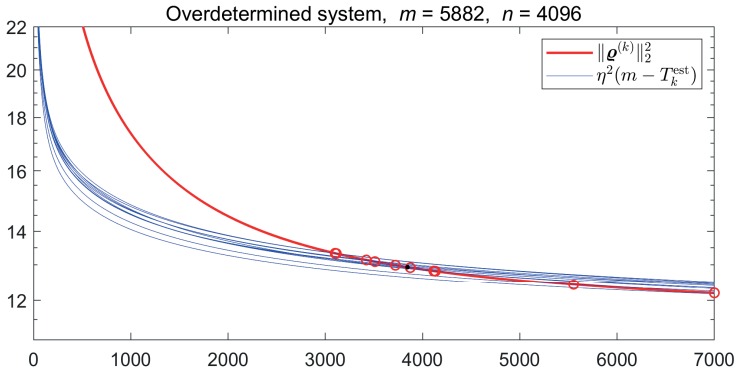

Figure 11.17. *Application of the trace estimate T_k^{est} in the FTNL stopping rule for Landweber's method applied to the overdetermined test problem from Example* 11.11. *We used* 10 *different random vectors $\boldsymbol{w}$ in* (11.33)*, and the corresponding* 10 *intersections between $\|\boldsymbol{\varrho}^{(k)}\|_2^2$ (thick red line) and $\eta^2\,(m - T_k^{\text{est}})$ (thin blue lines) are shown by the red circles. The black dot shows the intersection with the exact $\eta^2\,(m - T_k)$.*

Example 11.13. Continuing from the previous example, Figure 11.17 illustrates the use of the trace estimate T_k^{est} in the FTNL stopping rule. To show the variability of the stopping rule we used 10 different random vectors $\boldsymbol{w}$, leading to 10 different realizations of $\eta^2\,(m - T_k^{\text{est}})$. Their intersections with $\|\boldsymbol{\varrho}^{(k)}\|_2^2$ are shown by the red circles, corresponding to stopping the iterations at

$$k = 3100,\ 3112,\ 3421,\ 3512,\ 3722,\ 3875,\ 4117,\ 4133,\ 5553,\ 7000.$$

The black dot marks the intersection of $\|\boldsymbol{\varrho}^{(k)}\|_2^2$ with the exact $\eta^2\,(m - T_k)$, corresponding to iteration $k = 3846$. ∎

Exercises

For some of the exercises below, recall that we define the relative noise level as

$$\rho = \|\boldsymbol{e}\|_2 / \|\overline{\boldsymbol{b}}\|_2\,, \qquad \overline{\boldsymbol{b}} = \boldsymbol{A}\,\overline{\boldsymbol{x}}\,.$$

Given `rho` and a noise-free right-hand side `bex`, we can generate such a noise vector `e` and the corresponding noisy data `b` by means of the following code:

```
>> e = randn(size(bex));
>> e = rho*norm(bex)*e/norm(e);
>> b = bex + e;
```

11.1. **Return to the Very Small System**
Here we return to the small test problem from Exercise 9.1 with a 2×2 image and five parallel rays. Enter the system into MATLAB:

```
>> A = [1 0 0 0;1 0 1 0; 0 1 1 0; 0 1 0 1; 0 0 0 1]
>> b = [1; 3; 5; 7; 4]
```

Then use the function `kaczmarz` from AIR Tools II to perform 20 iterations with starting vector $\boldsymbol{x}^{(0)} = \mathbf{0}$, which is the default. Note that one "iteration" in AIR Tools always means one sweep over all the rows of the matrix; see (11.3). The iteration vectors are stored as the columns of the 4×20 output matrix `X`:

```
>> kmax = 20;
>> X = kaczmarz(A,b,1:kmax);
```

(The semicolon suppresses printing on the screen.) To see the last iteration vector, write `X(:,kmax)`. Is it correct, i.e., is `A*X(:,kmax) = b`?

11.2. **Convergence Study**
The exact solution to the above problem (the ground truth) is $\overline{\boldsymbol{x}} = (1, 3, 2, 4)^T$. Let us study how fast the iteration vectors of Kaczmarz's method converge to this solution, i.e., in each iteration we look at the largest error:

$$e_k = \|\overline{x} - x^{(k)}\|_2, \qquad k = 1, 2, \ldots, 20\,. \tag{11.34}$$

In MATLAB this computation takes the following form:

```
>> x_exact = [1 3 2 4]';
>> for i=1:kmax, e(i,1) = norm(x_exact - X(:,i)); end
```

Now let us plot the results:

```
>> figure(1)
>> subplot(2,1,1), plot(X')
>> subplot(2,1,2), semilogy(e)
```

On the top plot you should see that the components of the iteration vector $\boldsymbol{x}^{(k)}$ converge to $\overline{\boldsymbol{x}}$, and on the bottom plot you should see that the largest error in each iteration decreases very fast. How many iterations are needed to achieve a maximum error of 10^{-3}?

As a matter of fact, after the first iteration for this problem we have this relation between the maximum errors:

$$e_{k+1} = C \cdot e_k, \qquad k = 1, 2, \ldots, 19\,, \tag{11.35}$$

where C is a constant. What is the value of C?

11.3. **Setting Up a More Realistic Test Problem**
Let us now use the function `paralleltomo` from AIR Tools II to generate a more realistic test problem $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$. The ground truth image is $N \times N$ with $N = 64$, and we use the projection angles $3°, 6°, 9°, \ldots, 180°$. Hence, in MATLAB we write the following:

```
>> N = 64;
>> theta = 3:3:180;
>> [A,b,x] = paralleltomo(N,theta);
```

Note that both `x` and `b` are vectors, because that is how we formulate the problem in linear algebra terminology. How long are the two vectors?

To display these two vectors as images (the object and the sinogram, respectively), use the following MATLAB commands:

```
>> figure(1), imagesc(reshape(x,N,N)), colorbar
>> ntheta = length(theta);
>> p = length(b)/ntheta;  % Number of rays.
>> figure(2), imagesc(theta,1:p,reshape(b,p,ntheta)), colorbar
```

Finally, let us convince ourselves that the system matrix `A` is indeed a very sparse matrix, i.e., it consists mainly of zeros. To do that, use the MATLAB function `spy(A)` to display the nonzeros of the matrix.

11.4. **Using Kaczmarz's Method to Solve the Test Problem**
Now we use Kaczmarz's method to solve the above test problem, and to get a feeling of the convergence let us perform 20 iterations and plot these iterations as images:

```
>> kmax = 20;
>> X = kaczmarz(A,b,1:kmax);
>> figure(3)
>> for i=1:kmax
>>     subplot(4,5,i), imagesc(reshape(X(:,i),N,N))
>> end
```

You should see a very *slow* progression of the reconstructions toward the ground truth image.

Now, try to run `kaczmarz` with 50, 100, 200, 400, 800, and 1600 iterations and compute the corresponding maximum errors. In situations like this where we do not want to store all the iterations, it is possible to specify which iterations will be stored as shown below. To prepare for the computations to take a little while, we use the waitbar option:

```
>> k = [50,100,200,400,800,1600];
```

```
>> options.waitbar = true;
>> X = kaczmarz(A,b,k,[],options);
>> e = zeros(length(k),1);
>> for i=1:length(k), e(i) = norm(x-X(:,i),inf); end
>> semilogy(k,e);
```

You should see that the convergence is quite slow. Can you estimate how many iterations are needed to obtain an error less than 0.05?

11.5. **Using Cimmino's Method to Solve the Test Problem**
Perform the same experiments as in Exercise 11.4 but using Cimmino's method instead of Kaczmarz's method. All you need to do is substitute `cimmino` for `kaczmarz` in your MATLAB code.

Compared to Kaczmarz, would you say that the convergence of Cimmino is faster, slower, or about the same?

11.6. **Convergence Analysis for Kaczmarz and Cimmino**
We will now more thoroughly examine the convergence of Kaczmarz's and Cimmino's methods for the test problem from Exercise 11.3. To simplify the analysis, we will assume that the errors e_k (11.34) in each iteration satisfy the relation

$$e_k = C^k \cdot e_0, \qquad k = 1, 2, 3, \dots , \tag{11.36}$$

where e_0 is the initial error and C is a constant that depends on the problem and the method. That is, the error is reduced by the factor C in each iteration.

Now assume that we know the error for two iterations k and K with $K > k$. Show, using (11.36), that we can compute the constant C as

$$C = \exp\left(\frac{\log(e_K/e_k)}{K-k}\right). \tag{11.37}$$

Do this for Kaczmarz's and Cimmino's methods in the two previous exercises, e.g., using the results that you computed for 800 and 1600 iterations. What are the two constants for the two methods? Which one is preferable?

11.7. **When the System Is Inconsistent**
We return to the 3×3 test problem from Exercise 10.8. Run enough iterations of Kaczmarz's and Cimmino's methods to ensure that the methods have converged. Do these methods converge to the exact solution in (10.31), to the minimum-norm solution $\boldsymbol{x}_{\mathrm{LS}}^0$, or perhaps to something else?

11.8. **Consistent and Inconsistent Overdetermined Problems**
Let us try to experimentally verify the following convergence behavior for the case $m > n = r$:

- For consistent systems $\boldsymbol{b} \in \mathrm{Range}(\boldsymbol{A})$, both Kaczmarz and Cimmino converge to the least-squares solution $\boldsymbol{x}_{\mathrm{LS}}$, which is identical to the weighted least-squares solution $\boldsymbol{x}_{\mathrm{LS},\boldsymbol{M}}$.

- For inconsistent systems $\boldsymbol{b} \notin \mathrm{Range}(\boldsymbol{A})$, Cimmino converges to $\boldsymbol{x}_{\mathrm{LS},\boldsymbol{M}}$, which is *different* from $\boldsymbol{x}_{\mathrm{LS}}$, while Kaczmarz exhibits cyclic convergence.

We note that in MATLAB, for a full-rank matrix we compute the least-squares solution $\boldsymbol{x}_{\mathrm{LS}}$ by means of `xLS = A\b`.

Generate the test problem from Exercises 9.1 and 11.1 with the 5×4 system matrix $\boldsymbol{A}$ and the right-hand side $\boldsymbol{b}$ in (9.37), and solve the consistent system $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ by means of Kaczmarz and Cimmino. Check that both methods converge to the same solution and that this solution is identical to the least-squares solution $\boldsymbol{x}_{\mathrm{LS}}$.

We will now change the right-hand side $\boldsymbol{b}$ by adding a component that is orthogonal to the range $\mathrm{Range}(\boldsymbol{A})$:

$$\tilde{\boldsymbol{b}} = \boldsymbol{b} + 0.05\,\boldsymbol{e} \qquad \text{with} \qquad \boldsymbol{e} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}. \tag{11.38}$$

Use MATLAB to verify that $\boldsymbol{e}$ is orthogonal to $\mathrm{Range}(\boldsymbol{A})$, i.e., that

$$\boldsymbol{c}_i^T\,\boldsymbol{e} = 0 \qquad \text{for} \qquad i = 1, 2, 3, 4\,. \tag{11.39}$$

Then create the perturbed right-hand side `btilde`, solve the corresponding inconsistent system with both Kaczmarz and Cimmino, and compare with the least-squares solution $\boldsymbol{x}_{\mathrm{LS}}$ as well as the weighted least-squares solution $\boldsymbol{x}_{\mathrm{LS},\boldsymbol{M}}$ defined in (9.34). Do the two methods converge to $\boldsymbol{x}_{\mathrm{LS}}$, $\boldsymbol{x}_{\mathrm{LS},\boldsymbol{M}}$, or something else?

11.9. **The Advantage of Constraints**

We know from the underlying physics of the problem that the attenuation coefficients we wish to reconstruct—the elements of the vector $\boldsymbol{x}$—are nonnegative. And sometimes we also know an upper bound on the elements of this vector. Hence it often makes a lot of sense to include box constraints in the reconstruction process.

In this exercise we return to the test problem from Exercise 11.3, but here we impose the box constraints

$$\boldsymbol{x} \in \mathcal{C} = [0, 1]^n \qquad \Leftrightarrow \qquad 0 \leqslant x_i \leqslant 1 \quad \text{for} \quad i = 1, 2, \ldots, n\,. \tag{11.40}$$

These constraints are specified in the function `kaczmarz` by means of the `options` input as follows:

```
>> kmax = 20;
>> options.lbound = 0;
>> options.ubound = 1;
>> X = kaczmarz(A,b,1:kmax,[ ],options);
```

Plot the 20 reconstructions as images; you should see that they are considerably better than those from Exercise 11.4. What is the improvement in the error for the last iteration with and without constraints?

11.10. **Convergence Analysis for the Constrained Algorithm**
Do the constrained iterates converge faster? To answer this question, we will repeat Exercise 11.4 for the constrained algorithm, still looking at iterations 50, 100, 200, 400, 800, and 1600. Does the constrained Kaczmarz method converge slower or faster than the unconstrained method?

11.11. **Comparison with FBP**
In this exercise we compare the constrained Kaczmarz reconstruction for $k = 1600$ iterations from the previous exercise with the FBP solution computed by means of `fbp(A,b,theta)`—this is a function from AIR Tools II that is similar to the MATLAB `iradon` function, except that it takes the system matrix as input.

When displaying the FBP solution, note that some elements are outside the interval [0,1], so you may want to "chop" them or set the figure color axis to this interval by means of `caxis([0,1])`. Which reconstruction is better?

Optional bonus question. Repeat this comparison using a *limited-angle* problem with `N = 64` and `theta = 3:3:120`. Comment on the results.

11.12. **A Simple Illustration of Semiconvergence**
This exercise illustrates how the noise in the data gives rise to semiconvergence of the iterates $\boldsymbol{x}^{(k)}$ of Kaczmarz's method. We use a rather small system to keep the computing times reasonably small, namely, a parallel-beam test problem with the Shepp–Logan phantom, generated by means of the following code:

```
>> N = 64;
>> theta = 4:4:180;
>> [A,bex,xex] = paralleltomo(N,theta);
```

Run $k_{\max} = 800$ iterations of Kaczmarz's method with these relative noise levels:

$$\rho = 0,\ 0.0015,\ 0.0020,\ 0.0025\,. \tag{11.41}$$

(You can try others, as well, depending on your patience.)

Plot the error history, i.e., the relative error $\|\overline{\boldsymbol{x}}-\boldsymbol{x}^{(k)}\|_2/\|\overline{\boldsymbol{x}}\|_2$, as a function of k, for all four noise levels. You should see the characteristic semiconvergence behavior for all $\rho \neq 0$. At what iteration number k do we reach the smallest error for each ρ?

Note: You will get a different answer each time you run your code, because the results depend on the actual realization of the noise.

11.13. **Semiconvergence Is Associated with Inverse Problems**
Semiconvergence is the fundamental mechanism that makes the algebraic iterative reconstruction methods suited for solving noisy tomography problems (and inverse problems in general). As we have seen, for semiconvergence to be useful, the Picard condition (7.17) must be satisfied, i.e., the exact solution to an inverse problem must be dominated by the initial SVD components. We cannot expect semiconvergence to be useful for an arbitrary ill-conditioned system.

To illustrate this, we will perform a numerical experiment to see if we experience semiconvergence for a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with a synthetic random ill-conditioned matrix generated by the following code:

```
>> N = 32;
>> n = N^2;
>> [U,S,V] = svd(randn(n));
>> S = diag(logspace(0,-12,n));
>> A = U*S*V';
```

The condition number of this matrix is $\text{cond}(\boldsymbol{A}) = 10^{12}$. Perform an experiment similar to the above, this time with the "grains" test image from AIR Tools II:

```
>> Xex = phantomgallery('grains',N);
>> xex = Xex(:);
>> bex = A*xex;
```

Use the relative noise level $\rho = 0.02$ and perform $k_{\max} = 2000$ iterations. Again, you should see the characteristic semiconvergence (if you don't, run the experiment again with a different random `A` and a different noise vector).

Now show the best reconstruction; you will see that—in spite of the semiconvergence—it is really bad. Check whether the Picard condition is satisfied by plotting the singular values of the SVD coefficients of the right-hand side:

```
>> semilogy([diag(S),abs(U'*bex)])
```

Do the right-hand coefficients decay faster than the singular values?

This illustrates the fact that algebraic iterative reconstruction methods do not work when applied to an arbitrary ill-conditioned system; these methods only produce good results when applied to tomographic reconstruction problems and similar inverse problems that satisfy the Picard condition.

11.14. **Semiconvergence for Constrained Problems**
The theory says that we should also observe semiconvergence when we apply the algebraic iterative reconstruction methods to constrained problems. We will investigate this for Kaczmarz's method with box constraints, which we impose by setting `options.lbound = 0` and `options.ubound = 1`.

Repeat Exercise 11.12, still with $k_{\max} = 800$ and now with the box constraints. Discuss the error histories, and compare the results with those from Exercise 11.12.

11.15. **Surprising Semiconvergence?**
In this exercise we also use box constraints. We use a parallel-beam test problem with a special phantom with binary pixel values, generated by means of the following code:

```
>> N = 64;
>> theta = 4:4:180;
>> A = paralleltomo(N,theta);
>> Xex = phantomgallery('binary',N);
>> xex = Xex(:);
>> bex = A*xex;
```

Use the relative noise level $\rho = 0.002$; you only need to perform $k_{\max} = 100$ Kaczmarz iterations. Do you see any substantial growth in the noise error? Can you explain this behavior?

11.16. **SVD Analysis of Cimmino's Method**
The goals of this exercise are twofold: to illustrate how to perform SVD analysis of an iterative method, and to demonstrate how to do so for the basic Cimmino method with a relaxation parameter λ:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \lambda\, \boldsymbol{A}^T \boldsymbol{M} (\boldsymbol{b} - \boldsymbol{A}\, \boldsymbol{x}^{(k)}) \tag{11.42}$$

$$= \boldsymbol{x}^{(k)} + \lambda\, (\boldsymbol{M}^{1/2} \boldsymbol{A})^T \big((\boldsymbol{M}^{1/2} \boldsymbol{b}) - (\boldsymbol{M}^{1/2} \boldsymbol{A})\, \boldsymbol{x}^{(k)} \big) \tag{11.43}$$

$$= \boldsymbol{x}^{(k)} + \lambda\, \widehat{\boldsymbol{A}}^T (\widehat{\boldsymbol{b}} - \widehat{\boldsymbol{A}}\, \boldsymbol{x}^{(k)}), \tag{11.44}$$

with $\boldsymbol{M} = \operatorname{diag}(m \|\boldsymbol{r}_i\|_2^2)^{-1}$, and where we define

$$\widehat{\boldsymbol{A}} = \boldsymbol{M}^{1/2} \boldsymbol{A} \qquad \text{and} \qquad \widehat{\boldsymbol{b}} = \boldsymbol{M}^{1/2} \boldsymbol{b}. \tag{11.45}$$

Note that we compute the square root of the diagonal matrix $\boldsymbol{M}$ simply by computing the square roots of its diagonal elements. Then it follows that the SVD analysis of Cimmino's method follows that of Landweber's method, but with $\boldsymbol{A}$ and $\boldsymbol{b}$ replaced by $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{b}}$. In particular, we should use the SVD of $\widehat{\boldsymbol{A}}$ for the analysis.

Theory question: Assuming a zero starting vector $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, write down the expression for the kth iterate of Cimmino's method in terms of $\widehat{\boldsymbol{b}}$ and the SVD of $\widehat{\boldsymbol{A}}$:

$$\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{U}}\, \widehat{\boldsymbol{\Sigma}}\, \widehat{\boldsymbol{V}}^T. \tag{11.46}$$

Repeat for the iteration error and the noise error.

Now generate a parallel-beam test problem, and the corresponding `Z` = $\boldsymbol{M}^{1/2}$:

```
>> N = 64;
>> theta = 2:2:180;
>> [A,bex] = paralleltomo(N,theta);
>> [A,bex] = purge_rows(A,bex,3);
>> m = size(A,1);
>> d = sqrt(m*sum(A.^2,2));
>> Z = spdiags(1./d(:),0,sparse(m,m));  % Sparse diagonal matrix.
```

The statement `[A,bex] = purge_rows(A,bex,3)` removes all zero rows, as well as rows with fewer than four nonzero elements; this is to ensure a numerically stable computation of $\boldsymbol{M}$ and $\boldsymbol{M}^{1/2}$. To compute the SVD of the sparse matrix `Z*A` use the statement

```
>> [U,S,V] = svd(full(Z*A),0);
```

The second input argument `0` tells the MATLAB `svd` function to compute the economy-size version of the SVD in which

$$\mathtt{U} \in \mathbb{R}^{m \times n}\,, \qquad \mathtt{S} \in \mathbb{R}^{n \times n}\,, \qquad \mathtt{V} \in \mathbb{R}^{n \times n}\,.$$

Save the SVD for later use in Exercise 11.18: `save SavedSVD U S V`.

Now add noise with relative noise level $\rho = 0.02$, and run Cimmino's method on the noisy problem with relaxation parameter $\lambda = 1/\|\widehat{\boldsymbol{A}}\|_2^2 = 1/\mathtt{S}(1,1)^2$ and with $k_{\max} = 5000$ iterations. The relaxation parameter is specified via the `options` input. For the iterations

$$k = 10,\ 50,\ 100,\ 500,\ 1000,\ 5000\,,$$

use the SVD of $\widehat{\boldsymbol{A}}$ to compute and display

- the filter factors $\varphi_i^{(k)}$, $i = 1, 2, \ldots, n$;
- the kth iterate $\boldsymbol{x}^{(k)}$, comparing your result with the output from `cimmino`;
- the iteration error $\overline{\boldsymbol{x}} - \overline{\boldsymbol{x}}^{(k)}$ and its norm; and
- the noise error $\overline{\boldsymbol{x}}^{(k)} - \boldsymbol{x}^{(k)}$ and its norm.

Your results should behave qualitatively like those for Landweber's method. At which iteration k do the iteration error and the noise error have approximately the same norm?

11.17. **Deriving the Trace Term for Cimmino's Method**
This is a theoretical exercise which you may skip if you prefer to do numerical experiments only.

In Section 11.2.3 we derived the stopping rules DP and FTNL specifically for Landweber's method. The latter needs the trace T_k of the influence matrix $\boldsymbol{A}\,\boldsymbol{A}_k^{\#}$,

but it is not obvious how to compute this trace for more general unconstrained methods. Here we specifically consider Cimmino's method, and our task is to derive an expression for the trace T_k for this method.

The key point is to find an expression for the matrix $\boldsymbol{A}_k^{\#}$ corresponding to Cimmino's method, such that the Cimmino iterates can be written as $\boldsymbol{x}^{(k)} = \boldsymbol{A}_k^{\#}\boldsymbol{b}$. According to Eqs. (11.44) and (11.45) in the previous exercise, Cimmino's method is identical to Landweber's method applied to the matrix $\widehat{\boldsymbol{A}}$ and the right-hand side $\hat{\boldsymbol{b}}$ from (11.45). Thus, we can write the Cimmino iterates $\boldsymbol{x}^{(k)}$ as

$$\boldsymbol{x}^{(k)} = \widehat{\boldsymbol{A}}_k^{\#}\,\hat{\boldsymbol{b}} = \widehat{\boldsymbol{A}}_k^{\#}\boldsymbol{M}^{-1/2}\,\boldsymbol{b}\,, \tag{11.47}$$

where, similar to the Landweber algorithm, we define

$$\widehat{\boldsymbol{A}}_k^{\#} = \widehat{\boldsymbol{V}}\,\hat{\boldsymbol{\Phi}}^{(k)}\,\hat{\boldsymbol{\Sigma}}^{-1}\,\widehat{\boldsymbol{U}}^T, \quad \hat{\boldsymbol{\Phi}}^{(k)} = \operatorname{diag}\Big(\hat{\varphi}_i^{(k)}\Big), \quad \hat{\boldsymbol{\Sigma}} = \operatorname{diag}\big(\hat{\sigma}_i\big) \tag{11.48}$$

with

$$\hat{\varphi}_i^{(k)} = 1 - (1 - \lambda\,\hat{\sigma}_i{}^2)^k\,, \qquad i = 1, 2, \ldots, n\,. \tag{11.49}$$

Our goal is to derive an expression for the trace of the influence matrix $\boldsymbol{A}\,\boldsymbol{A}_k^{\#}$, but expressed in terms of the singular values $\hat{\sigma}_i$ of the matrix $\widehat{\boldsymbol{A}}$. First, show that

$$\boldsymbol{A}_k^{\#} = \widehat{\boldsymbol{V}}\,\hat{\boldsymbol{\Phi}}^{(k)}\,\hat{\boldsymbol{\Sigma}}^{-1}\,\widehat{\boldsymbol{U}}^T\boldsymbol{M}^{1/2}\,. \tag{11.50}$$

Then use this result to show that

$$\boldsymbol{A}\,\boldsymbol{A}_k^{\#} = \boldsymbol{M}^{-1/2}\,\widehat{\boldsymbol{U}}\,\hat{\boldsymbol{\Phi}}^{(k)}\widehat{\boldsymbol{U}}^T\boldsymbol{M}^{1/2} = (\boldsymbol{M}^{-1/2}\,\widehat{\boldsymbol{U}})\,\hat{\boldsymbol{\Phi}}^{(k)}(\boldsymbol{M}^{-1/2}\,\widehat{\boldsymbol{U}})^{-1}. \tag{11.51}$$

The multiplication from the left and from the right by $\boldsymbol{M}^{-1/2}\,\widehat{\boldsymbol{V}}$ and its inverse is a *similarity transform* which leaves the eigenvalues unchanged. Using the fact that the trace of a matrix is the sum of its eigenvalues, show that

$$T_k = \operatorname{trace}(\boldsymbol{A}\,\boldsymbol{A}_k^{\#}) = \sum_{i=1}^{n} \hat{\varphi}_i^{(k)}. \tag{11.52}$$

This result allows us to compute the function $\eta^2\,(m - T_k)$ needed in the FTNL stopping rule for Cimmino's method—provided that we know the singular values of $\widehat{\boldsymbol{A}}$. (There is no expression for T_k in terms of the singular values of $\boldsymbol{A}$.)

11.18. **Using the Stopping Rules**

We shall now test the two stopping rules DP and FTNL applied to Landweber's and Cimmino's methods, using the test problem from Exercise 11.16. To use the FTNL stopping rule, you need to compute the trace term T_k using the expressions in (11.22) and (11.32) for Landweber as well as (11.49) and (11.52) for Cimmino.

If you saved the SVD of the matrix $\widehat{\boldsymbol{A}}$ in Exercise 11.16, you can load it again by means of `load SavedSVD`.

We suggest that you use $k_{\max} = 2000$ iterations and a safety factor $\tau = 1.02$. Then test the DP and FTNL stopping rules, e.g., by finding a sign change in the two functions $\|\boldsymbol{\varrho}^{(k)}\|_2^2 - \tau\,\eta^2\,m$ and $\|\boldsymbol{\varrho}^{(k)}\|_2^2 - \tau\,\eta^2\,(m - T_k)$. Compare the number of iterations found by the stopping rules with the optimal k, i.e., the one that minimizes the reconstruction error $\|\overline{\boldsymbol{x}} - \boldsymbol{x}^{(k)}\|_2$.

To get an idea of the robustness of these methods, you can try to repeat the experiments with different realizations of the noise.

11.19. **The Trace Term Estimator**

This exercise is a continuation of the previous one. Here we replace the trace term T_k—which requires singular values—with the Monte Carlo trace estimate T_k^{est} from (11.32). The simplest way to compute a vector `tkest` with values of the trace estimate is to run the iterative method twice, first with $\boldsymbol{b}$ as the right-hand side and with a zero starting vector, and then with a zero right-hand side and a random starting vector. For Landweber's method, using the following code:

```
>> [m,n] = size(A);
>> X = landweber(A,b,1:kmax);
>> w = randn(n,1);
>> Xi = landweber(A,zeros(m,n),1:kmax,w);
>> for k=1:kmax, tkest(k) = n - w'*Xi(:,k); end
```

Obviously the same approach can be used to easily compute the trace estimates for Cimmino's method with `landweber` replaced by `cimmino`.

Repeat the experiments from the previous exercise with T_k replace by T_k^{est}, and compare the two approaches. You can try different random starting vectors to get a feeling for the robustness of the stopping rules based on T_k^{est}.