AIR Tools

A MATLAB Package of Algebraic Iterative Reconstruction Techniques

Version 1.2 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

October 2014

Abstract

This collection of MATLAB software contains implementations of several Algebraic Iterative Reconstruction methods for discretizations of inverse 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: Algebraic Reconstruction Techniques (ART) and Simultaneous Iterative Reconstruction Techniques (SIRT). 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:

- P. C. Hansen and M. Saxild-Hansen, *AIR Tools – A MATLAB Package of Algebraic Iterative Reconstruction Techniques*, Journal of Computational and Applied Mathematics, 236 (2012), pp. 2167–2178; doi:10.1016/j.cam.2011.09.039.

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 jth column of A, a^i is the transposed of the ith 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 Plešinger, and Nicolai Riis helped to prepare Version 1.2 with code and revisions.

Revisions Since Version 1.0

- Functions `fanbeamtomo` and `paralleltomo` now return the correct variable `d`.
- Function `rzr` allows input `b` with multiple columns.
- All iterative methods now include box constraint.
- All ART methods now include 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 $1/2(r^k)^T(r^{k+1} + r^k)/\|r^k\|_2$.
- The default value for `lambda` was changed to `1` in `kaczmarz` and `symkaczmarz`.
- The default value for `lambda` was changed to $1.9/\|A^T M A\|_2$ in all SIRT methods.
- The following new functions were added:
  - `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

Iterative ART Methods	
`cart`	Columnwise version of Kaczmarz's method
`kaczmarz`	Kaczmarz's method, aka the Algebraic Reconstruction Technique (ART)
`randkaczmarz`	The randomized Kaczmarz method
`symkaczmarz`	The symmetric Kaczmarz method

Iterative SIRT Methods	
`cav`	Component Averaging (CAV) method
`cimmino`	Cimmino's method
`drop`	Diagonally Relaxed Orthogonal Projections (DROP) method
`landweber`	Landweber's method
`sart`	Simultaneous Algebraic Reconstruction Technique (SART)

Training Routines	
`trainDPME`	Training strategy to find the best parameter τ when discrepancy principle or monotone error rule is used as stopping rule
`trainLambdaART`	Training strategy to find the best constant relaxation parameter λ for a given ART method
`trainLambdaSIRT`	Training strategy to find the best constant relaxation parameter λ for a given SIRT method

Test Problems	
`fanbeamtomo`	Creates a 2-D fan-beam tomography problem
`paralleltomo`	Creates a 2-D parallel-beam tomography problem
`phantomgallery`	A collection of different 2D phantoms
`seismictomo`	Creates a 2-D seismic tomography problem
`seismicwavetomo`	Similar to `seismictomo` but without a ray assumption

Demo Scripts	
`ARTdemo`	Illustrates the simple use of the ART methods
`nonnegdemo`	Illustrates the use of nonnegativity constraints
`SIRTdemo`	Illustrates the simple use of the SIRT methods
`trainingdemo`	Illustrates the use of the training routines as pre-processors for the SIRT and the ART methods

Auxiliary Routines	
`calczeta`	Calculates a specific root of a certain polynomial
`rzr`	Removes zero rows from A and corresponding elements of b

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:

```
[X,info,restart] = landweber(A,b,1:k);          % First call.
% Other code lines here.
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:

```
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.

calczeta

Purpose:

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

Synopsis:

```
z = calczeta(k)
```

Description:

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

$$(2k-1)z^{k-1} - (z^{k-2} + ... + z + 1) = 0, \qquad k \geq 2.$$

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 `calczeta` is used in the functions `cav`, `cimmino`, `drop`, `landweber`, `sart`, and `symkaczmarz`.

Example:

Calculate the roots for `k` from 2 up to 100 and plot the found roots.

```
k = 2:100;
z = calczeta(k);
figure, plot(k,z,'bo')
```

See also:

`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 `cart` consists of the following steps:

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

where $a_{:j}$ is the jth 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 putput 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 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` $=$ `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 $D \cdot \max_i\{\|a_{:j}\|_2^2\}$ 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_* 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 δ is the noise level and τ is user-chosen. This parameter is only needed for the stoprule type `DP`.

Example:

Generate a "noisy" 50×50 parallel beam tomography problem, compute 10 `cart` iterations, and show the last iterate:

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

Algorithm:

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

$$x^{k+1} = x^k + \lambda_k \, \mathtt{A}^T D_S(\mathtt{b} - \mathtt{A}x^k), \qquad D_S = \mathrm{diag}\left(\frac{w_i}{\sum_{j=1}^n s_j a_{ij}^2}\right),$$

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 $\mathtt{A}\,x = \mathtt{b}$. The starting vector is `x0`; if no starting vector is given then $\mathtt{x0} = 0$ 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 $\mathtt{K} = [\,]$, 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 $D_S^{1/2}\mathtt{A}$.

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 $0 < \texttt{c} < 2/\tilde{\sigma}_1^2$. 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 λ_k in each iteration.
  - `options.lambda = 'psi1'`, where the method `psi1` computes the values for λ_k using the Ψ_1-based relaxation.
  - `options.lambda = 'psi1mod'`, where the method `psi1mod` computes the values for λ_k using the modified Ψ_1-based relaxation with $\nu = 2$. The parameter ν can be changed in the code.
  - `options.lambda = 'psi2'`, where the method `psi2` computes the values for λ_k using the Ψ_2-based relaxation.
  - `options.lambda = 'psi2mod'`, where the method `psi2mod` computes the values for λ_k using the modified Ψ_2-based relaxation with $\nu = 1.5$. The parameter ν 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.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_* is chosen according to Normalized Cumulative Periodogram.
    - `options.stoprule.type = 'DP'`, where the stopping index k_* is determined according to the discrepancy principle (DP).
    - `options.stoprule.type = 'ME'`, where the stopping index k_* is determined according to the monotone error rule (ME).
  - `options.stoprule.taudelta` $= \tau\delta$, where δ is the noise level and τ 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" 50×50 parallel beam tomography problem, compute 50 `cav` iterations, and show the last iterate:

```
[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:

1. Y. Censor, D. Gordon, and R. Gordon, *Component averaging: An efficient iterative parallel algorithm for large sparse unstructured problems*, Parallel Computing, 27 (2001), pp. 777–808.

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:

$$x^{k+1} = x^k + \lambda_k \, \mathtt{A}^T D(b - \mathtt{A}x^k), \qquad D = \frac{1}{m}\mathrm{diag}\left(\frac{w_i}{\|\mathtt{A(i,:)}\|_2^2}\right),$$

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

Description:

The function implements Cimmino's iterative projection method for solving linear systems $\mathtt{A}\,x = \mathtt{b}$. The starting vector is `x0`; if no starting vector is given, then $\mathtt{x0} = 0$ is used.

The numbers given in the vector `K` are iteration numbers that specify which iterations are stored in the output matric `K`. If a stopping rule us selected (see below) and $\mathtt{K} = [\,]$, 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 $D^{1/2}\mathtt{A}$.

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 $0 < \texttt{c} < 2/\tilde{\sigma}_1^2$. 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 λ_k in each iteration.
  - `options.lambda = 'psi1'`, where the method `psi1` computes the values for λ_k using the Ψ_1-based relaxation.
  - `options.lambda = 'psi1mod'`, where the method `psi1mod` computes the values for λ_k using the modified Ψ_1-based relaxation with $\nu = 2$. The parameter ν can be changed in the code.
  - `options.lambda = 'psi2'`, where the method `psi2` computes the values for λ_k using the Ψ_2-based relaxation.
  - `options.lambda = 'psi2mod'`, where the method `psi2mod` computes the values for λ_k using the modified Ψ_2-based relaxation with $\nu = 1.5$. The parameter ν 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.M` = a vector with the diagonal of M.
  - `options.restart.s1` $= \tilde{\sigma}_1$, 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_* is chosen according to Normalized Cumulative Periodogram.
    - `options.stoprule.type = 'DP'`, where the stopping index k_* is determined according to the discrepancy principle (DP).
    - `options.stoprule.type = 'ME'`, where the stopping index k_* is determined according to the monotone error rule (ME).
  - `options.stoprule.taudelta` $= \tau\delta$, where δ is the noise level and τ 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" 50×50 parallel beam tomography problem, compute 50 `cimmino` iterations, and show the last iterate:

```
[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 = cimmino(A,b,1:50);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

See also:

`cav`, `drop`, `landweber`, `sart`.

References:

1. G. Cimmino, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, La Ricerca Scientifica, XVI, Series II, Anno IX, 1 (1938), pp. 326–333.

2. C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.

drop

Purpose:

Diagonally Relaxed Orthogonal Projections (DROP) iterative method.

Synopsis:

```
[X info restart] = drop(A,b,K)
[X info restart] = drop(A,b,K,x0)
[X info restart] = 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:

$$x^{k+1} = x^k + \lambda_k S^{-1} \mathtt{A}^T M(b - \mathtt{A}x^k), \qquad M = \operatorname{diag}\left(\frac{w_i}{\|\mathtt{A}(\mathtt{i},:)\|_2^2}\right),$$

where $S = \operatorname{diag}(s_j)$ and s_j is the number of nonzero elements in column j of `A` for $\mathtt{i} = 1, \ldots, \mathtt{m}$. 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 $\mathtt{A}\,x = \mathtt{b}$. The starting vector is `x0`; if no starting vector is given, then $\mathtt{x0} = 0$ 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 $\mathtt{K} = [\,]$, 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 $1/\tilde{\rho}$, where $\tilde{\rho}$ is an estimate of the spectral radius of $S^{-1}\mathtt{A}^T M\,\mathtt{A}$.

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 $0 < \texttt{c} < 2/\tilde{\rho}$. 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 λ_k in each iteration.
  - `options.lambda = 'psi1'`, where the method `psi1` computes the values for λ_k using the Ψ_1-based relaxation.
  - `options.lambda = 'psi1mod'`, where the method `psi1mod` computes the values for λ_k using the modified Ψ_1-based relaxation with $\nu = 2$. The parameter ν can be changed in the code.
  - `options.lambda = 'psi2'`, where the method `psi2` computes the values for λ_k using the Ψ_2-based relaxation.
  - `options.lambda = 'psi2mod'`, where the method `psi2mod` computes the values for λ_k using the modified Ψ_2-based relaxation with $\nu = 1.5$. The parameter ν 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.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 $\tilde{\sigma}_1 = \sqrt{\tilde{\rho}}$.
- `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_* 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.taudelta` $= \tau\delta$, where δ is the noise level and τ 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" 50×50 parallel beam tomography problem, compute 50 `drop` iterations, and show the last iterate:

```
[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 = drop(A,b,1:50);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

See also:

`cav`, `cimmino`, `landweber`, `sart`.

References:

1. Y. Censor, T. Elfving, G. Herman, and T. Nikazad, *On diagonally relaxed orthogonal projection methods*, SIAM J. Sci. Comput., 30 (2007/08), pp. 473–504.

fanbeamtomo

Purpose:

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

Synopsis:

```
[A b x theta p R w] = fanbeamtomo(N)
[A b x theta p R w] = fanbeamtomo(N,theta)
[A b x theta p R w] = fanbeamtomo(N,theta,p)
[A b x theta p R w] = fanbeamtomo(N,theta,p,R)
[A b x theta p R w] = fanbeamtomo(N,theta,p,R,w)
[A b x theta p R w] = fanbeamtomo(N,theta,p,R,w,isDisp)
```

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 $\texttt{N}^2$ 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 $\texttt{round}(\sqrt{2}\texttt{N})$. 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 `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 $\texttt{length(theta)} \cdot \texttt{p} \times \texttt{N}^2$, 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 $\texttt{N} = 100$. 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 ith ray through the jth 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 ith element in the right hand side `b` is

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

Example:

Create a test problem and visualize the exact image:

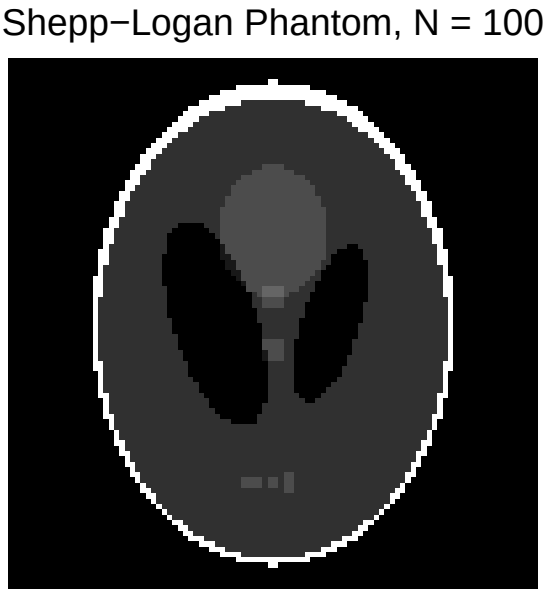
```
N = 64; theta = 0:5:359; p = 2*N; R = 2;
[A b x] = fanbeamtomo(N,theta,p,R);
imagesc(reshape(x,N,N))
colormap gray, axis image off
```

See also:

`paralleltomo`, `seismictomo`.

References:

1. A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*, SIAM, Philadelphia, 2001.



kaczmarz

Purpose:

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

Synopsis:

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

Algorithm:

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

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

Description:

The function implements Kaczmarz's iterative method for solving the linear system $\mathtt{A}\,x = \mathtt{b}$. The starting vector is `x0`; if no starting vector is given then $\mathtt{x0} = 0$ is used.

The numbers given in the vector `K` are iteration numbers, that specify which iterations are stored in the putput matrix `X`. If a stopping rule is selected (see below) and $\mathtt{K} = [\,]$, 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 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` $= \mathtt{c}$, a constant satisfying $0 < \mathtt{c} < 2$. 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 $D \cdot \max_i\{\|a^i\|_2^2\}$ 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 δ is the noise level and τ is user-chosen. This parameter is only needed for the stoprule type `DP`.

Example:

Generate a "noisy" 50×50 parallel beam tomography problem, compute 10 `kaczmarz` iterations, and show the last iterate:

```
[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:

1. G. T. Herman, *Fundamentals of Computerized Tomography, Image Reconstruction from Projections*, Springer, New York, 2009.

2. S. Kaczmarz, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bulletin de l'Académie Polonaise des Sciences et Lettres, A35 (1937), pp. 355–357.

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 $x^0 \in \mathbb{R}^{\mathtt{n}}$ the algorithm for `landweber` takes the following form:

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

Description:

The function implements the classical Landweber iterative method for solving the linear system $\mathtt{A}\,x = \mathtt{b}$. The starting vector is `x0`; if no starting vector is given then $\mathtt{x0} = 0$ 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 $\mathtt{K} = [\,]$, 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 $0 < \texttt{c} < 2/\tilde{\sigma}_1^2$. 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 λ_k in each iteration.
- `options.lambda = 'psi1'`, where the method `psi1` computes the values for λ_k using the Ψ_1-based relaxation.
- `options.lambda = 'psi1mod'`, where the method `psi1mod` computes the values for λ_k using the modified Ψ_1-based relaxation with $\nu = 2$. The parameter ν can be changed in the code.
- `options.lambda = 'psi2'`, where the method `psi2` computes the values for λ_k using the Ψ_2-based relaxation.
- `options.lambda = 'psi2mod'`, where the method `psi2mod` computes the values for λ_k using the modified Ψ_2-based relaxation with $\nu = 1.5$. The parameter ν 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` $= \tilde{\sigma}_1$, where $\tilde{\sigma}_1$ 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_* is chosen according to Normalized Cumulative Periodogram.
    - `options.stoprule.type = 'DP'`, where the stopping index k_* is determined according to the discrepancy principle (DP).
    - `options.stoprule.type = 'ME'`, where the stopping index k_* is determined according to the monotone error rule (ME).

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

Example:

We generate a "noisy" 50×50 parallel beam tomography problem, compute 50 `landweber` iterations, and show the last iterate:

```
[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:

1. L. Landweber, *An iteration formula for Fredholm integral equations of the first kind*, American Journal of Mathematics, 73 (1951), pp. 615–624.

paralleltomo

Purpose:

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

Synopsis:

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

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 $\texttt{length(theta)} \cdot \texttt{p} \times \texttt{N}^2$, 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 $\texttt{N} = 100$. 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 ith ray through the jth 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 ith element in the right hand side `b` is

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

Example:

Create a test problem and visualize the exact image:

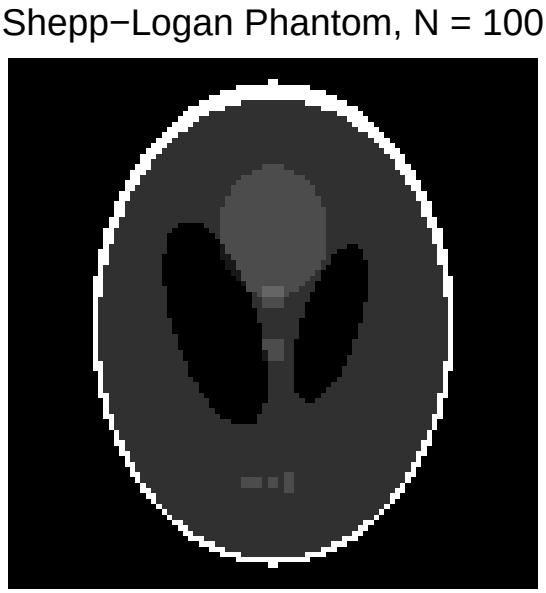
```
N = 64; theta = 0:5:179; p = 2*N;
[A b x] = paralleltomo(N,theta,p);
imagesc(reshape(x,N,N))
colormap gray, axis image off
```

See also:

`fanbeamtomo`, `seismictomo`.

References:

1. A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*, SIAM, Philadelphia, 2001.



phantomgallery

Purpose:

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

Synopsis:

```
im = paralleltomo(name,N)
im = paralleltomo(name,N,P1)
im = paralleltomo(name,N,P1,P2)
im = paralleltomo(name,N,P1,P2,P3)
```

Description:

This function creates several different two-dimensional phantoms for use in tomography test problem. The size is `N` × `N` and the pixel values are between 0 and 1. The type of phantom is specified by `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

  `im = phantomgallery('smooth',N,P1)`
  `P1` = 1, 2, 3 or 4 defines four different smooth functions (default = 4)

  The image is constructed by adding four different Gaussian functions.

- `name = 'binary'`: a random image with binary pixel values arranged in domains

  `im = phantomgallery('binary',N,P1)`
  `P1` = seed for random number generator

  The image is dominated by horizontal structures.

- `name = 'threephases'`: a random image with pixel values 0, 0.5, 1 arranged in domains

  `im = phantomgallery('threephases',N,P1,P2)`
  `P1` controls the number of size of the domains (default = `100`)
  `P2` = seed for random number generator

  The image is a model of a three-phase object.

- `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

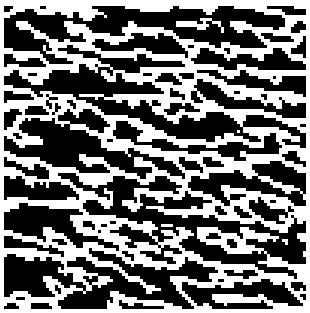
The larger the `P2` the larger the domains of nonzero pixels.

Example:

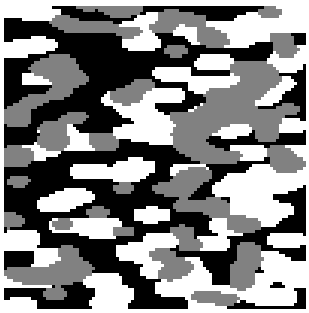
To use these images on connection with `fanbeamtomo`, `paralleltomo`, `semismictomo` and `semismicwavetomo`, use the commands:

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

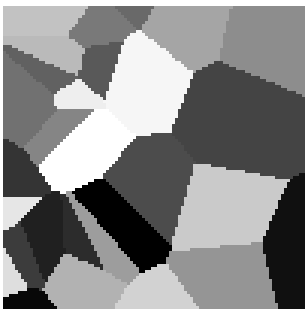
'binary'



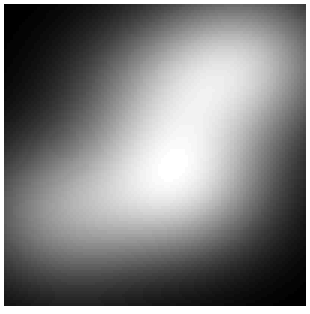
'threephases'



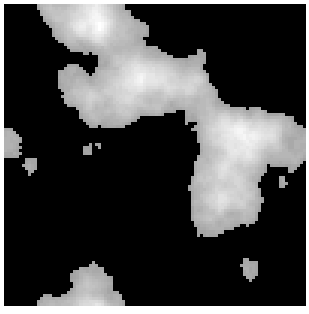
'grains'



'smooth'



'ppower'



randkaczmarz

Purpose:

The randomized Kaczmarz method.

Synopsis:

```
[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:

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

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 `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` 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 $D \cdot \max_i\{\|a^i\|_2^2\}$ 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 δ is the noise level and τ is user-chosen. This parameter is only needed for the stoprule type `DP`.

Example:

Generate a "noisy" 50×50 parallel beam tomography problem, compute 10 `randkaczmarz` iterations, and show the last iterate:

```
[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:

1. T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm for linear systems with exponential convergence*, J. Fourier Analysis Appl., 15 (2009), pp. 262–278.

rzr

Purpose:

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

Synopsis:

```
[A b] = rzr(A,b)
[A b] = 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] = paralleltomo(16);
[A b] = rzr(A,b)
```

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

```
A = paralleltomo(16);
A = 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:

$$x^{k+1} = x^k + \lambda_k V^{-1} \mathtt{A}^T W^{-1} (\mathtt{b} - \mathtt{A} x^k),$$

where $V = \operatorname{diag}\left(\|a^i\|_1\right)$ and $W = \operatorname{diag}\left(\|a_j\|_1\right)$.

Description:

The function implements the SART (Simultaneous Algebraic Reconstruction Technique) iterative method for solving the linear system $\mathtt{A}\,x = \mathtt{b}$. The starting vector is `x0`; if no starting vector is given then $\mathtt{x0} = 0$ 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 $\mathtt{K} = [\,]$, 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 $\rho(V^{-1}\mathtt{A}^T W^{-1}\mathtt{A}) = 1$, 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 λ_k in each iteration.
  - `options.lambda = 'psi1'`, where the method `psi1` computes the values for λ_k using the Ψ_1-based relaxation.
  - `options.lambda = 'psi1mod'`, where the method `psi1mod` computes the values for λ_k using the modified Ψ_1-based relaxation with $\nu = 2$. The parameter ν can be changed in the code.
  - `options.lambda = 'psi2'`, where the method `psi2` computes the values for λ_k using the Ψ_2-based relaxation.
  - `options.lambda = 'psi2mod'`, where the method `psi2mod` computes the values for λ_k using the modified Ψ_2-based relaxation with $\nu = 1.5$. The parameter ν 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_* 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 δ is the noise level and τ is user chosen. This parameter is only needed for the stoprule types `DP` and `ME`.

Example:

Generate a "noisy" 50×50 parallel beam tomography problem, compute 50 `sart` iterations, and show the last iterate:

```
[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 = sart(A,b,1:50);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

See also:

`cav`, `cimmino`, `drop`, `landweber`.

References:

1. A. H. Andersen and A. C. Kak, *Simultaneous algebraic reconstruction technique (SART): A superior implementation of the ART algorithm*, Ultrasonic Imaging, 6 (1984), pp. 81–94.

seismictomo

Purpose:

Creates a two-dimensional seismic tomography test problem.

Synopsis:

```
[A b x s p] = seismictomo(N)
[A b x s p] = seismictomo(N,s)
[A b x s p] = seismictomo(N,s,p)
[A b x s p] = seismictomo(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 $\texttt{N}^2$ 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 $\texttt{s} = \texttt{N}$ sources and $\texttt{p} = 2\texttt{N}$ 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 $\texttt{p} \cdot \texttt{s} \times \texttt{N}^2$, 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 $\texttt{N} = 100$. 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 ith ray through the jth 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 ith element in the right hand side `b` is

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

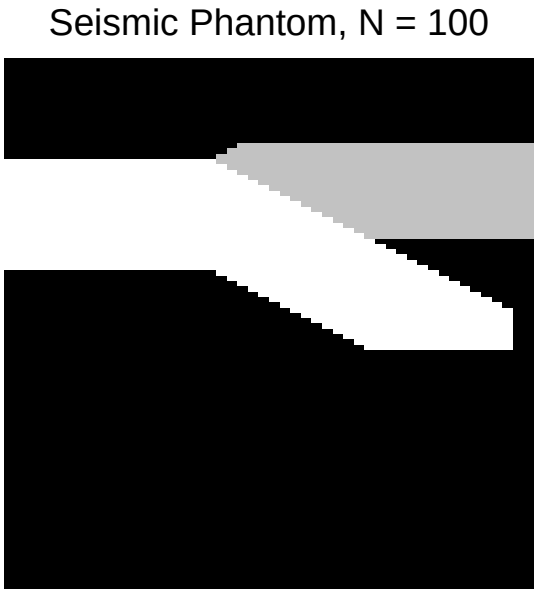
Example:

Create a test problem and visualize the exact image:

```
N = 100; s = N; p = 2*N;
[A b x] = seismictomo(N,s,p);
imagesc(reshape(x,N,N))
colormap gray, axis image off
```

See also:

`fanbeamtomo`, `paralleltomo`, `seismicwavetomo`.



seismicwavetomo

Purpose:

Seismic tomography problem without the ray assumption.

Synopsis:

```
[A b x s p] = seismicwavetomo(N)
[A b x s p] = seismicwavetomo(N,s)
[A b x s p] = seismicwavetomo(N,s,p)
[A b x s p] = 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 $\texttt{N}^2$ 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 $\texttt{s} = \texttt{N}$ sources and $\texttt{p} = 2\texttt{N}$ 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 approxmation, 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 $\texttt{omega} \to \infty$).

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 $\texttt{p} \cdot \texttt{s} \times \texttt{N}^2$, 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 $\texttt{N} = 100$. 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 ith ray through the jth 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 ith element in the right hand side `b` is

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

Example:

Create a test problem and visualize the exact image:

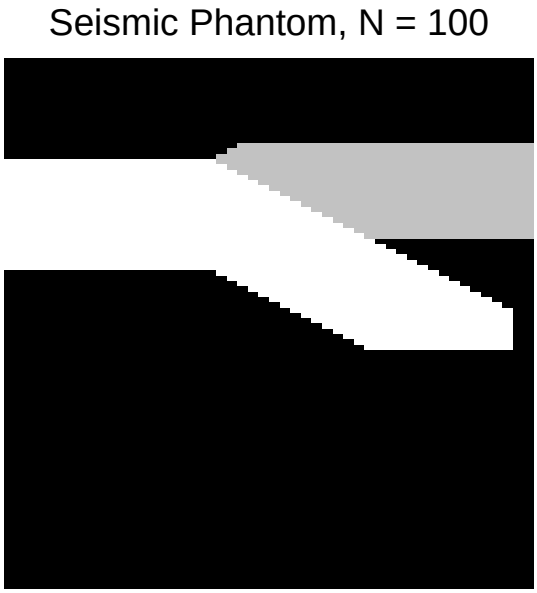
```
N = 100; s = N; p = 2*N;
[A b x] = seismictomo(N,s,p);
imagesc(reshape(x,N,N))
colormap gray, axis image off
```

See also:

`fanbeamtomo`, `paralleltomo`.

References:

1. J. M. Jensen, B. H. Jacobsen, and J. Christensen-Dalsgaard, *Sensitivity kernels for time-distance inversion*, Solar Physics, 192 (2000), pp. 231–239.



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:

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

Description:

The function implements the symmetric Kaczmarz iterative method for solving the linear system $\mathtt{A}\,x = \mathtt{b}$. The starting vector is `x0`; if no vector is given then $\mathtt{x0} = 0$ 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 $\mathtt{K} = [\,]$, 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 $0 < \mathtt{c} < 2$. A warning is given if this requirement is estimated to be violated.
  - `options.lambda = 'psi1'`, where the method `psi1` computes the values for λ_k using the Ψ_1-based relaxation.

    - `options.lambda = 'psi2'`, where the method `psi2` computes the values for λ_k using the Ψ_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`
        - `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 δ is the noise level and τ is user-chosen. This parameter is only needed for the stoprule type `DP`.

Example:

Generate a "noisy" 50×50 parallel beam tomography problem, compute 10 `symkaczmarz` iterations, and show the last iterate:

```
[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:

1. Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163.

trainDPME

Purpose:

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

Synopsis:

```
tau = trainlambda(A,b,x_exact,method,type,delta,s)
tau = trainlambda(A,b,x_exact,method,type,delta,s,options)
```

Description:

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

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

Example:

Generate a "noisy" 50×50 parallel beam tomography problem. Then the parameter `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] = paralleltomo(50,0:5:179,150);
delta = 0.05;
tau = trainDPME(A,b,x,@cimmino,'ME',delta,20);
e = randn(size(b)); e = e/norm(e);
b = b + delta*norm(b)*e;
options.stoprule.type = 'ME';
options.stoprule.taudelta = tau*delta;
[X info] = cimmino(A,b,200,[],options);
imagesc(reshape(X(:,end),50,50))
colormap gray, axis image off
```

See also:

`cav`, `cimmino`, `drop`, `kaczmarz`, `landweber`, `randkaczmar`, `sart`, `symkaczmarz`.

References:

1. T. Elfving and T. Nikazad, *Stopping rules for Landweber-type iteration*, Inverse Problems, 23 (2007), pp. 1417–1432.

trainLambdaART

Purpose:

Strategy to find the best constant relaxation parameter λ for a given ART method.

Synopsis:

```
lambda = trainLambdaART(A,b,x_exact,method)
lambda = trainLambdaART(A,b,x_exact,method,kmax)
```

Description:

This function implements a training strategy for finding the optimal constant relaxation parameter λ for a given ART method that solves the linear system $\mathtt{A}\,x = \mathtt{b}$, 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 `kmax` iterations of the ART method. If `kmax` 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 λ.
2. The second part is a modified version of a golden section search in which the optimal value of λ 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" 50×50 parallel beam tomography problem, train to find the optimal value of λ for the ART method `kaczmarz`, and use the found value to perform 10 iterations of this method. Finally the last iterate is shown:

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

See also:

`trainLambdaSIRT`

trainLambdaSIRT

Purpose:

Strategy to find best constant relaxation parameter λ for a given SIRT method.

Synopsis:

```
lambda = trainLambdaSIRT(A,b,x_exact,method)
lambda = trainLambdaSIRT(A,b,x_exact,method,kmax)
lambda = trainLambdaSIRT(A,b,x_exact,method,kmax,options)
```

Description:

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

1. In the first step the resolution limit is calculated using `kmax` iterations of the SIRT method. If `kmax` is not given or empty, the default value is 1000. To determine the resolution limit the default value of λ is used together with the contents of `options`.
2. The second part is a modified version of a golden section search in which the optimal value of λ is found within the convergence interval of the specified iterative method.

The method returns the optimal value in the output `lambda`.

Use of `options`:

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

- `options.restart`: See the specific method for correct use.
- `options.w`: If the chosen method allows weigths this parameter can be set.

Example:

Generate a "noisy" 50×50 parallel beam tomography problem, train to find the optimal value of λ for the SIRT method `cimmino`, and use the found value to perform 50 iterations of this method. Finally the last iterate is shown:

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

See also:

`trainLambdaART`