that the least-squares approach works for this noise-free problem. And when we use the synthetic data g_f, which better represents the real measured data, then the reconstruction is no longer perfect, but it better resembles the reconstructions that we compute from real data coming from an object that is not pixelated on the given pixel grid. ∎

Exercises

9.1. **A Very Small System**
We consider the CT problem with a 2×2 image and five parallel rays shown in Figure 9.14, where the geometry is scaled such that the length of each ray through one pixel is one.

Show that the corresponding system of linear equations $\boldsymbol{A}\,\boldsymbol{f} = \boldsymbol{b}$ for the line model takes the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \\ 4 \end{pmatrix}. \tag{9.37}$$

Can you guess the solution?

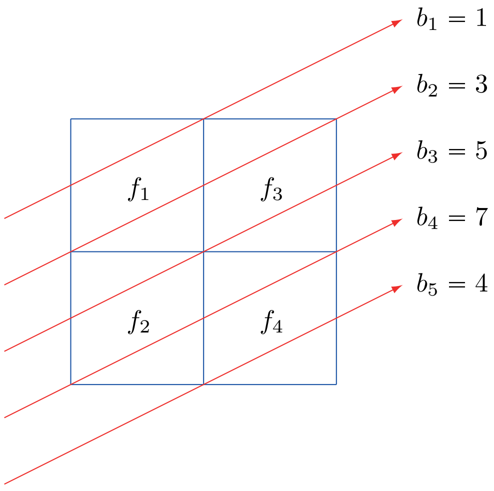

Figure 9.14. *A very small system; the length of each ray through a pixel is one.*

9.2. **Setting Up a System Matrix**
We use the function `paralleltomo` from the MATLAB software package AIR Tools II to generate parallel-beam CT problems. In MATLAB you can use `help paralleltomo` or `doc paralleltomo` to get information about how to use this function.

Write `A = paralleltomo(125,4:4:360)` to generate a 15930×15625 system matrix `A` for a problem with image size $N \times N = 125 \times 125$ and 90 projection angles $4°, 8°, 12°, \ldots, 360°$. The default number of rays (detector elements) is $\texttt{round}(\sqrt{2}N) = 177$. Use `spy(A)` to display the nonzero structure of the matrix. How dense is it?

Let $\boldsymbol{e}_j$ denote the vector of all zeros, except for a one in element j. Convince yourself that the product $\boldsymbol{A}\,\boldsymbol{e}_j$ is the jth column $\boldsymbol{c}_j$ of the system matrix $\boldsymbol{A}$. Hence, $\boldsymbol{c}_j$ is the Radon transform (the sinogram) of an image with a single nonzero pixel. Show the sinograms for several values of j by means of the commands

```
ntheta = 90;                  % Number of angles
p = size(A,1)/ntheta;         % Number of detector elements
imagesc( reshape(A(:,j),p,ntheta) ) % Sinogram
```

You should see discrete sinusoids—the discrete nature is due to the finite size of the image and the detector elements.

Now recall that the ith row $\boldsymbol{r}_i^T$ of $\boldsymbol{A}$ represent a discretization of the line integral that produces the data in the ith detector element; see (9.24). Perform the same type of experiment as above, where you plot several rows of $\boldsymbol{A}$ as 125×125 images. You should see straight lines that correspond to the rays; the jagged appearance is again due to discretization issues.

9.3. **Sinograms of Simple Images**
Figure 9.15 shows the sinograms of two simple geometric objects on a zero background, generated with the same system matrix as above. Can you figure out what these objects are just by looking at the sinograms?

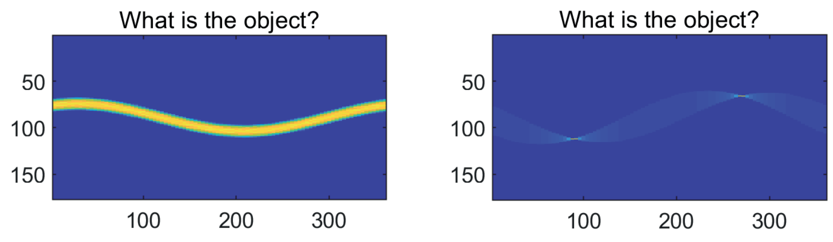

Figure 9.15. *Two sinograms of simple objects.*

Try to generate the corresponding images `X` and show the corresponding sinograms `S = reshape(A*X(:),p,ntheta)`. Can you reproduce the two sinograms?

9.4. **A "Bad Pixel" in the Sinogram**
Assume that we have a single bad pixel in the detector, e.g., a detector element that always has intensity one (due to a fault in the CCD). Such a bad pixel produces a horizontal line in the sinogram, because it is independent of the projection angle.

Let us study the influence of this bad pixel on the reconstruction. Generate a sinogram `S` which is zero everywhere, except for the pixels in a horizontal line, which all have intensity one. Then perform these two experiments with `b = S(:)`.

- Compute the back-projection `A'*b` and show it as an image.
- Compute an FBP reconstruction and show it as an image; use the MATLAB function `iradon` or the function `fbp` from AIR Tools II (or both).

In both cases you should see a distinct structure. This structure will appear as a very annoying artifact when we compute a reconstruction from sinogram data with a bad pixel.

9.5. **Interpretation of Projection Followed by Back-Projection**
In Chapter 6 we introduced the back-projection $\mathcal{R}^\sharp$ (6.1) and explained it as a smearing and summation process. Moreover, we saw that the process of applying a forward-projection to an image f, followed by a back-projection, produces a blurred version of f; see Figure 6.8. This can easily be verified experimentally by means of `radon` and `iradon`.

At this stage, where we have matrices $\boldsymbol{A}$ and $\boldsymbol{A}^T$ that represent the forward- and back-projections, we can illustrate this with an example. Recall that a black image with a single white pixel is represented by a vector $\boldsymbol{f}^{[j]}$ with zero elements except for a single element $x_j = 1$ in position j. What happens when we forward-project this image and then apply back-projection? The result is an image which we can write as

$$\tilde{\boldsymbol{f}}^{[j]} = \boldsymbol{A}^T \boldsymbol{A}\, \boldsymbol{f}^{[j]} = \boldsymbol{A}^T \boldsymbol{c}_j \ . \tag{9.38}$$

This image shows the point spread function for pixel π_j associated with the forward-/back-projection process.

Choose an image size `N` and a vector `theta` of projection angles, generate the system matrix $\boldsymbol{A}$, compute and display $\tilde{\boldsymbol{x}}^{[j]}$ as an image for different choices of j, and comment on the results.