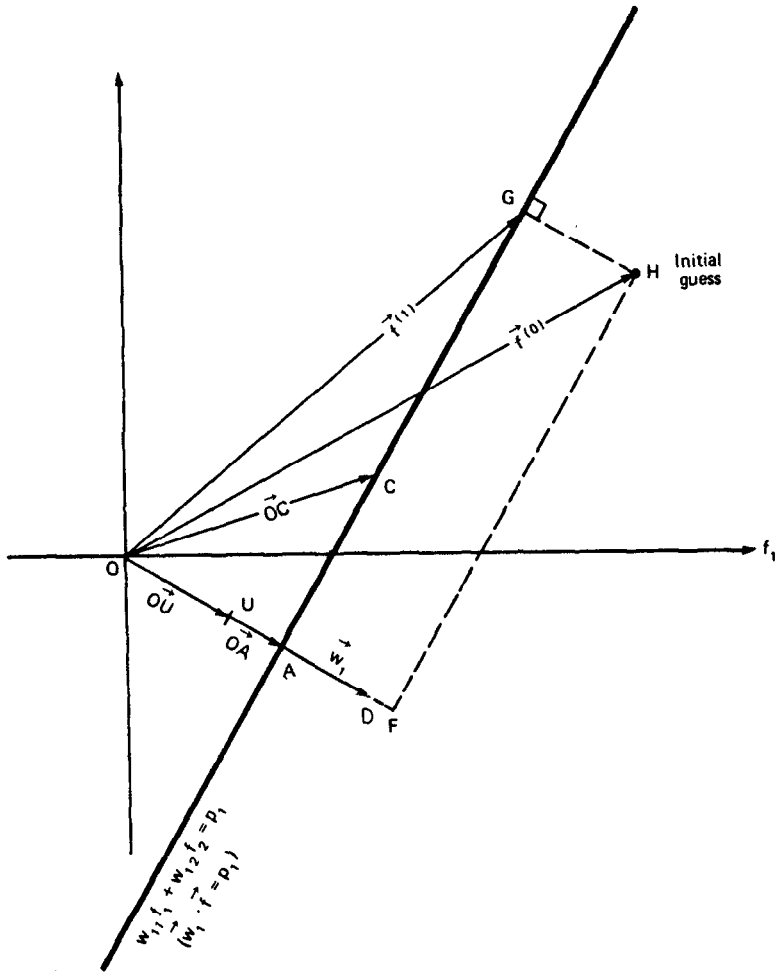

Fig. 7.3: *The hyperplane $\vec{w}_1 \cdot \vec{f} = p_1$ (represented by a line in this two-dimensional figure) is perpendicular to the vector $\vec{w}_1$. (From [Ros82].)*

follows:

$$\vec{w}_1 \cdot \vec{f} = p_1. \tag{5}$$

The hyperplane represented by this equation is perpendicular to the vector $\vec{w}_1$. This is illustrated in Fig. 7.3, where the vector $\overrightarrow{OD}$ represents $\vec{w}_1$. This equation simply says that the projection of a vector $\overrightarrow{OC}$ (for any point C on the hyperplane) on the vector $\vec{w}_1$ is of constant length. The unit vector $\overrightarrow{OU}$ along $\vec{w}_1$ is given by

$$\overrightarrow{OU} = \frac{\vec{w}_1}{\sqrt{\vec{w}_1 \cdot \vec{w}_1}} \tag{6}$$

and the perpendicular distance of the hyperplane from the origin, which is

equal to the length of $\overrightarrow{OA}$ in Fig. 7.3, is given by $\overrightarrow{OC} \cdot \overrightarrow{OU}$:

$$|\overrightarrow{OA}| = \overrightarrow{OU} \cdot \overrightarrow{OC} = \frac{1}{\sqrt{\vec{w}_1 \cdot \vec{w}_1}} (\vec{w}_1 \cdot \overrightarrow{OC})$$

$$= \frac{1}{\sqrt{\vec{w}_1 \cdot \vec{w}_1}} (\vec{w}_1 \cdot \vec{f}) = \frac{p_1}{\sqrt{\vec{w}_1 \cdot \vec{w}_1}} . \tag{7}$$

Now to get $\vec{f}^{(1)}$ we have to subtract from $\vec{f}^{(0)}$ the vector $\overrightarrow{HG}$:

$$\vec{f}^{(1)} = \vec{f}^{(0)} - \overrightarrow{HG} \tag{8}$$

where the length of the vector $\overrightarrow{HG}$ is given by

$$|\overrightarrow{HG}| = |\overrightarrow{OF}| - |\overrightarrow{OA}|$$

$$= \vec{f}^{(0)} \cdot \overrightarrow{OU} - |\overrightarrow{OA}|. \tag{9}$$

Substituting (6) and (7) in this equation, we get

$$|\overrightarrow{HG}| = \frac{\vec{f}^{(0)} \cdot \vec{w}_1 - p_1}{\sqrt{\vec{w}_1 \cdot \vec{w}_1}} . \tag{10}$$

Since the direction of $\overrightarrow{HG}$ is the same as that of the unit vector $\overrightarrow{OU}$, we can write

$$\overrightarrow{HG} = |\overrightarrow{HG}| \overrightarrow{OU} = \frac{\vec{f}^{(0)} \cdot \vec{w}_1 - p_1}{\vec{w}_1 \cdot \vec{w}_1} \vec{w}_1. \tag{11}$$

Substituting (11) in (8), we get (4).

As mentioned before, the computational procedure for algebraic reconstruction consists of starting with an initial guess for the solution, taking successive projections on the hyperplanes represented by the equations in (2), eventually yielding $\vec{f}^{(M)}$. In the next iteration, $\vec{f}^{(M)}$ is projected on the hyperplane represented by the first equation in (2), and then successively onto the rest of the hyperplanes in (2), to yield $\vec{f}^{(2M)}$, and so on. Tanabe [Tan71] has shown that if there exists a unique solution $\vec{f}_s$ to the system of equations (2), then

$$\lim_{k \to \infty} \vec{f}^{(kM)} = \vec{f}_s. \tag{12}$$

A few comments about the convergence of the algorithm are in order here. If in Fig. 7.2 the two hyperplanes are perpendicular to each other, the reader may easily show that given for an initial guess any point in the (f_1, f_2)-plane, it is possible to arrive at the correct solution in only two steps like (4). On the other hand, if the two hyperplanes have only a very small angle between them, k in (12) may acquire a large value (depending upon the initial guess) before the correct solution is reached. Clearly the angles between the

hyperplanes considerably influence the rate of convergence to the solution. If the M hyperplanes in (2) could be made orthogonal with respect to one another, the correct solution would be arrived at with only one pass through the M equations (assuming a unique solution does exist). Although theoretically such orthogonalization is possible using, for example, the Gram–Schmidt procedure, in practice it is computationally not feasible. Full orthogonalization will also tend to enhance the effects of the ever present measurement noise in the final solution. Ramakrishnan *et al.* [Ram79] have suggested a pairwise orthogonalization scheme which is computationally easier to implement and at the same time considerably increases the speed of convergence. A simpler technique, first proposed in [Hou72] and studied in [Sla85], is to carefully choose the order in which the hyperplanes are considered. Since each hyperplane represents a distinct ray integral, it is quite likely that adjacent ray integrals (and thus hyperplanes) will be nearly parallel. By choosing hyperplanes representing widely separated ray integrals, it is possible to improve the rate of convergence of the Kaczmarz approach.

A not uncommon situation in image reconstruction is that of an overdetermined system in the presence of measurement noise. That is, we may have $M > N$ in (2) and $p_1, p_2, \cdots, p_m$ corrupted by noise. No unique solution exists in this case. In Fig. 7.4 we have shown a two-variable system represented by three "noisy" hyperplanes. The broken line represents the course of the solution as we successively implement (4). Now the "solution" doesn't converge to a unique point, but will oscillate in the neighborhood of the intersections of the hyperplanes.

When $M < N$ a unique solution of the set of linear equations in (2) doesn't exist, and, in fact, an infinite number of solutions are possible. For example, suppose we have only the first of the two equations in (3) to use for calculating the two unknowns f_1 and f_2; then the solution can be anywhere on the line corresponding to this equation. Given the initial guess $\vec{f}^{(0)}$ (see Fig. 7.3), the best one could probably do under the circumstances would be to draw a projection from $\vec{f}^{(0)}$ on this line, and call the resulting $\vec{f}^{(1)}$ a solution. Note that the solution obtained in this manner corresponds to that point on the line which is closest to the initial guess. This result has been rigorously proved by Tanabe [Tan71] who has shown that when $M < N$, the iterative approach described above converges to a solution, call it $\vec{f}_s'$, such that $|\vec{f}^{(0)} - \vec{f}_s'|$ is minimized.

Besides its computational efficiency, another attractive feature of the iterative approach presented here is that it is now possible to incorporate into the solution some types of a priori information about the image one is reconstructing. For example, if it is known a priori that the image $f(x, y)$ is nonnegative, then in each of the solutions $\vec{f}^{(k)}$, successively obtained by using (4), one may set the negative components equal to zero. One may similarly incorporate the information that $f(x, y)$ is zero outside a certain area, if this is known.

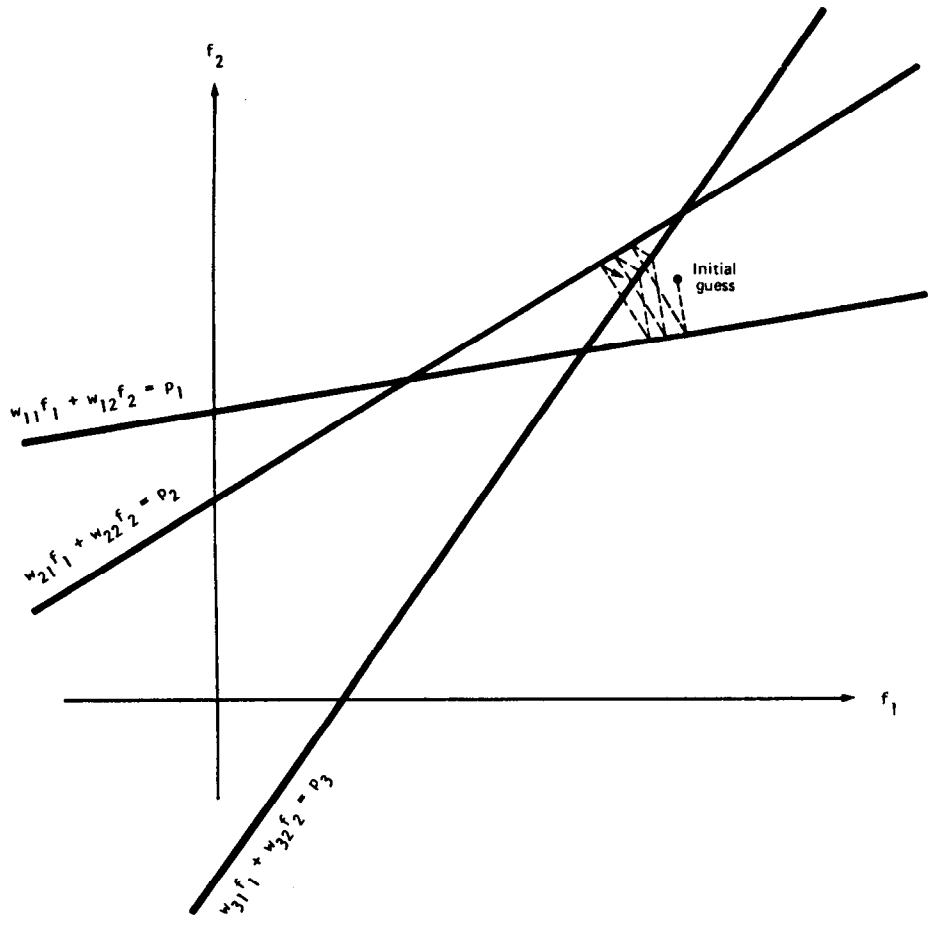

Fig. 7.4: *Illustrated here is the case when the number of equations is greater than the number of unknowns. The lines don't intersect at a single unique point, because the observations p_1, p_2, p_3 have been assumed to be corrupted by noise. No unique solution exists in this case, and the final solution will oscillate in the neighborhood of intersections of the three lines. (From [Ros82].)*

In applications requiring a large number of views and where large-sized reconstructions are made, the difficulty with using (4) can be in the calculation, storage, and fast retrieval of the weight coefficients w_{ij}. Consider the case where we wish to reconstruct an image on a 100×100 grid from 100 projections with 150 rays in each projection. The total number of weights, w_{ij}, needed in this case is 10^8, which is an enormous number and can pose problems in fast storage and retrieval in applications where reconstruction speed is important. This problem is somewhat eased by making approximations, such as considering w_{ij} to be only a function of the perpendicular distance between the center of the ith ray and the center of the jth cell. This perpendicular distance can then be computed at run time.

To get around the implementation difficulties caused by the weight coefficients, a myriad of other algebraic approaches have also been suggested, many of which are approximations to (4). To discuss some of the more implementable approximations, we first recast (4) in a slightly different

form:

$$f_j^{(i)} = f_j^{(i-1)} + \frac{p_i - q_i}{\sum_{k=1}^{N} w_{ik}^2} w_{ij} \tag{13}$$

where

$$q_i = \vec{f}^{(i-1)} \cdot \vec{w}_i \tag{14}$$

$$= \sum_{k=1}^{N} f_k^{(i-1)} w_{ik}. \tag{15}$$

These equations say that when we project the $(i - 1)$th solution onto the ith hyperplane [ith equation in (2)] the gray level of the jth element, whose current value is $f_j^{(i-1)}$, is obtained by correcting its current value by $\Delta f_j^{(i)}$, where

$$\Delta f_j^{(i)} = f_j^{(i)} - f_j^{(i-1)} = \frac{p_i - q_i}{\sum_{k=1}^{N} w_{ik}^2} w_{ij}. \tag{16}$$

Note that while p_i is the measured ray-sum along the ith ray, q_i may be considered to be the computed ray-sum for the same ray based on the $(i - 1)$th solution for the image gray levels. The correction Δf_j to the jth cell is obtained by first calculating the difference between the measured ray-sum and the computed ray-sum, normalizing this difference by $\Sigma_{k=1}^{N} w_{ik}^2$, and then assigning this value to all the image cells in the ith ray, each assignment being weighted by the corresponding w_{ij}.

With the preliminaries presented above, we will now discuss three different computer implementations of algebraic algorithms. These are represented by the acronyms ART, SIRT, and SART.

7.2 ART (Algebraic Reconstruction Techniques)

In many ART implementations the w_{ik}'s in (16) are simply replaced by 1's and 0's, depending upon whether the center of the kth image cell is within the ith ray. This makes the implementation easier because such a decision can easily be made at computer run time. In this case the denominator in (16) is given by $\Sigma_{k=1}^{N} w_{ik}^2 = N_i$ which is the number of image cells whose centers are within the ith ray. The correction to the jth image cell from the ith equation in (2) may now be written as

$$\Delta f_j^{(i)} = \frac{p_i - q_i}{N_i} \tag{17}$$

for all the cells whose centers are within the ith ray. We are essentially smearing back the difference $(p_i - q_i)/N_i$ over these image cells. In (17), q_i's are calculated using the expression in (15), except that one now uses the binary approximation for w_{ik}'s.

The approximation in (17), although easy to implement, often leads to artifacts in the reconstructed images, especially if N_i isn't a good approximation to the denominator. Superior reconstructions may be obtained if (17) is replaced by

$$\Delta f_j^{(i)} = \frac{p_i}{L_i} - \frac{q_i}{N_i} \tag{18}$$

where L_i is the length (normalized by δ, see Fig. 7.1) of the ith ray through the reconstruction region.

ART reconstructions usually suffer from *salt and pepper noise*, which is caused by the inconsistencies introduced in the set of equations by the approximations commonly used for w_{ik}'s. The result is that the computed ray-sums in (15) are usually poor approximations to the corresponding measured ray-sums. The effect of such inconsistencies is exacerbated by the fact that as each equation corresponding to a ray in a projection is taken up, it changes some of the pixels just altered by the preceding equation in the same projection. The SIRT algorithm described briefly below also suffers from these inconsistencies in the forward process [appearing in the computation of q_i's in (16)], but by eliminating the continual and competing pixel update as each new equation is taken up, it results in smoother reconstructions.

It is possible to reduce the effects of this noise in ART reconstructions by relaxation, in which we update a pixel by $\alpha \cdot \Delta f_j^{(i)}$, where α is less than 1. In some cases, the relaxation parameter α is made a function of the iteration number; that is, it becomes progressively smaller with increase in the number of iterations. The resulting improvements in the quality of reconstruction are usually at the expense of convergence.

7.3 SIRT (Simultaneous Iterative Reconstructive Technique)

In this approach, which at the expense of slower convergence usually leads to better looking images than those produced by ART, we again use (17) or (18) to compute the change $\Delta f_j^{(i)}$ in the jth pixel caused by the ith equation in (2). However, the value of the jth cell isn't changed at this time. Before making any changes, we go through all the equations, and then only at the end of each iteration are the cell values changed, the change for each cell being the average value of all the computed changes for that cell. This constitutes one iteration of the algorithm. In the second iteration, we go back to the first equation in (2) and the process is repeated.