AIR Tools

A MATLAB Package of Algebraic Iterative Reconstruction Techniques

Version 1.3 for MATLAB 8.0

Per Christian Hansen
Maria Saxild-Hansen

Department of Applied Mathematics and Computer Science
Building 324, Technical University of Denmark
DK-2800 Lyngby, Denmark

July 2015
Abstract

This collection of MATLAB software contains implementations of several Algebraic Iterative Reconstruction methods for tomographic reconstruction (and other imaging problems). These so-called row action methods rely on semi-convergence for achieving the necessary regularization of the problem. Two classes of methods are implemented:

**ART** The class of *algebraic reconstruction techniques* includes Kaczmarz’s classical method, the symmetric Kaczmarz method, and randomized Kaczmarz.

**SIRT** The class of *simultaneous iterative reconstruction techniques* includes Landweber’s and Cimmino’s classical methods plus CAV, DROP, and SART.

In addition we provide a few simplified test problems from medical and seismic tomography. For each iterative method, a number of strategies are available for choosing the relaxation parameter and the stopping rule. The relaxation parameter can be fixed, or chosen adaptively in each iteration; in the former case we provide a new “training” algorithm that finds the optimal parameter for a given test problem. The stopping rules provided are the discrepancy principle, the monotone error rule, and the NCP criterion; for the first two methods “training” can be used to find the optimal discrepancy parameter. The corresponding manuscript is:


We have included the most common algebraic iterative reconstruction methods in the package – but we left out block versions of the methods, which are better suited for other programming languages than MATLAB. Our main contribution is the design of new training algorithms for the optimal relaxation parameter, and the “packaging” of all the methods with identical calling sequences and functionality plus strategies for the various parameters and suitable stopping rules. We are not aware of other MATLAB packages with this functionality.

Notation

All vectors are column vectors, $a_j$ is the $j$th column of $A$, $a_i^T$ is the transposed of the $i$th row of $A$, $\langle x, y \rangle = x^T y$ is the standard inner product, and $\rho(\cdot)$ is the spectral radius (the largest positive eigenvalue).

Acknowledgements

This work is part of the project CSI: Computational Science in Imaging, supported by grant no. 274-07-0065 from the Danish Research Council for Technology and Production Sciences. We are grateful to Jakob Sauer Jørgensen for providing efficient MATLAB code to compute the sparse matrix in the test problems, and to Klaus Mosegaard for suggesting the seismic travel-time tomography test problem. We also thank Tommy Elfving for encouragement and advice during the development of the package, Jim Nagy for pointing out the need for nonnegativity constraints, and the referee for constructive comments and suggestions.

Mikkel Brettschneider, Knud Cordua, Jacob Frøsig, Jakob Sauer Jørgensen, Martin Plesinger, Mikhail Romanov, and Nicolai Riis helped to prepare Version 1.3 with code and revisions.
Revisions Since Version 1.0

- All iterative methods run faster and use less memory, and include box constraints.
- Functions `fanbeam_tomo` and `parallel_tomo` now return the correct variable \( d \).
- Function `rrz` allows input \( b \) with multiple columns.
- All ART methods now allow a damping factor to avoid division by very small norms.
- All SIRT methods had some of the stopping rules changed:
  - The discrepancy principle 'DP' now always uses the standard residual vector \( r^k = b - Ax^k \).
  - The monotone error rule 'ME' now always uses \( 1/(r^k)^T(r^{k+1} + r^k)/\|r^k\|^2 \).
- The default value for \( \lambda \) was changed to 1 in `kaczmarz` and `symkaczmarz`.
- The default value for \( \lambda \) was changed to \( 1/\|A^TMA\|_2 \) in all SIRT methods.
- The following new functions were added:
  - `fbp` – filtered back project which is similar to MATLAB’s `iradon` but conforms to AIR Tools’s use of matrices and vectors.
  - `cart` – columnwise version of Kaczmarz’s method,
  - `phantomgallery` – a collection of 2D phantoms,
  - `seismicwavetomo` – similar to `seismictomo` but without an underlying high-frequency assumption, i.e., no ray model.

Note: for all ART methods NCP is implemented correct but it is not recommended as it often leads to a very early termination. CART with a random choice of columns converges very slowly and is not included.
# Overview of the Package

<table>
<thead>
<tr>
<th>Iterative ART Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cart</strong></td>
</tr>
<tr>
<td><strong>kaczmarz</strong></td>
</tr>
<tr>
<td><strong>randkaczmarz</strong></td>
</tr>
<tr>
<td><strong>symkaczmarz</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iterative SIRT Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cav</strong></td>
</tr>
<tr>
<td><strong>cimmino</strong></td>
</tr>
<tr>
<td><strong>drop</strong></td>
</tr>
<tr>
<td><strong>landweber</strong></td>
</tr>
<tr>
<td><strong>sart</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training Routines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>trainDPME</strong></td>
</tr>
<tr>
<td><strong>trainLambdaART</strong></td>
</tr>
<tr>
<td><strong>trainLambdaSIRT</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fanbeamtomo</strong></td>
</tr>
<tr>
<td><strong>paralleltomo</strong></td>
</tr>
<tr>
<td><strong>phantomgallery</strong></td>
</tr>
<tr>
<td><strong>seismictomo</strong></td>
</tr>
<tr>
<td><strong>seismicwavetomo</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Demo Scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ARTdemo</strong></td>
</tr>
<tr>
<td><strong>nonnegdemo</strong></td>
</tr>
<tr>
<td><strong>SIRTdemo</strong></td>
</tr>
<tr>
<td><strong>trainingdemo</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Auxiliary Routines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>calczeta</strong></td>
</tr>
<tr>
<td><strong>fbp</strong></td>
</tr>
<tr>
<td><strong>rrzr</strong></td>
</tr>
</tbody>
</table>
The Demo Scripts

The demo ARTdemo illustrates the use of the three ART methods kaczmarz, symkaczmarz and randkaczmarz. First the demo creates a parallel-beam tomography test problem using the test problem paralleltomo. Then noise is added to the right-hand side, and the noisy problem is solved using the ART methods with 10 iterations. The result is shown as four images, where one contains the exact image and the remaining images show the solutions computed by means of the three ART methods.

The demo nonnegdemo illustrates the use of nonnegativity constraints in the cimmino and kaczmarz methods. The demo creates a parallel-beam test problem, then adds noise and solves the problem with and without the constraints.

The demo SIRTdemo illustrates the use of the five SIRT methods landweber, cimmino, cav, drop, and sart. First the demo creates a parallel-beam tomography test problem using the test problem paralleltomo. Then noise is added to the right-hand side, and the noisy problem is solved using the SIRT methods with 50 iterations. The result is shown as seven images, where one contains the exact image and the remaining images show the solutions computed by means of the five SIRT methods.

The demo trainingdemo illustrates the use of the training functions trainLambdaART, trainLambdaSIRT, and trainDPME followed by the use of an ART or a SIRT method. In this demo the used SIRT method is cimmino and the used ART method is kaczmarz. First the demo function creates a parallel-beam tomography test problem using the test problem paralleltomo, and noise is added to the right-hand side. Then the training strategy trainLambdaSIRT is used to find the relaxation parameter for cimmino and trainLambdaART is used to find the relaxation parameter for kaczmarz. Including this information the stopping parameter is found for each of the methods, where cimmino uses the ME stopping rule and kaczmarz uses the DP stopping rule. After this we solve the problem with the specified relaxation parameter and stopping rule. The exact image and the results from the methods are shown.

The Use of the restart Parameter

The parameter restart allows one to continue the iterations of an iterative method, continuing from the last iteration of a previous call:

```matlab
[X,info,restart] = landweber(A,b,1:k); % First call.  
options.testart = restart; % Second call with iterations 
moreX = landweber(A,b,1:k2,X(:,end),options); % from k+1 to k+k2.
```

If the same matrix A is involved in repeated calls to the same iterative method, restart can be used to avoid re-computation of the spectral radius:

```matlab
options.lambda = lambda; % Fixed lambda. 
[X,info,restart] = landweber(A,b1,k,[],options); % First call. 
options.restart = restart; 
[X,info] = landweber(A,b2,k,[],options); % Second call. 
```

The above example is for a fixed λ; the same technique can be applied when using a relaxation-parameter rule that involves the use of the spectral radius.

5
calczeta

Purpose:
Calculates a specific root of a certain polynomial (used in the SIRT methods).

Synopsis:
\[ z = \text{calczeta}(k) \]

Description:
This function uses Newton’s method to compute the unique root in the interval \((0, 1)\) of the polynomial equation:

\[
(2k - 1)z^{k-1} - (z^{k-2} + \ldots + z + 1) = 0, \quad k \geq 2.
\]

The input \(k\) can be given as both a scalar or a vector, and the corresponding root or roots are returned in the output \(z\).

The function \text{calczeta} is used in the functions \text{cav}, \text{cimmino}, \text{drop}, \text{landweber}, \text{sart}, and \text{symkaczmarz}.

Example:
Calculate the roots for \(k\) from 2 up to 100 and plot the found roots.

\begin{verbatim}
    k = 2:100;
    z = calczeta(k);
    figure, plot(k,z,'bo')
\end{verbatim}

See also:
\text{cav, cimmino, drop, landweber, sart, symkaczmarz.
**cart**

**Purpose:**

Columnwise version of Kaczmarz’s method.

**Synopsis:**

```
[X info] = cart(A,b,K)
[X info] = cart(A,b,K,x0)
[X info] = cart(A,b,K,x0,options)
```

**Algorithm:**

For arbitrary starting vector \( x^0 \in \mathbb{R}^n \) one iteration of the algorithm \texttt{cart} consists of the following steps:

\[
x_j \leftarrow x_j + \lambda \frac{a_{j}^T(b - A x)}{\|a_{j}\|_2^2} \quad j = 1, \ldots, n,
\]

where \( a_j \) is the \( j \)th columns of \( A \).

**Description:**

The function implements the columnwise version of Kaczmarz’s iterative method for solving the linear system \( A x = b \). The starting vector is \( x0 \); if no starting vector is given then \( x0 = 0 \) is used.

The numbers given in the vector \( K \) are iteration numbers, that specify which iterations are stored in the output matrix \( X \). If a stopping rule is selected (see below) and \( K = [ ] \), then \( X \) contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector \( K \) or by the stopping rule specified in the field \texttt{stoprule} in the struct \texttt{options}. If \( K \) is empty a stopping rule must be specified.

The relaxation parameter is given in the field \texttt{lambda} in the struct \texttt{options} as a constant. As default \texttt{lambda} is set to 0.25.

The second output \texttt{info} is a vector with two elements. The first element is an indicator that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, 1 denotes that the NCP-rule stopped the iterations and 2 denotes that the DP-rule stopped the iterations. The second element in \texttt{info} is the number of used iterations.

**Use of options:**

The following fields in \texttt{options} are used in this function:

- \texttt{options.lambda} = \( c \), a constant satisfying \( 0 < c < 2 \). A warning is given if this requirement is estimated to be violated.

- \texttt{options.nonneg} Logical; if true then nonnegativity in enforced in each step.

- \texttt{options.box} Double; upper bound \( L \) in box constraint \([0, L]\) on pixel values.
- **options.damping** Double; a parameter $D$ to avoid division by very small row norms by adding $D \cdot \max_i \{\|a_j\|_2^2\}$ to $\|a_j\|_2^2$.

- **options.stoprule**
  - **options.stoprule.type**
    - **options.stoprule.type** = 'none', where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    - **options.stoprule.type** = 'NCP', where the optimal number of iterations $k_n$ is chosen according to Normalized Cumulative Periodogram.
    - **options.stoprule.type** = 'DP', where the stopping index is determined according to the discrepancy principle (DP).
    - **options.stoprule.taudelta** = $\tau \delta$, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule type DP.

**Example:**

Generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute 10 cart iterations, and show the last iterate:

```matlab
[A b x] = paralleltomo(50,0:5:179,150);
e = randn(size(b)); e = e/norm(e);
b = b + 0.05*norm(b)*e;
X = cart(A,b,1:10);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

**See also:**

kaczmarz, randkaczmarz, symkaczmarz.
cav

Purpose:
Component Averaging (CAV) iterative method.

Synopsis:

\[ [X \text{ info restart}] = \text{cav}(A,b,K) \]
\[ [X \text{ info restart}] = \text{cav}(A,b,K,x0) \]
\[ [X \text{ info restart}] = \text{cav}(A,b,K,x0,\text{options}) \]

Algorithm:

For arbitrary \( x^0 \in \mathbb{R}^n \) the algorithm for cav takes the following form:

\[
x^{k+1} = x^k + \lambda_k A^T D_S(b - Ax^k), \quad D_S = \text{diag}\left(\frac{w_i}{\sum_{j=1}^n s_j a_{ij}^2}\right),
\]

where \( s_j \) is the number of nonzero elements in column \( j \) of \( A \), and \( w_i \) are user specified weights (default: \( w_i = 1 \)).

Description:

The function implements the Component Averaging (CAV) iterative method for solving the linear system \( Ax = b \). The starting vector is \( x0 \); if no starting vector is given then \( x0 = 0 \) is used.

The numbers given in the vector \( K \) are iteration numbers, that specify which iterations are stored in the output matrix \( X \). If a stopping rule is selected (see below) and \( K = [\] , then \( X \) contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector \( K \) or by the stopping rule specified in the field stoprule in the struct options. If \( K \) is empty a stopping rule must be specified.

The relaxation parameter is given in the field lambda in the struct options, either as a constant or as a string that determines the method to compute lambda. As default lambda is set to \( 1/\hat{\sigma}_1^2 \), where \( \hat{\sigma}_1 \) is an estimate of the largest singular value of \( D_S^{1/2} A \).

The second output info is a vector with two elements. The first element is an indicator, that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, 1 denotes that the NCP-rule stopped the iterations, 2 denotes that the DP-rule stopped the iteration and 3 denotes that the ME-rule stopped the iterations. The second element in info is the number of used iterations.

The struct restart, which can be given as output, contains in the field s1 the estimated largest singular value. restart also returns a vector containing the diagonal of the matrix \( D_S \) in the field M and an empty vector in the field T. The struct restart can also be given as input in the struct options such that the program does not have to recompute the contained values. We recommend only to use this, if the user has good knowledge of MATLAB and is completely sure of the use of restart as input.
Use of options:
The following fields in options are used in this function:

- **options.lambda**:
  - **options.lambda = c**, where c is a constant, satisfying $0 < c < 2/\tilde{\sigma}_1^2$. A warning is given if this requirement is estimated to be violated.
  - **options.lambda = 'line'**, where the line search method is used to compute the value for $\lambda_k$ in each iteration.
  - **options.lambda = 'psi1'**, where the method psi1 computes the values for $\lambda_k$ using the $\Psi_1$-based relaxation.
  - **options.lambda = 'psi1mod'**, where the method psi1mod computes the values for $\lambda_k$ using the modified $\Psi_1$-based relaxation with $\nu = 2$. The parameter $\nu$ can be changed in the code.
  - **options.lambda = 'psi2'**, where the method psi2 computes the values for $\lambda_k$ using the $\Psi_2$-based relaxation.
  - **options.lambda = 'psi2mod'**, where the method psi2mod computes the values for $\lambda_k$ using the modified $\Psi_2$-based relaxation with $\nu = 1.5$. The parameter $\nu$ can be changed in the code.

- **options.nonneg** Logical; if true then nonnegativity is enforced in each iteration.

- **options.box** Double; upper bound $L$ in box constraint $[0, L]$ on pixel values.

- **options.restart**
  - **options.restart.M =** a vector with the diagonal of $D_S$.
  - **options.restart.s1 = $\tilde{\sigma}_1$**, the estimated largest singular value of $D_S^{1/2} A$.

- **options.stoprule**
  - **options.stoprule.type**
    - **options.stoprule.type = 'none'**, where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    - **options.stoprule.type = 'NCP'**, where the optimal number of iterations $k_s$ is chosen according to Normalized Cumulative Periodogram.
    - **options.stoprule.type = 'DP'**, where the stopping index $k_s$ is determined according to the discrepancy principle (DP).
    - **options.stoprule.type = 'ME'**, where the stopping index $k_s$ is determined according to the monotone error rule (ME).
    - **options.stoprule.taudelta = $\tau \delta$**, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule types DP and ME.

- **options.w**, an $m$-dimensional vector of weights.
Example:

Generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute 50 cav iterations, and show the last iterate:

```matlab
[A b x] = paralleltomo(50,0:5:179,150);
e = randn(size(b)); e = e/norm(e);
b = b + 0.05*norm(b)*e;
X = cav(A,b,1:50);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

See also: cimmino, drop, landweber, sart.

References:


**cimmino**

**Purpose:**

Cimmino’s iterative projection method.

**Synopsis:**

```
[X info restart] = cimmino(A,b,K)
[X info restart] = cimmino(A,b,K,x0)
[X info restart] = cimmino(A,b,K,x0,options)
```

**Algorithm:**

For arbitrary $x^0 \in \mathbb{R}^n$ the algorithm for `cimmino` take the following form:

$$
x^{k+1} = x^k + \lambda_k A^T D (b - Ax^k),
$$

$$
D = \frac{1}{m} \text{diag} \left( \frac{w_i}{\|A(:,i)\|_2} \right),
$$

where $w_i$ are use specified weights (default: $w_i = 1$).

**Description:**

The function implements Cimmino’s iterative projection method for solving linear systems $Ax = b$. The starting vector is $x0$; if no starting vector is given, then $x0 = 0$ is used.

The numbers given in the vector $K$ are iteration numbers that specify which iterations are stored in the output matrix $X$. If a stopping rule us selected (see below) and $K = [ ]$, then $X$ contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector $K$ or by the stopping rule specified in the field `stoprule` in the struct `options`. If $K$ is empty a stopping rule must be specified.

The relaxation parameter is given in the field `lambda` in the struct `options`, either as a constant or as a string that determines the method to compute `lambda`. As default `lambda` is set to $1/\sigma_1^2$, where $\sigma_1$ is an estimate of the largest singular value of $D^{1/2}A$.

The second output `info` is a vector with two elements. The first element is an indicator that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, 1 denotes that the NCP-rule stopped the iterations, 2 denotes that the DP-rule stopped the iteration and 3 denotes that the ME-rule stopped the iterations. The second element in `info` is the number of used iterations.

The struct `restart`, which can be given as output, contains in the field `s1` the estimated largest singular value. `restart` also returns a vector containing the diagonal of the matrix $M$ in the field `M` and an empty vector in the field `T`. The struct `restart` can also be given as input in the struct `options`, such that the program do not have to recompute the contained values. We recommend only to use this, if the user has good knowledge of MATLAB and is completely sure of the use of `restart` as input.
Use of options

The following fields in options are used in this function:

- **options.lambda**:
  - `options.lambda = c`, where `c` is a constant, satisfying $0 < c < 2/\bar{\sigma}_1^2$. A warning is given if this requirement is estimated to be violated.
  - `options.lambda = 'line'`, where the line search method is used to compute the value for $\lambda_k$ in each iteration.
  - `options.lambda = 'psi1'`, where the method `psi1` computes the values for $\lambda_k$ using the $\Psi_1$-based relaxation.
  - `options.lambda = 'psi1mod'`, where the method `psi1mod` computes the values for $\lambda_k$ using the modified $\Psi_1$-based relaxation with $\nu = 2$. The parameter $\nu$ can be changed in the code.
  - `options.lambda = 'psi2'`, where the method `psi2` computes the values for $\lambda_k$ using the $\Psi_2$-based relaxation.
  - `options.lambda = 'psi2mod'`, where the method `psi2mod` computes the values for $\lambda_k$ using the modified $\Psi_2$-based relaxation with $\nu = 1.5$. The parameter $\nu$ can be changed in the code.

- **options.nonneg** Logical; if true then nonnegativity is enforced in each iteration.

- **options.box** Double; upper bound $L$ in box constraint $[0, L]$ on pixel values.

- **options.restart**
  - `options.restart.M = a vector with the diagonal of $M$.
  - `options.restart.s1 = \bar{\sigma}_1`, the estimated largest singular value of $M^{1/2}A$.

- **options.stoprule**
  - `options.stoprule.type`
    - `options.stoprule.type = 'none'`, where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    - `options.stoprule.type = 'NCP'`, where the optimal number of iterations $k_\star$ is chosen according to Normalized Cumulative Periodogram.
    - `options.stoprule.type = 'DP'`, where the stopping index $k_\star$ is determined according to the discrepancy principle (DP).
    - `options.stoprule.type = 'ME'`, where the stopping index $k_\star$ is determined according to the monotone error rule (ME).
    - `options.stoprule.taudelta = \tau \delta`, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule types DP and ME.

- **options.w**, an $m$-dimensional vector of weights.
Example:

Generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute $50$ cimmino iterations, and show the last iterate:

\[
[A \ b \ x] = \text{paralleltomo}(50, 0:5:179, 150);
\]
\[
e = \text{randn(size(b))}; \ e = e/\text{norm(e)};
\]
\[
b = b + 0.05*\text{norm(b)}*e;
\]
\[
X = \text{cimmino}(A,b,1:50);
\]
\[
\text{imagesc}(\text{reshape}(X(:,end),50,50))
\]
\[
\text{colormap gray, axis image off}
\]

See also:

cav, drop, landweber, sart.

References:


**drop**

**Purpose:**
Diagonally Relaxed Orthogonal Projections (DROP) iterative method.

**Synopsis:**

\[ [X \text{ info restart}] = \text{drop}(A,b,K) \]
\[ [X \text{ info restart}] = \text{drop}(A,b,K,x0) \]
\[ [X \text{ info restart}] = \text{drop}(A,b,K,x0,options) \]

**Algorithm:**
For arbitrary \( x^0 \in \mathbb{R}^n \) the algorithm for the drop method takes the following form:

\[ x^{k+1} = x^k + \lambda_k S^{-1} A^T M(b - A x^k), \quad M = \text{diag} \left( \frac{w_i}{\|A(i,:)\|_2} \right), \]

where \( S = \text{diag} (s_j) \) and \( s_j \) is the number of nonzero elements in column \( j \) of \( A \) for \( i = 1, \ldots, m \). Moreover, \( w_i \) are user specified weights (default \( w_i = 1 \)).

**Description:**
The function implements the Diagonally Relaxed Orthogonal Projections (DROP) iterative method for solving the linear system \( A x = b \). The starting vector is \( x^0 \); if no starting vector is given, then \( x^0 = 0 \) is used.

The numbers given in the vector \( K \) are the iteration numbers, that specify which iterations are stored in the output matrix \( X \). If a stopping rule is selected (see below) and \( K = [\] \), then \( X \) contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector \( K \) or by the stopping rule specified in the field stoprule in the struct options. If \( K \) is empty a stopping rule must be specified.

The relaxation parameter is given in the field lambda in the struct options, either as a constant or as a string that determines the method to compute lambda. As default lambda is set to \( 1/\hat{\rho} \), where \( \hat{\rho} \) is an estimate of the spectral radius of \( S^{-1} A^T M A \).

The second output info is a vector with two elements. The first element is an indicator, that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, denotes that the NCP-rule stopped the iterations, 2 denotes that the DP-rule stopped the iterations and 3 denotes that the ME-rule stopped the iterations. The second element in info is the number of used iterations.

The struct restart, which can be given as output, contains in the field s1 the estimated largest singular value. restart also returns a vector containing the diagonal of the matrix \( M \) in the field M and the diagonal of the matrix \( S \) in the field T. The struct restart can also be given as input in the struct options, such that the program do not have to recompute the contained values. We recommend only to use this, if the user has good knowledge of MATLAB and is completely sure of the use of restart as input.
Use of options

The following fields in options are used in this function:

- **options.lambda**:
  - **options.lambda = c**, where $c$ is a constant, satisfying $0 < c < 2/\hat{\rho}$. A warning is given if this requirement is estimated to be violated.
  - **options.lambda = 'line'**, where the line search method is used to compute the value for $\lambda_k$ in each iteration.
  - **options.lambda = 'psi1'**, where the method psi1 computes the values for $\lambda_k$ using the $\Psi_1$-based relaxation.
  - **options.lambda = 'psi1mod'**, where the method psi1mod computes the values for $\lambda_k$ using the modified $\Psi_1$-based relaxation with $\nu = 2$. The parameter $\nu$ can be changed in the code.
  - **options.lambda = 'psi2'**, where the method psi2 computes the values for $\lambda_k$ using the $\Psi_2$-based relaxation.
  - **options.lambda = 'psi2mod'**, where the method psi2mod computes the values for $\lambda_k$ using the modified $\Psi_2$-based relaxation with $\nu = 1.5$. The parameter $\nu$ can be changed in the code.

- **options.nonneg** Logical; if true then nonnegativity is enforced in each iteration.

- **options.box** Double; upper bound $L$ in box constraint $[0, L]$ on pixel values.

- **options.restart**
  - **options.restart.M** = a vector containing the diagonal of $M$.
  - **options.restart.T** = a vector containing the diagonal of $S^{-1}$.
  - **options.restart.s1** = $\tilde{\sigma}_1$, where $\tilde{\sigma}_1 = \sqrt{\hat{\rho}}$.

- **options.stoprule**
  - **options.stoprule.type**
    - **options.stoprule.type = 'none'**, where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    - **options.stoprule.type = 'NCP'**, where the optimal number of iterations $k_\star$ is chosen according to Normalized Cumulative Periodogram.
    - **options.stoprule.type = 'DP'**, where the stopping index is determined according to the discrepancy principle (DP).
    - **options.stoprule.type = 'ME'**, where the stopping index is determined according to the monotone error rule (ME).
  - **options.stoprule.taudeelta** = $\tau \delta$, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule types DP and ME.

- **options.w**, an $m$-dimensional vector of weights.
Example:

Generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute 50 drop iterations, and show the last iterate:

$$[A \ b \ x] = paralleltomo(50,0:5:179,150);$$
$$e = \text{randn(size}(b)); \ e = e/norm(e);$$
$$b = b + 0.05*\text{norm}(b)*e;$$
$$X = \text{drop}(A,b,1:50);$$
$$\text{imagesc}(\text{reshape}(X(:,\text{end}),50,50))$$
$$\text{colormap gray, axis image off}$$

See also:

cav, cimmino, landweber, sart.

References:

fanbeamtomo

Purpose:

Creates a two-dimensional fan-beam tomography test problem.

Synopsis:

\[
\begin{align*}
[A\ b\ x\ \theta\ p\ R\ w] &= \text{fanbeamtomo}(N) \\
[A\ b\ x\ \theta\ p\ R\ w] &= \text{fanbeamtomo}(N,\theta) \\
[A\ b\ x\ \theta\ p\ R\ w] &= \text{fanbeamtomo}(N,\theta,p) \\
[A\ b\ x\ \theta\ p\ R\ w] &= \text{fanbeamtomo}(N,\theta,p,R) \\
[A\ b\ x\ \theta\ p\ R\ w] &= \text{fanbeamtomo}(N,\theta,p,R,w) \\
[A\ b\ x\ \theta\ p\ R\ w] &= \text{fanbeamtomo}(N,\theta,p,R,w,\text{isDisp})
\end{align*}
\]

Description:

This function creates a two-dimensional tomography test problem using fan beams.

- A 2-dimensional domain is divided into \( N \) equally spaced intervals in both dimension creating \( N^2 \) cells.

- For each specified angle \( \theta \), given in degrees, a source is located with distance \( R \) to the center of the domain. The default values for the angles is \( \theta = 0:359 \). The distance from the center of the domain to the sources is given in the unit of side lengths and the default value of \( R \) is 2.

- From each source \( p \) equiangular rays penetrate the domain with a span of \( w \) between the first and the last ray. The number of rays \( p \) has default value equal to \( \text{round}(\sqrt{2}N) \). The default value of the span \( w \) is calculated such that from \((0,RN)\) the first ray hits the point \((-N/2,N/2)\) and the last hits \((N/2,N/2)\).

If the input \( \text{isDisp} \) is different from 0 then the function also creates an illustration of the problem with the used angles and rays etc. As default \( \text{isDisp} \) is 0.

The function returns a coefficient matrix \( A \) with the dimension \( \text{length}(\theta) \cdot p \times N^2 \), the right hand side \( b \), and the phantom head reshaped as a vector \( x \) with elements between 0 and 1. The figure below illustrates the phantom head for \( N = 100 \). In case that default values are used the function also returns the used angles \( \theta \), the number of used rays for each angle \( p \), the used distance from the source to the center of the domain given in side lengths \( R \) and the used span of the rays \( w \).

Algorithm:

The matrix element \( a_{ij} \) is defined as the length of the \( i \)th ray through the \( j \)th cell, with \( a_{ij} = 0 \) if ray \( i \) does not go through cell \( j \). The exact solution of the head phantom is reshaped as a vector \( x \), and the \( i \)th element in the right hand side \( b \) is

\[
b_i = \sum_{j=1}^{N^2} a_{ij} x_j, \quad i = 1, \ldots, \text{length}(\theta) \cdot p.
\]
Example:
Create a test problem and visualize the exact image:

\[
N = 64; \theta = 0:5:359; p = 2*N; R = 2; \\
[A \ b \ x] = \text{fanbeamtomo}(N,\theta,p,R); \\
\text{imagesc}(\text{reshape}(x,N,N)) \\
\text{colormap gray, axis image off}
\]

See also:

\text{paralleltomo, seismictomo}.

References:

Purpose:
Filtered back projection.

Synopsis:

\[
x = \text{kaczmarz}(A, b, \theta)
\]

Algorithm:
The algorithm consists of two steps: a filtering which is identical to that in MATLAB’s \text{iradon}
and a back projection which is implemented as a multiplication with the transposed of the
system matrix \(A\).

Description:
This function in quite similar to MATLAB’s \text{iradon}, but it is designed to conform with AIR Tools. The input parameters are the system matrix \(A\) from \text{paralleltomo}, the right-hand side vector \(b\) (corresponding to the sinogram), the vector \(\theta\) of projection angles, and the optional string \text{filter} (see the documentation for \text{iradon} for details). The output vector \(x\) is the computed reconstruction.

Example:
Generate a \(256 \times 256\) parallel-beam tomography problem, and compute the filtered back projection reconstruction using the standard Ram-Lak filter:

\[
N = 256;
\theta = 1:180;
[A, b] = \text{paralleltomo}(N, \theta);
x = \text{fbp}(A, b, \theta);
\text{imagesc}(\text{reshape}(x, N, N))
\text{colormap gray, axis image off}
\]

See also:
MATLAB’s \text{iradon}.

References:
**kaczmarz**

**Purpose:**

Kaczmarz’s method also known as the algebraic reconstruction technique (ART).

**Synopsis:**

\[
\begin{align*}
[X \text{ info}] &= \text{kaczmarz}(A,b,K) \\
[X \text{ info}] &= \text{kaczmarz}(A,b,K,x0) \\
[X \text{ info}] &= \text{kaczmarz}(A,b,K,x0,\text{options})
\end{align*}
\]

**Algorithm:**

For arbitrary starting vector \( x^0 \in \mathbb{R}^n \) one iteration of the algorithm \text{kaczmarz} consists of the following steps:

\[
x \leftarrow x + \lambda_k \frac{b_i - \langle a^i, x \rangle}{\|a^i\|_2^2} a^i, \quad i = 1, \ldots, m.
\]

**Description:**

The function implements Kaczmarz’s iterative method for solving the linear system \( A x = b \). The starting vector is \( x_0 \); if no starting vector is given then \( x_0 = 0 \) is used.

The numbers given in the vector \( K \) are iteration numbers, that specify which iterations are stored in the output matrix \( X \). If a stopping rule is selected (see below) and \( K = [ ] \), then \( X \) contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector \( K \) or by the stopping rule specified in the field \text{stoprule} in the struct \text{options}. If \( K \) is empty a stopping rule must be specified.

The relaxation parameter is given in the field \text{lambda} in the struct \text{options} as a constant. As default \text{lambda} is set to 0.25.

The second output \text{info} is a vector with two elements. The first element is an indicator that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, 1 denotes that the NCP-rule stopped the iterations and 2 denotes that the DP-rule stopped the iterations. The second element in \text{info} is the number of used iterations.

**Use of \text{options}:**

The following fields in \text{options} are used in this function:

- \text{options.lambda} = \( c \), a constant satisfying \( 0 < c < 2 \). A warning is given if this requirement is estimated to be violated.

- \text{options.nonneg} Logical; if true then nonnegativity in enforced in each step.

- \text{options.box} Double; upper bound \( L \) in box constraint \([0, L]\) on pixel values.

- \text{options.damping} Double; a parameter \( D \) to avoid division by very small row norms by adding \( D \cdot \max_i \{\|a^i\|_2^2\} \) to \( \|a^i\|_2^2 \).
- `options.stoprule`
  
  - `options.stoprule.type`
    
    - `options.stoprule.type` = 'none', where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    
    - `options.stoprule.type` = 'NCP', where the optimal number of iterations $k$, is chosen according to Normalized Cumulative Periodogram.
    
    - `options.stoprule.type` = 'DP', where the stopping index is determined according to the discrepancy principle (DP).
    
    - `options.stoprule.taudelta` = $\tau\delta$, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule type DP.

Example:

Generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute 10 `kaczmarz` iterations, and show the last iterate:

```matlab
[A b x] = paralleltomo(50,0:5:179,150);
e = randn(size(b)); e = e/norm(e);
b = b + 0.05*norm(b)*e;
X = kaczmarz(A,b,1:10);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

See also:

cart, randkaczmarz, symkaczmarz.

References:


landweber

Purpose:
The Landweber iterative method.

Synopsis:
[X info restart] = landweber(A,b,K)
[X info restart] = landweber(A,b,K,x0)
[X info restart] = landweber(A,b,K,x0,options)

Algorithm:
For arbitrary \( x^0 \in \mathbb{R}^n \) the algorithm for landweber takes the following form:
\[
x^{k+1} = x^k + \lambda_k A^T(b - Ax^k).
\]

Description:
The function implements the classical Landweber iterative method for solving the linear system \( Ax = b \). The starting vector is \( x^0 \); if no starting vector is given then \( x^0 = 0 \) is used.

The numbers given in the vector \( K \) are iteration numbers, that specify which iterations are stored in the output matrix \( X \). If a stopping rule is selected (see below) and \( K = [ ] \), then \( X \) contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector \( K \) or by the stopping rule specified in the field stoprule in the struct options. If \( K \) is empty a stopping rule must be specified.

The relaxation parameter is given in the field lambda in the struct options, either as a constant or as a string that determines the method to compute lambda. As default lambda is set to \( 1/\tilde{\sigma}_1^2 \), where \( \tilde{\sigma}_1 \) is an estimate of the largest singular value of \( A \).

The second output is a vector with two elements. The first element is an indicator, that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, 1 denotes that the NCP-rule stopped the iterations, 2 denotes that the DP-rule stopped the iterations and 3 denotes that the ME-rule stopped the iterations. The second element is info is the number of used iterations.

The struct restart, which can be given as output, contains in the field s1 the estimated largest singular value. restart also returns an empty vector in both the fields M and T. The struct restart can also be given as input in the struct options, such that the program does not have to recompute the contained values. We recommend only to use this, if the user has good knowledge of MATLAB and is completely sure of the use of restart as input.

Use of options:
The following fields in options are used in this function:

- options.lambda:
- `options.lambda = c`, where `c` is a constant, satisfying $0 < c < 2/\hat{\sigma}_1^2$. A warning is given if this requirement is estimated to be violated.

- `options.lambda = 'line'`, where the line search method is used to compute the value for $\lambda_k$ in each iteration.

- `options.lambda = 'psi1'`, where the method `psi1` computes the values for $\lambda_k$ using the $\Psi_1$-based relaxation.

- `options.lambda = 'psi1mod'`, where the method `psi1mod` computes the values for $\lambda_k$ using the modified $\Psi_1$-based relaxation with $\nu = 2$. The parameter $\nu$ can be changed in the code.

- `options.lambda = 'psi2'`, where the method `psi2` computes the values for $\lambda_k$ using the $\Psi_2$-based relaxation.

- `options.lambda = 'psi2mod'`, where the method `psi2mod` computes the values for $\lambda_k$ using the modified $\Psi_2$-based relaxation with $\nu = 1.5$. The parameter $\nu$ can be changed in the code.

- `options.nonneg` Logical; if true then nonnegativity in enforced in each iteration.

- `options.box` Double; upper bound $L$ in box constraint $[0, L]$ on pixel values.

- `options.restart`
  - `options.restart.s1 = \hat{\sigma}_1`, where $\hat{\sigma}_1$ is the estimated largest singular value of $A$.

- `options.stoprule`
  - `options.stoprule.type`
    - `options.stoprule.type = 'none'`, where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    - `options.stoprule.type = 'NCP'`, where the optimal number of iterations $k_s$ is chosen according to Normalized Cumulative Periodogram.
    - `options.stoprule.type = 'DP'`, where the stopping index $k_s$ is determined according to the discrepancy principle (DP).
    - `options.stoprule.type = 'ME'`, where the stopping index $k_s$ is determined according to the monotone error rule (ME).

- `options.stoprule.taudelta = \tau \delta`, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule types DP and ME.

**Example:**

We generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute 50 landweber iterations, and show the last iterate:

```matlab
[A b x] = paralleltomo(50,0:5:179,150);
e = randn(size(b)); e = e/norm(e);
b = b + 0.05*norm(b)*e;
X = landweber(A,b,1:50);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```
See also:
cav, cimmino, drop, sart

References:

**paralleltomo**

**Purpose:**

Creates a two-dimensional parallel-beam tomography test problem.

**Synopsis:**

```matlab
[A b x theta p w] = paralleltomo(N)
[A b x theta p w] = paralleltomo(N,theta)
[A b x theta p w] = paralleltomo(N,theta,p)
[A b x theta p w] = paralleltomo(N,theta,p,w)
[A b x theta p w] = paralleltomo(N,theta,p,w,isDisp)
```

**Description:**

This function creates a two-dimensional tomography test problem using parallel beams.

- A 2-dimensional domain is divided into $N$ equally spaced intervals in both dimensions creating $N^2$ cells.

- For each specified angle `theta`, given in degrees, $p$ parallel rays penetrate the domain. The default values for the angles are $\theta = 0:179$. The rays are arranged symmetrically around the center of the domain, such that the width from the first to the last ray is $w$. The default value of $w$ is $\frac{p}{2N}$.

- The number of rays $p$ has the default value equal to $\text{round}(\sqrt{2N})$.

If the input `isDisp` is different from 0 then the function also creates an illustration of the problem with the used angles and rays etc. As default `isDisp` is 0.

The function returns a coefficient matrix $A$ with the dimension $\text{length(theta)} \times p \times N^2$, the right hand side $b$, and the phantom head reshaped as a vector $x$ with elements between 0 and 1. The figure below illustrates the phantom head for $N = 100$. In case the default values are used, the function also returns the used angles `theta`, the number of used rays for each angle $p$, and the used width of the rays $w$.

**Algorithm:**

The matrix element $a_{ij}$ is defined as the length of the $i$th ray through the $j$th cell, with $a_{ij} = 0$ if ray $i$ does not go through cell $j$. The exact solution of the head phantom is reshaped as a vector $x$, and the $i$th element in the right hand side $b$ is

$$b_i = \sum_{j=1}^{p^2} a_{ij} x_j, \quad i = 1, \ldots, \text{length(theta)} \cdot p.$$

**Example:**

Create a test problem and visualize the exact image:
N = 64; theta = 0:5:179; p = 2*N;
[A b x] = paralleltomo(N,theta,p);
imagesc(reshape(x,N,N))
colormap gray, axis image off

See also:

fanbeamto mo, seismictomo.

References:

phantomgallery

Purpose:

Creates different two-dimensional phantoms for use in tomography test problem.

Synopsis:

\[
\begin{align*}
im &= \text{paralleltomo}(\text{name},N) \\
im &= \text{paralleltomo}(\text{name},N,P1) \\
im &= \text{paralleltomo}(\text{name},N,P1,P2) \\
im &= \text{paralleltomo}(\text{name},N,P1,P2,P3)
\end{align*}
\]

Description:

This function creates several different two-dimensional phantoms for use in tomography test problem. The size is \( N \times N \) and the pixel values are between 0 and 1. The type of phantom is specified by \text{name} and the characteristics are controlled by optional parameters. The following phantoms are available (for examples see the figure below).

- **name = ‘smooth’**: a smooth image
  \[
  \begin{align*}
im &= \text{phantomgallery}('\text{smooth}',N,P1) \\
P1 &= 1, 2, 3 \text{ or } 4 \text{ defines four different smooth functions (default } = 4) 
\end{align*}
\]
  The image is constructed by adding four different Gaussian functions.

- **name = ‘binary’**: a random image with binary pixel values arranged in domains
  \[
  \begin{align*}
im &= \text{phantomgallery}('\text{binary}',N,P1) \\
P1 &= \text{seed for random number generator} 
\end{align*}
\]
  The image is dominated by horizontal structures.

- **name = ‘threephases’**: a random image with pixel values 0, 0.5, 1 arranged in domains
  \[
  \begin{align*}
im &= \text{phantomgallery}('\text{threephases}',N,P1,P2) \\
P1 &= \text{controls the number of domains (default } = 100) \\
P2 &= \text{seed for random number generator} 
\end{align*}
\]
  The image is a model of a three-phase object.

- **name = ‘threephasessmooth’**: similar to ‘threephases’, but the domains have smoothly varying pixel values and there is a smooth background
  \[
  \begin{align*}
im &= \text{phantomgallery}('\text{threephasessmooth}',N,P1,P2,P3) \\
P1 &= \text{controls the number of domains (default } = 100) \\
P2 &= \text{controls the intensity variation within each domain (default } = 1.5) \\
P3 &= \text{seed for random number generator} 
\end{align*}
\]

- **name = ‘fourphases’**: a random image similar to ‘binary’ but with three phases separated by (thin) structures that form the fourth phase
im = phantomgallery('fourphases',N,P1)
P1 = seed for random number generator

- **name** = 'grains': a random image with Voronoi cells
  
im = phantomgallery('grains',N,P1,P2)
P1 = number of cells in the image (default = 3*sqrt(N))
P2 = seed for random number generator

  The image is a model of grains with different pixel intensities.

- **name** = 'ppower': a random image with patterns of nonzero pixels
  
im = phantomgallery('ppower',P1,P2)
P1 = the ratio of nonzero pixels, between 0 and 1 (default = 0.3)
P2 = the smoothness of the image, greater than 0 (default = 2)
P3 = seed for random number generator

  The larger the P2 the larger the domains of nonzero pixels.

Example:

To use these images in connection with **fanbeamtomo, paralleltomo, semismictomo** and **semismicwavetomo**, use the commands:

```matlab
    im = phantomgallery(name,N,...);
x = im(:);
A = matrix generated, e.g., by paralleltomo
b = A*x;
```
**randkaczmarz**

**Purpose:**

The randomized Kaczmarz method.

**Synopsis:**

```matlab
[X info] = randkaczmarz(A,b,K)
[X info] = randkaczmarz(A,b,K,x0)
[X info] = randkaczmarz(A,b,K,x0,options)
```

**Algorithm:**

For arbitrary starting vector \( x^0 \in \mathbb{R}^n \) one iteration of the algorithm for `randkaczmarz` consists of \( m \) steps of the form:

\[
x \leftarrow x + \lambda \frac{b_{r(i)} - \langle a^{r(i)}, x \rangle}{\|a^{r(i)}\|_2^2} a^{r(i)}, \quad i = 1, \ldots, m,
\]

where the index \( r(i) \) is chosen from the set \( \{1, \ldots, m\} \) randomly with probability proportional with \( \|a^{i}\|_2^2 \).

**Description:**

The function implements the randomized Kaczmarz iterative method for solving the linear system \( A x = b \). The starting vector is \( x_0 \); if no starting vector is given then \( x_0 = 0 \) is used.

The numbers given in the vector \( K \) are iteration numbers, that specify which iterations are stored in the output matrix \( X \). If a stopping rule is selected (see below) and \( K = [] \), then \( X \) contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector \( K \) or by the stopping rule specified in the field `stoprule` in the struct `options`. If \( K \) is empty a stopping rule must be specified.

The relaxation parameter is given in the field `lambda` in the struct `options` as a constant. As default `lambda` is set to 1, since this corresponds to the original method.

The second output `info` is a vector with two elements. The first element is an indicator that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, 1 denotes that the NCP-rule stopped the iterations, and 2 denotes that the DP-rule stopped the iterations. The second element is `info` is the number of used iterations.

**Use of options:**

The following fields in `options` are used in this function:

- `options.lambda = c`, a constant satisfying \( 0 < c < 2 \). A warning is given if this requirement is estimated to be violated.
- `options.nonneg` Logical; if true then nonnegativity in enforced in each step.
- `options.box` Double; upper bound \( L \) in box constraint \([0, L]\) on pixel values.
- **options.damping** Double; a parameter $D$ to avoid division by very small row norms by adding $D \cdot \max\{\|a^i\|_2^2\}$ to $\|a^i\|_2^2$.

- **options.stoprule**
  - **options.stoprule.type**
    - **options.stoprule.type = 'none'**, where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    - **options.stoprule.type = 'NCP'**, where the optimal number of iterations $k_\star$ is chosen according to Normalized Cumulative Periodogram.
    - **options.stoprule.type = 'DP'**, where the stopping index is determined according to the discrepancy principle (DP).
    - **options.stoprule.taudelta** = $\tau \delta$, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule type DP.

**Example:**

Generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute 10 `randkaczmarz` iterations, and show the last iterate:

```matlab
[A b x] = paralleltomo(50,0:5:179,150);
e = randn(size(b)); e = e/norm(e);
b = b + 0.05*norm(b)*e;
X = randkaczmarz(A,b,1:10);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

**See also:**

cart, kaczmarz, symkaczmarz.

**References:**

rzr

Purpose:

Removes zero rows of A and the corresponding elements of b.

Synopsis:

\[
[A \ b] = \text{rzr}(A,b)
\]
\[
[A \ b] = \text{rzr}(A,b,Nthr)
\]

Description:

This function identifies zero rows of the coefficient matrix A and removes them. If a right-hand side b is present, the corresponding elements of b are also removed (b can be a matrix of several right-hand sides).

If a positive Nthr is given as the third argument, then all rows with less than or equal to Nthr nonzero elements are removed.

Use this function to “clean up” a discretized tomography problem. Zero rows do not contribute to the reconstruction. Rows with few nonzero elements correspond to pixels near the corners of the image, whose reconstructions are highly sensitive to noise.

Example:

Generate a small parallel-beam test problem and remove the zero rows from the system.

\[
[A \ b] = \text{paralleltomo}(16);
\]
\[
[A \ b] = \text{rzr}(A,b)
\]

Generate a small parallel-beam test matrix and remove all rows of A that have 5 or less nonzero elements.

\[
A = \text{paralleltomo}(16);
\]
\[
A = \text{rzr}(A,[],5)
\]
sart

Purpose:

The Simultaneous Algebraic Reconstruction Technique (SART) iterative method.

Synopsis:

[X info restart] = sart(A,b,K)
[X info restart] = sart(A,b,K,x0)
[X info restart] = sart(A,b,K,x0,options)

Algorithm:

For arbitrary \(x^0 \in \mathbb{R}^n\) the algorithm for sart takes the following form:

\[x^{k+1} = x^k + \lambda_k V^{-1} A^T W^{-1} (b - Ax^k),\]

where \(V = \text{diag} (\|a_i\|_1)\) and \(W = \text{diag} (\|a_j\|_1)\).

Description:

The function implements the SART (Simultaneous Algebraic Reconstruction Technique) iterative method for solving the linear system \(Ax = b\). The starting vector is \(x0\); if no starting vector is given then \(x0 = 0\) is used.

The numbers given in the vector \(K\) are iteration numbers, that specify which iterations are stored in the output matrix \(X\). If a stopping rule is selected (see below) and \(K = []\), then \(X\) contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector \(K\) or by the stopping rule specified in the field stoprule in the struct options. If \(K\) is empty a stopping rule must be specified.

The relaxation parameter is given in the field lambda in the struct options, either as a constant or as a string that determines the method to compute lambda. The spectral radius is \(\rho(V^{-1}A^TW^{-1}A) = 1\), and as default lambda is set to 1.

The second output info is a vector with two elements. The first element is an indicator, that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached 1 denotes that the NCP-rule stopped the iterations, 2 denotes that the DP-rule stopped the iterations, and 3 denote that the ME-rule stopped the iterations. The second element in info is the number of used iterations.

The struct restart, which can be given as output, contains in the field s1 the estimated largest singular value. restart also returns a vector containing the diagonal of the matrix \(W^{-1}\) in the field M and the diagonal of the matrix \(V^{-1}\) in the field T. The struct restart can also be given as input in the struct options, such that the program do not have to recompute the contained values. We recommend only to use this, if the user has good knowledge of MATLAB and is completely sure of the use of restart as input.
Use of options:

The following fields in options are used in this function:

- **options.lambda**:
  - **options.lambda = c**, where c is a constant, satisfying $0 < c < 2$. A warning is given if this requirement is estimated to be violated.
  - **options.lambda = 'line'**, where the line search method is used to compute the value for $\lambda_k$ in each iteration.
  - **options.lambda = 'psi1'**, where the method psi1 computes the values for $\lambda_k$ using the $\Psi_1$-based relaxation.
  - **options.lambda = 'psi1mod'**, where the method psi1mod computes the values for $\lambda_k$ using the modified $\Psi_1$-based relaxation with $\nu = 2$. The parameter $\nu$ can be changed in the code.
  - **options.lambda = 'psi2'**, where the method psi2 computes the values for $\lambda_k$ using the $\Psi_2$-based relaxation.
  - **options.lambda = 'psi2mod'**, where the method psi2mod computes the values for $\lambda_k$ using the modified $\Psi_2$-based relaxation with $\nu = 1.5$. The parameter $\nu$ can be changed in the code.

- **options.nonneg** Logical; if true then nonnegativity in enforced in each iteration.

- **options.restart**
  - **options.restart.M** = a vector containing the diagonal of $W^{-1}$.
  - **options.restart.T** = a vector containing the diagonal of $V^{-1}$.
  - **options.restart.s1** = 1.

- **options.stoprule**
  - **options.stoprule.type**
    - **options.stoprule.type = 'none'**, where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
    - **options.stoprule.type = 'NCP'**, where the optimal number of iterations $k_\nu$ is chosen according to the Normalized Cumulative Periodogram.
    - **options.stoprule.type = 'DP'**, where the stopping index is determined according to the discrepancy principle (DP).
    - **options.stoprule.type = 'ME'**, where the stopping index is determined according to the monotone error rule (ME).
  - **options.stoprule.taudelta** = $\tau \delta$, where $\delta$ is the noise level and $\tau$ is user chosen. This parameter is only needed for the stoprule types DP and ME.
Example:

Generate a “noisy” 50 \times 50 parallel beam tomography problem, compute 50 \texttt{sart} iterations, and show the last iterate:

\[
[A \ b \ x] = \text{paraleltomo}(50,0:5:179,150);
\]
\[
e = \text{randn(size(b))}; \ e = e/\text{norm}(e);
\]
\[
b = b + 0.05*\text{norm}(b)*e;
\]
\[
X = \text{sart}(A,b,1:50);
\]
\[
\text{imagesc(reshape(X(:,end),50,50))}
\]
\[
\text{colormap gray, axis image off}
\]

See also:

cav, cimmino, drop, landweber.

References:

seismictomo

Purpose:

Creates a two-dimensional seismic tomography test problem.

Synopsis:

\[
\begin{align*}
[A \ b \ x \ s \ p] &= \text{seismictomo}(N) \\
[A \ b \ x \ s \ p] &= \text{seismictomo}(N,s) \\
[A \ b \ x \ s \ p] &= \text{seismictomo}(N,s,p) \\
[A \ b \ x \ s \ p] &= \text{seismictomo}(N,s,p,isDisp)
\end{align*}
\]

Description:

This function creates a two-dimensional seismic tomography test problem.

- A two-dimensional domain illustrating a cross section of the subsurface is divided into \(N\) equally spaced intervals in both dimensions creating \(N^2\) cells.
- On the right boundary \(s\) sources are located and each source transmits waves to the \(p\) seismographs or receivers, which are scattered on the surface and on the left boundary.
- As default \(s = N\) sources and \(p = 2N\) receivers are chosen.

If the input \(isDisp\) is different from 0 then the function also creates an illustration of the problem with the used angles and rays etc. As default \(isDisp\) is 0.

The function returns a coefficient matrix \(A\) with the dimensions \(p \cdot s \times N^2\), the right hand side \(b\), and the phantom subsurface reshaped as a vector \(x\) with elements between 0 and 1. The figure below illustrates the subsurface created when \(N = 100\). In case the default values are used, the function also returns the used number of sources \(s\) and the used number of receivers \(p\).

Algorithm:

The matrix element \(a_{ij}\) is defined as the length of the \(i\)th ray through the \(j\)th cell, with \(a_{ij} = 0\) if ray \(i\) does not go through cell \(j\). The exact solution of the subsurface phantom is reshaped as a vector \(x\), and the \(i\)th element in the right hand side \(b\) is

\[
b_i = \sum_{j=1}^{N^2} a_{ij} x_j, \quad i = 1, \ldots, s \cdot p.
\]

Example:

Create a test problem and visualize the exact image:

\[
\begin{align*}
N &= 100; \ s = N; \ p = 2\cdot N; \\
[A \ b \ x] &= \text{seismictomo}(N,s,p); \\
\text{imagesc}(\text{reshape}(x,N,N)) \\
\text{colormap gray, axis image off}
\end{align*}
\]
See also:

fanbeamtomo, paralleltomo, seismicwavetomo.
seismicwavetomo

Purpose:
Seismic tomography problem without the ray assumption.

Synopsis:

\[ [A \ b \ x \ s \ p] = \text{seismicwavetomo}(N) \]
\[ [A \ b \ x \ s \ p] = \text{seismicwavetomo}(N,s) \]
\[ [A \ b \ x \ s \ p] = \text{seismicwavetomo}(N,s,p) \]
\[ [A \ b \ x \ s \ p] = \text{seismicwavetomo}(N,s,p,isDisp) \]

Description:
This function creates a two-dimensional seismic tomography test problem.

- A two-dimensional domain illustrating a cross section of the subsurface is divided into \( N \) equally spaced intervals in both dimensions creating \( N^2 \) cells.
- On the right boundary \( s \) sources are located and each source transmits waves to the \( p \) seismographs or receivers, which are scattered on the surface and on the left boundary.
- As default \( s = N \) sources and \( p = 2N \) receivers are chosen.
- If the frequency of the wave is very high then the path of the wave can be approximated by a straight-line ray, as done in seismictomo. For lower frequencies this is not a good approximation, and this function assumes instead that the wave travels within the first Fresnel zone. The wave frequency is determined by omega, and the higher the frequency the narrower the Fresnel zone (but note that \( A \) and \( b \) produced here do not converge to those from seismictomo as \( \omega \to \infty \)).

If the input isDisp is different from 0 then the function also creates an illustration of the problem with the used angles and rays etc. As default isDisp is 0.

The function returns a coefficient matrix \( A \) with the dimensions \( p \cdot s \times N^2 \), the right hand side \( b \), and the phantom subsurface reshaped as a vector \( x \) with elements between 0 and 1.

The figure below illustrates the subsurface created when \( N = 100 \). In case the default values are used, the function also returns the used number of sources \( s \) and the used number of receivers \( p \).

Algorithm:
The matrix element \( a_{ij} \) is defined as the length of the \( i \)th ray through the \( j \)th cell, with \( a_{ij} = 0 \) if ray \( i \) does not go through cell \( j \). The exact solution of the subsurface phantom is reshaped as a vector \( x \), and the \( i \)th element in the right hand side \( b \) is

\[ b_i = \sum_{j=1}^{N^2} a_{ij} x_j, \quad i = 1, \ldots, s \cdot p. \]
Example:

Create a test problem and visualize the exact image:

\[
N = 100; s = N; p = 2\cdot N;
[A \ b \ x] = 
\text{seismictomo}(N,s,p);
\text{imagesc}\left(\text{reshape}(x,N,N)\right)
\text{colormap gray, axis image off}
\]

See also:

fanbeamtomo, paralleltomo.

References:

symkaczmarz

Purpose:

The symmetric Kaczmarz iterative method.

Synopsis:

[X info] = symkaczmarz(A,b,K)
[X info] = symkaczmarz(A,b,K,x0)
[X info] = symkaczmarz(A,b,K,x0,options)

Algorithm:

For arbitrary starting vector $x^0 \in \mathbb{R}^n$ one iteration of the algorithm for symkaczmarz consists of a Kaczmarz sweep followed by a Kaczmarz sweep with the equations in reverse order, i.e., the following steps:

$$x \leftarrow x + \lambda_k \frac{b_i - \langle a_i, x \rangle}{\|a_i\|_2^2} a_i, \quad i = 1, \ldots, m - 1, m, m - 1, \ldots, 2.$$  

Description:

The function implements the symmetric Kaczmarz iterative method for solving the linear system $Ax = b$. The starting vector is $x_0$; if no vector is given then $x_0 = 0$ is used.

The numbers given in the vector $K$ are iteration numbers, that specify which iterations are stored in the output matrix $X$. If a stopping rule is selected (see below) and $K = []$, then $X$ contains the last iterate only.

The maximum number of iterations is determined either by the maximum number in the vector $K$ or by the stopping rule specified in the field stoprule in the struct options. If $K$ is empty a stopping rule must be specified.

The relaxation parameter is given in the field lambda in the struct options, either as a constant or as a string that determines the method to compute lambda. As default lambda is set to 0.25.

The second output info is a vector with two elements. The first element is an indicator, that denotes why the iterations were stopped. The number 0 denotes that the iterations were stopped because the maximum number of iterations were reached, 1 denotes that the NCP-rule stopped the iterations, and 2 denotes that the DP-rule stopped the iterations. The second element in info is the number of used iterations.

Use of options:

The following fields in options are used in this function:

- options.lambda:
  - options.lambda = c, where c is a constant, satisfying $0 < c < 2$. A warning is given if this requirement is estimated to be violated.
  - options.lambda = 'psi1', where the method psi1 computes the values for $\lambda_k$ using the $\Psi_1$-based relaxation.


- `options.lambda = 'psi2'`, where the method `psi2` computes the values for $\lambda_k$ using the $\Psi_2$-based relaxation.

- `options.nonneg` Logical; if true then nonnegativity in enforced in each step.

- `options.box` Double; upper bound $L$ in box constraint $[0, L]$ on pixel values.

- `options.damping` Double; a parameter $D$ to avoid division by very small row norms by adding $D \cdot \max_i \{\|a^i\|_2^2\}$ to $\|a^i\|_2^2$.

- `options.stoprule`
  - `options.stoprule.type = 'none'`, where no stopping rule is given and only the maximum number of iterations is used to stop the algorithm. This choice is default.
  - `options.stoprule.type = 'NCP'`, where the optimal number of iterations $k_*$ is chosen according to Normalized Cumulative Periodogram.
  - `options.stoprule.type = 'DP'`, where the stopping index is determined according to the discrepancy principle (DP).
  - `options.stoprule.taudelta = \tau \delta`, where $\delta$ is the noise level and $\tau$ is user-chosen. This parameter is only needed for the stoprule type DP.

**Example:**

Generate a “noisy” $50 \times 50$ parallel beam tomography problem, compute 10 `symkaczmarz` iterations, and show the last iterate:

```matlab
[A b x] = paralleltomo(50,0:5:179,150);
e = randn(size(b)); e = e/norm(e);
b = b + 0.05*norm(b)*e;
X = symkaczmarz(A,b,1:10);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

See also: `cart`, `kaczmarz`, `randkaczmarz`.

**References:**

trainDPME

Purpose:

Training strategy to estimate the best parameter when the discrepancy principle or the monotone error rule is used as stopping rule.

Synopsis:

\[
\text{tau} = \text{trainlambda}(A, b, x_{\text{exact}}, \text{method}, \text{type}, \text{delta}, s)
\]

\[
\text{tau} = \text{trainlambda}(A, b, x_{\text{exact}}, \text{method}, \text{type}, \text{delta}, s, \text{options})
\]

Description:

This function implements the training strategy for estimation of the parameter \( \tau \), when using the discrepancy principle or the monotone error rule as stopping rule. From the test solution \( x_{\text{exact}} \) and the corresponding noise free right-hand side \( b \), we generate \( s \) noisy samples with noise level \( \text{delta} \), i.e., we add a noise vector \( e \) with \( \|e\|_2 = \text{delta} \). From each sample the solutions for the given method \( \text{method} \) are calculated, and according to which type of stopping rule is chosen in \( \text{type} \) an estimate of \( \text{tau} \) is calculated and returned.

A default maximum number of iterations is chosen for the SIRT methods to be 1000 and for the ART methods to 100. If this is not enough it can be changed in line 74 for the SIRT methods and in line 87 for the ART methods.

Use of options:

The following fields in \( \text{options} \) are used in this function.

- \( \text{options.lambda} \): See the chosen method \( \text{method} \) for the choices of this parameter.
- \( \text{options.restart} \): Only available when \( \text{method} \) is a SIRT method. See the specific method for correct use.
- \( \text{options.w} \): If the chosen method allows weights this parameter can be set.

Example:

Generate a “noisy” 50 \( \times \) 50 parallel beam tomography problem. Then the parameter \( \text{tau} \) is found using training for DP, and this parameter is used with DP to stop the iterations. Finally the last iterate is shown.

\[
[A \ b \ x] = \text{paralleltomo}(50,0:5:179,150);
\text{delta} = 0.05;
\text{tau} = \text{trainDPME}(A, b, x, @\text{cimmino}, '\text{ME}', \text{delta}, 20);
\text{e} = \text{randn(size(b))}; \text{e} = \text{e/norm(e)};
\text{b} = \text{b} + \text{delta*norm(b)}*\text{e};
\text{options.stoprule.type} = '\text{ME}';
\text{options.stoprule.taudelta} = \text{tau*delta};
[X, \text{info}] = \text{cimmino}(A, b, 200, [], \text{options});
\text{imagesc(reshape(X(:,end),50,50))}
\text{colormap gray, axis image off}
\]
See also:
cav, cimmino, drop, kaczmarz, landweber, randkaczmar, sart, symkaczmarz.

References:
trainLambdaART

Purpose:

Strategy to find the best constant relaxation parameter $\lambda$ for a given ART method.

Synopsis:

\[
\lambda = \text{trainLambdaART}(A, b, x_{\text{exact}}, \text{method}) \\
\lambda = \text{trainLambdaART}(A, b, x_{\text{exact}}, \text{method}, k_{\text{max}})
\]

Description:

This function implements a training strategy for finding the optimal constant relaxation parameter $\lambda$ for a given ART method that solves the linear system $Ax = b$, as given as a function handle in method. The training strategy builds on a two-part strategy.

1. In the first part the resolution limit is calculated using $k_{\text{max}}$ iterations of the ART method. If $k_{\text{max}}$ is not given or empty, the default value is 100. The first part of the strategy is to determine the resolution limit for the a specific value of $\lambda$.

2. The second part is a modified version of a golden section search in which the optimal value of $\lambda$ is found within the convergence interval of the specified iterative method.

The method returns the optimal value in the output $\lambda$.

Example:

Generate a “noisy" 50 $\times$ 50 parallel beam tomography problem, train to find the optimal value of $\lambda$ for the ART method kaczmarz, and use the found value to perform 10 iterations of this method. Finally the last iterate is shown:

\[
[B \ b \ x] = \text{paralleltomo}(50,0:5:179,150); \\
e = \text{randn(size}(b)); \ e = e/\text{norm}(e); \\
b = b + 0.05*\text{norm}(b)*e; \\
\lambda = \text{trainLambdaART}(A,b,x,@kaczmarz); \\
\text{options.lambda} = \lambda; \\
X = \text{kaczmarz}(A,b,1:10,[],\text{options}); \\
\text{imagesc(reshape}(X(:,\text{end}),50,50)) \\
\text{colormap gray, axis image off}
\]

See also:

trainLambdaSIRT
trainLambdaSIRT

Purpose:

Strategy to find best constant relaxation parameter $\lambda$ for a given SIRT method.

Synopsis:

$$\lambda = \text{trainLambdaSIRT}(A,b,x_{\text{exact}},\text{method})$$
$$\lambda = \text{trainLambdaSIRT}(A,b,x_{\text{exact}},\text{method},k_{\text{max}})$$
$$\lambda = \text{trainLambdaSIRT}(A,b,x_{\text{exact}},\text{method},k_{\text{max}},\text{options})$$

Description:

This function implements a training strategy for finding the optimal constant relaxation parameter $\lambda$ for a given SIRT method that solves the linear system $Ax = b$, as given as a function handle in $\text{method}$. The training strategy builds on a two part strategy.

1. In the first step the resolution limit is calculated using $k_{\text{max}}$ iterations of the SIRT method. If $k_{\text{max}}$ is not given or empty, the default value is 1000. To determine the resolution limit the default value of $\lambda$ is used together with the contents of $\text{options}$.

2. The second part is a modified version of a golden section search in which the optimal value of $\lambda$ is found within the convergence interval of the specified iterative method.

The method returns the optimal value in the output $\lambda$.

Use of $\text{options}$:

The following fields in $\text{options}$ are used in this function.

- $\text{options.restart}$: See the specific method for correct use.
- $\text{options.w}$: If the chosen method allows weights this parameter can be set.

Example:

Generate a “noisy” 50 × 50 parallel beam tomography problem, train to find the optimal value of $\lambda$ for the SIRT method $\text{cimmino}$, and use the found value to perform 50 iterations of this method. Finally the last iterate is shown:

$$[A \ b \ x] = \text{paralleltomo}(50,0:5:179,150);$$
$$e = \text{randn(size(b))}; e = e/\text{norm(e)};$$
$$b = b + 0.05*\text{norm(b)}*e;$$
$$\lambda = \text{trainLambdaSIRT}(A,b,x,\text{@cimmino});$$
$$\text{options.lambda} = \lambda;$$
$$X = \text{cimmino}(A,b,1:50,[],\text{options});$$
$$\text{imagesc(reshape(X(:,end),50,50))}$$
$$\text{colormap gray, axis image off}$$

See also:

trainLambdaART