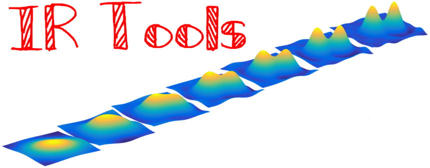
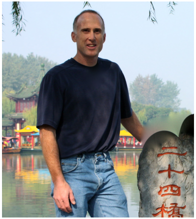
IR Tools: A Matlab Package with Iterative Regularization Methods for Inverse Problems

Per Christian Hansen
DTU Compute, Technical University of Denmark

Joint work with Silvia Gazzola, Univ. of Bath & Jim Nagy, Emory Univ.





Why (Matlab) Software Packages?

For teaching, training and research:

- Get to know a collection of methods that focus on a common theme.
- Solve the same problem with different methods; performance study.
- Solve different problems with the same method; robustness study.
- Use the package in a variety of applied mathematics courses.

Why (Matlab) Software Packages?

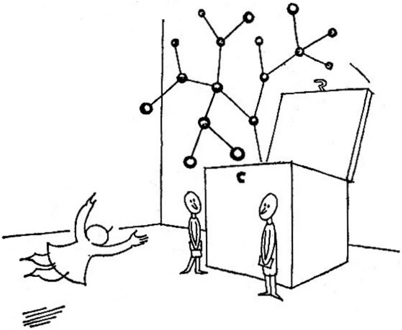
For teaching, training and research:

- Get to know a collection of methods that focus on a common theme.
- Solve the same problem with different methods; performance study.
- Solve different problems with the same method; robustness study.
- Use the package in a variety of applied mathematics courses.

For problem solving:

- Solve a difficult problem with an advanced method, without the need to carefully implement the method yourself.
- Software templates can be used for specialized implementations.
- Make modern numerical methods available to the users.
- Get the methods out, beyond papers in specialized journals.

Problems Solvers



THE ONLY SOLUTION

We shall have to evolve
problem-solvers galore -
since each problem they solve
creates ten problems more.

Piet Hein, Denmark, 1905–96

Overview of Talk

1. Inverse problems and regularization.
2. Overview of IR Tools.
3. Tikhonov regularization, regularizing iterations, and `IRcgls`.
4. Hybrid regularization methods and `IRlsqr_hybrid`.
5. Illustration of some test problems and the use of iterative methods:
    - image deblurring,
    - computed tomography,
    - inverse interpolation.

Overview of Talk

1. Inverse problems and regularization.
2. Overview of IR Tools.
3. Tikhonov regularization, regularizing iterations, and `IRcgls`.
4. Hybrid regularization methods and `IRlsqr_hybrid`.
5. Illustration of some test problems and the use of iterative methods:
   - image deblurring,
   - computed tomography,
   - inverse interpolation.

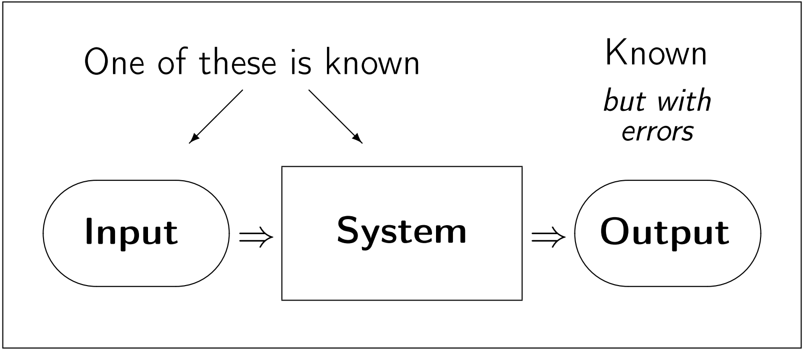
Get the software here: `http://people.compute.dtu.dk/pcha/IRtools/`

S. Gazzola, P. C. Hansen, and J. G. Nagy, *IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems*, Numerical Algorithms, 81 (2019), pp. 773-811. doi: 10.1007/s11075-018-0570-7.

What is an Inverse Problem?

In a **forward problem**, we use a mathematical model to compute the output from a "system" given the input – or compute the "system" given the input and the output.

In an **inverse problem** we estimate a quantity that is not directly observable, using indirect measurements.



Discretized Linear Inverse Problems

The basic problem

$$\boxed{\text{Solve } A\,x = b \text{ with } A \text{ ill conditioned.}}$$

The underlying model

$$b = A\,\bar{x} + e\ , \qquad \bar{x} = \text{exact solution,} \quad e = \text{noise.}$$

There are no restrictions on the dimensions of A and the noise is unknown.

Discretized Linear Inverse Problems

The basic problem

$$\boxed{\text{Solve } A\,x = b \text{ with } A \text{ ill conditioned.}}$$

The underlying model

$$b = A\,\bar{x} + e\ , \qquad \bar{x} = \text{exact solution}, \quad e = \text{noise}.$$

There are no restrictions on the dimensions of A and the noise is unknown.

- Our *analysis tool:* the SVD $A = U\,\mathrm{diag}(\sigma_i)\,V^T$.
- The singular values σ_i decay do zero with no gap anywhere.
- The exact right-hand side $\bar{b} = A\,\bar{x}$ satisfies the Picard condition:

  the coefficients $u_i^T\,\bar{b}$ decay *faster* than the σ_i.

The dimensions of A – i.e., the amount of data and the number of unknowns – are large $\rightarrow$ iterative methods.

Regularization Methods

The "naive solution" to an inverse problem

$$x^{\text{naive}} = A^{-1} b = A^{-1} \bar{b} + A^{-1} e = \bar{x} + A^{-1} e$$

is dominated by the inverted noise $A^{-1} e$, due to the ill conditioned A.

Use *regularization* to handle the amplification of noise in $A^{-1} e$.

- **Truncated SVD:** $x_k \equiv \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i$ – small problems only.
- **Tikhonov:** $x_\lambda \equiv \arg\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \mathcal{R}(x) \right\}$.
- **Regularization term** $\mathcal{R}(x)$:
  - smoothness: $\|x\|_2^2$ or $\|L\,x\|_2^2$
  - sparsity: $\|x\|_1$
  - total variation: sparse gradient magnitude.
- **Regularizing iterations:** truncated the iterations of a (least squares) solver, such as Kacmarz, Landweber, Cimmino, CGLS, and GMRES.

Matlab Software Packages for Inverse Problems

Regularization Tools (H, 1994, 1999, 2007)

- Basic methods for small problems. Everything is based on the SVD.
- Tiny, easy, and outdated test problems.

Matlab Software Packages for Inverse Problems

Regularization Tools (H, 1994, 1999, 2007)

- Basic methods for small problems. Everything is based on the SVD.
- Tiny, easy, and outdated test problems.

Restore Tools (Nagy, Palmer, Perrone, 2004, 2007, 2012)

- Image deblurring problems. Mainly iterative solvers. Object oriented.
- Deblurring test problems only.

Matlab Software Packages for Inverse Problems

Regularization Tools (H, 1994, 1999, 2007)

- Basic methods for small problems. Everything is based on the SVD.
- Tiny, easy, and outdated test problems.

Restore Tools (Nagy, Palmer, Perrone, 2004, 2007, 2012)

- Image deblurring problems. Mainly iterative solvers. Object oriented.
- Deblurring test problems only.

AIR Tools II (H, Jørgensen, 2018)

- Expanded & improved version of AIR Tools (H, Saxild-Hansen, 2012).
- Algebraic iterative reconstruction methods for tomography problems.
- Tomography test problems only.

Matlab Software Packages for Inverse Problems

Regularization Tools (H, 1994, 1999, 2007)

- Basic methods for small problems. Everything is based on the SVD.
- Tiny, easy, and outdated test problems.

Restore Tools (Nagy, Palmer, Perrone, 2004, 2007, 2012)

- Image deblurring problems. Mainly iterative solvers. Object oriented.
- Deblurring test problems only.

AIR Tools II (H, Jørgensen, 2018)

- Expanded & improved version of AIR Tools (H, Saxild-Hansen, 2012).
- Algebraic iterative reconstruction methods for tomography problems.
- Tomography test problems only.

IR Tools (Gazzola, H, Nagy, 2019)

- Iterative regularization methods for large-scale problems.
- Tikhonov-type probelms and regularizing iterations.
- Realistic 2D test problems (how many years will they last?).

The 10 Conventions in IR Tools

1. Easy installation; no compilation; no need for additional toolboxes.
2. Interface to the package AIR Tools II for computed tomography.
3. All iterative solvers have the form

   ```
   [X, Info] = IR___(A, b, K, options)
   ```

4. Information about the performance is returned in the `Info` structure.
5. Stopping rules are integrated in the iterative methods.
6. All test problem generators have the form

   ```
   [A, b, x, ProbInfo] = PR___(n, options)
   ```

7. Realistic 2D test problems that require no background knowledge.
8. Default values are provided for all parameters.
9. Users can take full control via an optional `options` input structure.
10. Visualization of b and x is always done by `PRshowb` and `PRshowx`.

What We Can Do With IR Tools

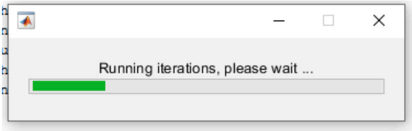
Solve a 2D image deblurring problem with CGLS regularizing iterations.

First generate a deblurring test problem with std. parameters:

```
NoiseLevel = 0.01;
[A, b, x, ProbInfo] = PRblurspeckle;
[bn, NoiseInfo] = PRnoise(b, 'gauss', NoiseLevel);
```

Run CGLS with the discrepancy principle stopping rule:

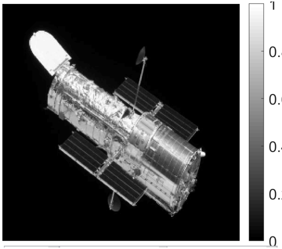
```
options = IRset('NoiseLevel', NoiseLevel);
[Xcgls, IterInfo] = IRcgls(A, bn, options);
```

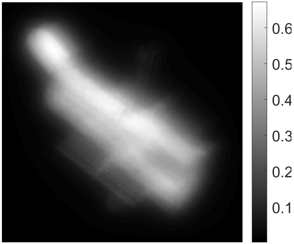


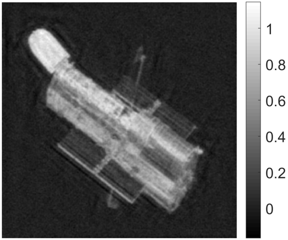
And Now – the Results

Plot the results:

```
figure(1), PRshowx(x, ProbInfo)
figure(2), PRshowb(b, ProbInfo)
figure(3), PRshowx(Xcgls, ProbInfo)
```







Types of Problems That Can be Solved with IR Tools

Problem type	Functions
$\min_x \|A\,x - b\|_2^2$ + semi-convergence	`IRart`, `IRcgls`, `IRenrich`, `IRsirt`, `IRrrgmres` ($M = N$ only)
$\min_x \|A\,x - b\|_2^2$ s.t. $x \geq 0$ + semi-convergence	`IRmrnsd`, `IRnnfcgls`
$\min_x \|A\,x - b\|_2^2$ s.t. $x \in \mathcal{C}$ + semi-convergence	`IRconstr_ls`, `IRfista`
$\min_x \|A\,x - b\|_2^2 + \lambda^2 \|L\,x\|_2^2$	`IRcgls`, `IRhybrid_lsqr`, `IRhybrid_gmres` ($M = N$ only)
$\min_x \|A\,x - b\|_2^2 + \lambda^2 \|L\,x\|_2^2$ s.t. $x \in \mathcal{C}$	`IRconstr_ls`, `IRfista` ($L = I$ only)
$\min_x \|A\,x - b\|_2^2 + \lambda \|x\|_1$	`IRell1` ($M = N$ only), `IRhybrid_fgmres` ($M = N$ only), `IRirn`
$\min_x \|A\,x - b\|_2^2 + \lambda \|x\|_1$ s.t. $x \geq 0$,	`IRirn`
$\min_x \|A\,x - b\|_2^2 + \lambda \mathrm{TV}(x)$ with or without constraint $x \geq 0$	`IRhtv`

The matrix L must have full rank.
$\mathcal{C}$ is either the box $[\mathtt{xMin}, \mathtt{xMax}]^N$ or the set defined by $\|x\|_1 = \mathtt{xEnergy}$.

Test Problems in IR Tools

Test problem type	**Function**	**Type of A**
Image deblurring	`PRblur` (generic function)	
– spatially invariant blur	`PRblurdefocus`, `PRblurgauss`, `PRblurmotion`, `PRblurshake`, `PRblurspeckle`	Object
– spatially variant blur	`PRblurrotation`	Sparse matrix
Inverse diffusion	`PRdiffusion`	Function handle
Inverse interpolation	`PRinvinterp2`	Function handle
NMR relaxometry	`PRnmr`	Function handle
Tomography		Sparse matrix or
– travel-time tomography	`PRseismic`	function handle
– spherical means tomography	`PRspherical`	ditto
– X-ray computed tomography	`PRtomo`	ditto

Add noise to the data (Gauss, Laplace, multiplicative): `PRnoise`

Visualize the data b and the solution x in appropriate formats: `PRshowb`, `PRshowx`

Solving a Least Squares Problem

Consider the least squares problem without regularization

$$x_{\mathrm{LS}} = \arg\min_{x} \|A\,x - b\|_2^2 \ ,$$

with the equivalent formulation

$$A^T A\,x = A^T b \ .$$

We can use `IRcgls` to solve this problem.

Solving a Least Squares Problem

Consider the least squares problem without regularization

$$x_{\mathrm{LS}} = \arg\min_x \|A\,x - b\|_2^2 \ ,$$

with the equivalent formulation

$$A^T A\,x = A^T b \ .$$

We can use `IRcgls` to solve this problem.

Relevant stopping rules:

$$k = \mathtt{MaxIter} \ ,$$

$$\left\|A^T A\,x^{(k)} - A^T b\right\|_2 \leq \mathtt{NE\_Rtol} \cdot \|A^T b\|_2 \ .$$

Calling `IRcgls`

The simplest call, using all default parameters:

```
[X, info] = IRcgls(A, b);
```

`X` holds the final iterate, and `info` is a structure with lots of information.
E.g., `info.its` is the number of the last computed iteration.

Calling `IRcgls`

The simplest call, using all default parameters:

```
[X, info] = IRcgls(A, b);
```

`X` holds the final iterate, and `info` is a structure with lots of information. E.g., `info.its` is the number of the last computed iteration.

Specify which iterates are stored in `X`:

```
K = 25:25:500;
[X, info] = IRcgls(A, b, K);
```

Note that `MaxIter` = max(`K`) and that `info.saved_iterations` holds the iteration numbers of the iterates stored in `X`.

Calling IRcgls

The simplest call, using all default parameters:

```
[X, info] = IRcgls(A, b);
```

`X` holds the final iterate, and `info` is a structure with lots of information. E.g., `info.its` is the number of the last computed iteration.

Specify which iterates are stored in `X`:

```
K = 25:25:500;
[X, info] = IRcgls(A, b, K);
```

Note that `MaxIter` = max(`K`) and that `info.saved_iterations` holds the iteration numbers of the iterates stored in `X`.

Set your own options:

```
options = IRset('MaxIter',500, 'NE_Rtol',1e-8)
K = 25:25:500;
[X, info] = IRcgls(A, b, K, options);
```

Alternatively: `options.MaxIter = 500; options.NE_Rtol = 1e-8;`

What Information is in info From IRcgls?

```
info: structure with the following fields:
  its       - number of the last computed iteration
  saved_iterations - iteration numbers of iterates stored in X
  StopFlag - a string that describes the stopping condition:
              * Reached maximum number of iterations
              * Residual tolerance satisfied (discrepancy principle)
              * Normal equation residual tolerance satisfied
  StopReg  - struct containing information about the solution that
           satisfies the stopping criterion, with the fields:
              It   : iteration where the stopping criterion is satisfied
              X    : the solution satisfying the stopping criterion
              Enrm : the best relative error (requires x_true)
  Rnrm     - relative residual norms at each iteration
  NE_Rnrm  - normal eqs relative residual norms
  Xnrm     - solution norms at each iteration
  Enrm     - relative error norms (requires x_true) at each iteration
  BestReg  - struct containing information about the solution that
           minimizes Enrm (requires x_true), with the fields:
              It   : iteration where the minimum is attained
              X    : best solution
              Enrm : best relative error
```

Tikhonov Regularization

Now consider the Tikhonov regularization problem

$$x_\lambda = \arg\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \|L\,x\|_2^2 \right\} \ ,$$

where L may be the identity matrix or an approximation to a derivative operator. There are two equivalent formulations:

$$(A^T A + \lambda^2 L^T L)\,x = A^T b \ , \qquad \min_x \left\| \begin{pmatrix} A \\ \lambda L \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2 \ .$$

We can also use `IRcgls` to solve this linear least squares problem.

Tikhonov Regularization

Now consider the Tikhonov regularization problem

$$x_\lambda = \arg\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \|L\,x\|_2^2 \right\} ,$$

where L may be the identity matrix or an approximation to a derivative operator. There are two equivalent formulations:

$$(A^T A + \lambda^2 L^T L)\,x = A^T b \, , \qquad \min_x \left\| \begin{pmatrix} A \\ \lambda L \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2 .$$

We can also use `IRcgls` to solve this linear least squares problem.

Relevant stopping rules:

$$k = \texttt{MaxIter} \, ,$$

$$\left\| (A^T A + \lambda^2 L^T L)\,x^{(k)} - A^T b \right\|_2 \leq \texttt{NE\_Rtol} \cdot \|A^T b\|_2 \, .$$

Calling `IRcgls` for Tikhonov Regularization

The simplest call, using a fixed regularization parameter λ and the default regularization matrix $L = I$:

```
options = IRset('RegParam',λ)
[X, info] = IRcgls(A, b, options);
```

Note that we do not need to specify the number of iterations `K`; the default maximum number of iterations is `MaxIter = 100` (quite small).

Calling `IRcgls` for Tikhonov Regularization

The simplest call, using a fixed regularization parameter λ and the default regularization matrix $L = I$:

```
options = IRset('RegParam',λ)
[X, info] = IRcgls(A, b, options);
```

Note that we do not need to specify the number of iterations `K`; the default maximum number of iterations is `MaxIter = 100` (quite small).

Use `options` to specify a regularization matrix $L \neq I$:

- `'Laplacian1D'` and `'Laplacian2D'` give second-order smoothing.
- A matrix L specified by the user.
- A function handle to a function, written by the user, that computes matrix-vector products with L and L^T.

CGLS Regularizing Iterations

If we apply CGLS to the un-regularized problem, then the iterates satisfy

$$x^{(k)} = \arg\min_x \|A\,x - b\|_2 \qquad \text{s.t.} \qquad x \in \mathcal{K}_k \ ,$$

where

$$\mathcal{K}_k = \text{span}\{A^T b, (A^T A)\, A^T b, \ldots, (A^T A)^{k-1} A^T b\} \ .$$

The challenge is to stop the iterations when k is just large enough, and stopping rule = regularization-parameter choice (GCV, L-curve, etc.).

CGLS Regularizing Iterations

If we apply CGLS to the un-regularized problem, then the iterates satisfy

$$x^{(k)} = \arg\min_x \|A\,x - b\|_2 \qquad \text{s.t.} \qquad x \in \mathcal{K}_k \ ,$$

where

$$\mathcal{K}_k = \operatorname{span}\{A^T b, (A^T A)\,A^T b, \ldots, (A^T A)^{k-1} A^T b\} \ .$$

The challenge is to stop the iterations when k is just large enough, and stopping rule = regularization-parameter choice (GCV, L-curve, etc.).

Recall our model $b = A\,\bar{x} + e$. We implemented the *discrepancy principle:*

$$\text{stop as soon as } \|A\,x^{(k)} - b\|_2 \leq \eta\,\|e\|_2 \ ,$$

where η is a "safety factor" (default 1.01).

`options = IRset('NoiseLevel',`$\|e\|_2/\|b\|_2$`);` NB: rel. noise level.

`options = IRset(options, 'eta',1.2);` If we want to set η.

Studying Convergence in IR Tools

Monitor the iterations and the convergence – beyond the iteration where the stopping rule is satisfied – assuming that we know the true solution $\bar{x}$.

```
options = IRset('x_true',x̄, 'NoStop','on');
[X, info] = IRcgls(A, b, K, options);
```

Studying Convergence in IR Tools

Monitor the iterations and the convergence – beyond the iteration where the stopping rule is satisfied – assuming that we know the true solution $\bar{x}$.

```
options = IRset('x_true',x̄, 'NoStop','on');
[X, info] = IRcgls(A, b, K, options);
```

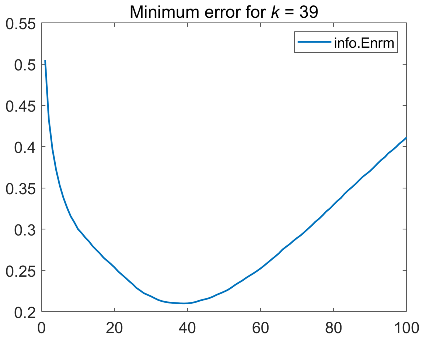
Pay attention to these fields in the output `info` structure:

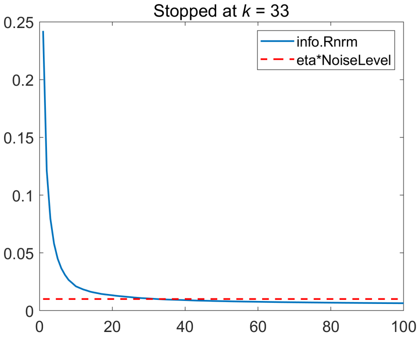
`StopFlag`	a string that describes the stopping condition
`Xnrm`	solution norms at each iteration
`Enrm`	relative error norms at each iteration (requires `x_true`)
`Stopreg.It`	iteration where the stopping criterion is satisfied
`StopReg.X`	the solution that satisfying the stopping criterion
`StopReg.Enrm`	the corresponding relative error (requires `x_true`)
`BestReg.It`	iteration where the minimum of `Enrm` is attained
`BestReg.X`	best solution
`BestReg.Enrm`	best relative error

Illustration of Convergence Study

```
NoiseLevel = 0.01;                         % Relative noise level.
[A, b, x, ProbInfo] = PRblurspeckle;       % Atmospheric turbulence blur.
[bn, NoiseInfo] = PRnoise(b, 'gauss', NoiseLevel);  % Additive noise.
options = IRset('NoiseLevel', NoiseLevel, 'NoStop','on', 'x_true',x);
[X, info] = IRcgls(A, bn, options);
```

```
info.StopFlag    : 'Residual tolerance satisfied'
info.StopReg.It  : 33
info.BestReg.It  : 39
```





Pros and Cons of Tikhonov & CGLS

Tikhonov regularization. In terms of the SVD $A = \sum_{i=1}^{n} u_i \, \sigma_i \, v_i^T$ we have

$$x_\lambda = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \, \frac{u_i^T b}{\sigma_i} \, v_i \ ,$$

clearly showing the filtering of the SVD components.

But we may need to try many different values of λ.

Regularizing iterations (CGLS). Here the regularization is achieve by restricting the solution $x^{(k)}$ to lie in the Krylov subspace $\mathcal{K}_k$, and it is convenient that k is a regularization parameter.

But noise may enter in $x^{(k)}$ if $\mathcal{K}_k$ picks up unwanted SVD components.

$\Rightarrow$ Combine the two methods $\rightarrow$ next slide.

A Hybrid Method Based on LSQR

LSQR is an alternative implementation of CGLS; at iteration k we have

$$A\,V_k = U_{k+1}\,B_k \qquad \text{and} \qquad x^{(k)} = V_k\,y_k\ ,$$

where $\mathcal{K}_k = \mathrm{range}(V_k)$ and

$$y_k = \arg\min_k \|B_k\,y - (U_{k+1}^T b)\|_2^2\ .$$

A Hybrid Method Based on LSQR

LSQR is an alternative implementation of CGLS; at iteration k we have

$$A\,V_k = U_{k+1}\,B_k \qquad \text{and} \qquad x^{(k)} = V_k\,y_k \ ,$$

where $\mathcal{K}_k = \mathrm{range}(V_k)$ and

$$y_k = \arg\min_k \|B_k\,y - (U_{k+1}^T b)\|_2^2 \ .$$

The **hybrid** method:

$$\boxed{y_k = \arg\min_k \left\{ \|B_k\,y - (U_{k+1}^T b)\|_2^2 + \lambda_k^2\,\|y\|_2^2 \right\}}$$

where we choose a regularization parameter λ_k in each iteration, by means of the discrepancy principle, GCV, the L-curve, etc.

We implemented this in the function `IRhybrid_lsqr`.

A GMRES-based hybrid method implemented is in `IRhybrid_gmres`.

IRhybrid_lsqr in Action

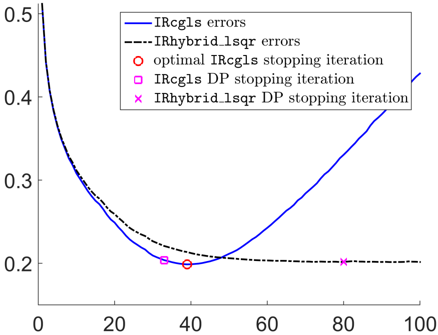
If we set the regularization parameter to a fixed value λ,

```
options = IRset('RegParam',λ);
```

then IRhybrid_lsqr is identical to IRcgls appl. to the Tikhonov problem.

We obtain a true hybrid method if the regularization parameter λ_k is chosen in each iteration; here we use *weighted GCV*.

```
options = IRset(options, 'RegParam', 'wgcv');
[X, iter] = IRhybrid_lsqr(A, bn, options);
```



Many Other Methods in IR Tools

General-form regularization (`IRcgls`, `IRhybrid_lsqr`, `IRhybrid_gmres`):

$$\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2\,\|L\,x\|_2^2 \right\} \ .$$

Many Other Methods in IR Tools

General-form regularization (`IRcgls`, `IRhybrid_lsqr`, `IRhybrid_gmres`):

$$\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \, \|L\,x\|_2^2 \right\} \, .$$

In the regularizing iterations we can incorporate L by *priorconditioning* with $M = (L^T L)^{-1}$ (`IRcgls`, `IRhybrid_lsqr`):

$$x^{(k)} \in \mathrm{span}\{ M\,A^T b, (M\,A^T A)\,M\,A^T b, \ldots (M\,A^T A)^{k-1} M\,A^T b \} \, .$$

Many Other Methods in IR Tools

General-form regularization (IRcgls, IRhybrid_lsqr, IRhybrid_gmres):

$$\min_x \left\{ \|A\, x - b\|_2^2 + \lambda^2 \, \|L\, x\|_2^2 \right\} \ .$$

In the regularizing iterations we can incorporate L by *priorconditioning* with $M = (L^T L)^{-1}$ (IRcgls, IRhybrid_lsqr):

$$x^{(k)} \in \mathrm{span}\{M\, A^T b, (M\, A^T A)\, M\, A^T b, \ldots (M\, A^T A)^{k-1} M\, A^T b\} \ .$$

We can *enrich* the Krylov subspace (IRenrich):

$$x^{(k)} \in \mathrm{span}\{A^T b, (A^T A)\, A^T b, \ldots (A^T A)^{k-1} A^T b\} + \mathrm{span}\{w_1, \ldots, w_p\} \ .$$

Many Other Methods in IR Tools

General-form regularization (IRcgls, IRhybrid_lsqr, IRhybrid_gmres):

$$\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2\,\|L\,x\|_2^2 \right\} \ .$$

In the regularizing iterations we can incorporate L by *priorconditioning* with $M = (L^T L)^{-1}$ (IRcgls, IRhybrid_lsqr):

$$x^{(k)} \in \mathrm{span}\{ M\,A^T b, (M\,A^T A)\,M\,A^T b, \ldots (M\,A^T A)^{k-1} M\,A^T b \} \ .$$

We can *enrich* the Krylov subspace (IRenrich):

$$x^{(k)} \in \mathrm{span}\{ A^T b, (A^T A)\,A^T b, \ldots (A^T A)^{k-1} A^T b \} + \mathrm{span}\{ w_1, \ldots, w_p \} \ .$$

We can add nonnegativity (IRmrnsd, IRconstr_ls, IRnnfcgls, IRirn).

Many Other Methods in IR Tools

General-form regularization (IRcgls, IRhybrid_lsqr, IRhybrid_gmres):

$$\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2\, \|L\,x\|_2^2 \right\} \, .$$

In the regularizing iterations we can incorporate L by *priorconditioning* with $M = (L^T L)^{-1}$ (IRcgls, IRhybrid_lsqr):

$$x^{(k)} \in \operatorname{span}\{ M\, A^T b, (M\, A^T A)\, M\, A^T b, \ldots (M\, A^T A)^{k-1} M\, A^T b \} \, .$$

We can *enrich* the Krylov subspace (IRenrich):

$$x^{(k)} \in \operatorname{span}\{ A^T b, (A^T A)\, A^T b, \ldots (A^T A)^{k-1} A^T b \} + \operatorname{span}\{ w_1, \ldots, w_p \} \, .$$

We can add nonnegativity (IRmrnsd, IRconstr_ls, IRnnfcgls, IRirn).

We can use other regularization terms: sparsity $\lambda \|x\|_1$ (IRell1, IRfista, IRhybrid_fgmres), total variation (IRhtv).

Test Problem: Image Deblurring PRblur

The basic call:

```
[A, b, x, ProbInfo] = PRblur;
```

ProbInfo is a structure with information about the problem:

```
problemType: 'deblurring'
      xType: 'image2D'
      xSize: [256 256]
      bType: 'image2D'
      bSize: [256 256]
        psf: [256x256double]
```

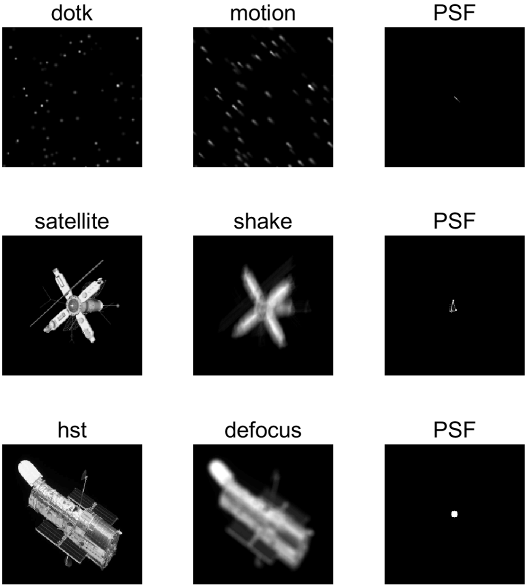
The general call:

```
[A, b, x, ProbInfo] = PRblur(n, options);
```

The image is n $\times$ n (so $x \in \mathbb{R}^{\mathtt{n}^2}$), and options has such fields as:

- trueImage – test image, e.g., 'ppower', 'satellite', 'hst'
- PSF – point spread function, e.g., 'gauss', 'defocus', 'shake'
- BlurLevel – severity of the blur: 'mild', 'medium', 'severe'

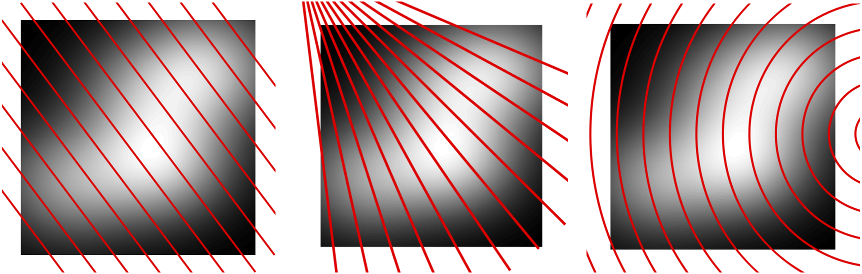
PRblur: Some Test Images and Point Spread Functions



Test Problems: Computed Tomography (CT)

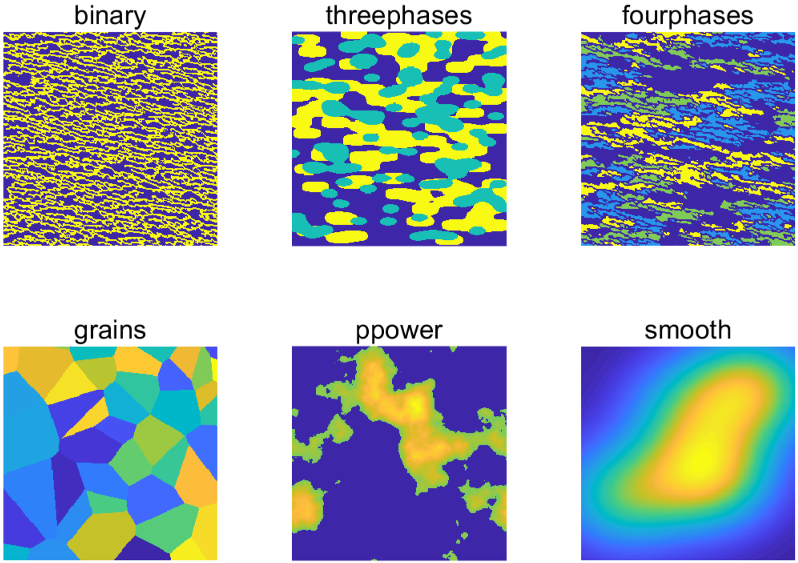
We provide three X-ray CT test problems.

- **Parallel beam:** `PRtomo` with `options.CTtype = 'parallel'`.
- **Fan beam:** `PRtomo` with `options.CTtype = 'fancurved'`.
- **Spherical means:** `PRspherical`.



Full control over the measurement geometry via `options`.

PRtomo and PRspherical: Test Images (Phantoms



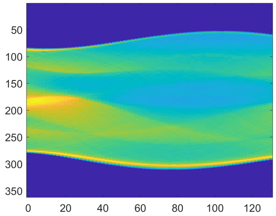
Example: Limited-Angle Fan Beam CT Test Problem

Fan beam geometry, limited-range projection angles, multiplicative noise.

```
n = 256;
options.CTtype = 'fancurved';
options.angles = 0:2:130;
[A,b,x,ProbInfo] = PRtomo(n,options);
[bn,NoiseInfo] = PRnoise(b,'multiplicative');
```

The fields of `ProbInfo`:

```
problemType: 'tomography'
      xType: 'image2D'
      bType: 'image2D'
      xSize: [256 256]
      bSize: [362 66]
```



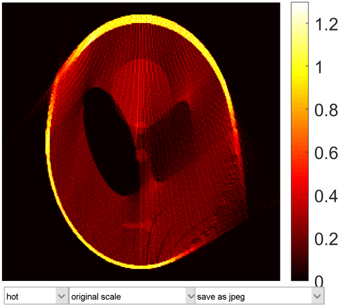
The fields of `NoiseInfo`:

```
 kind: 'multiplicative'
level: 1.0000e-02
noise: [23892x1 double]
```

Reconstruction by IRart (Kaczmarz)

Nonnegativity constraints and the discrepancy principle stopping criterion:

```
options.stopCrit = 'discrep';
options.NoiseLevel = NoiseInfo.level;
options.eta = 1.5;
options.nonnegativity = 'on';
[X,info] = IRart(A,b,options);
PRshowx(X,ProbInfo);
```

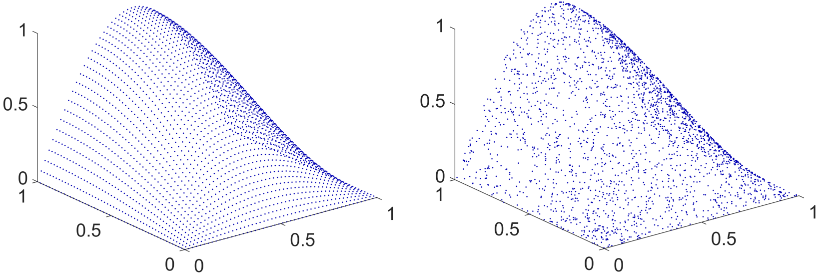


Severe artifacts due to the limited-angle geometry.

Test Problem: 2D Inverse Interpolation PRinvinterp2

Inverse interpolation (gridding): compute values of a function on a regular grid, given function values on arbitrarily located points.

```
[A, b, x, ProbInfo] = PRinvinterp2;
PRshowx(x, ProbInfo)
PRshowb(b, ProbInfo)
```



Interpolation of the gridded function values (the unknowns) must produce the given values (the data). We provide nearest-neighbour, linear (default), cubic, and spline interpolation.

Solution by Priorconditioned CGLS, Part I

Define a small test problem with a 32×32 grid:

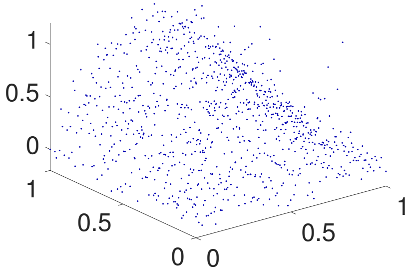
```
[A, b, x, ProbInfo] = PRinvinterp(32);
bn = PRnoise(b, 0.05);
```

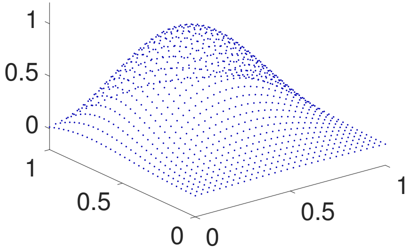
Standard CGLS fails to recognize a good stopping iteration; the final solution is poor.

```
[X1, IterInfo1] = IRcgls(A, bn, 1:200);
```

Priorconditioned CGLS with L representing the 2D Laplacian enforces zero boundary conditions everywhere, which is undesired.

```
options.RegMatrix = 'Laplacian2D';
[X2, IterInfo2] = IRcgls(A, bn, 1:200, options);
```

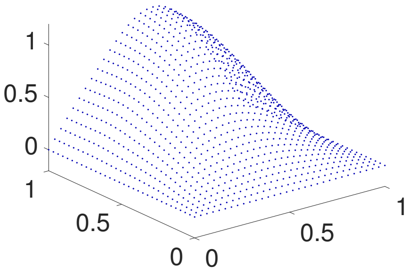




Solution by Priorconditioned CGLS, Part II

We create our own prior-conditioning matrix L that is similar to the 2D Laplacian, except we enforce a *zero derivative* on the appropriate boundary.

```
L1 = spdiags([ones(n,1),-2*ones(n,1),ones(n,1)],[-1,0,1],n,n);
L1(1,1:2) = [1,0]; L1(n,n-1:n) = [0,1];
L2 = L1; L2(n,n-1:n) = [-1,1];
L = [ kron(speye(n),L2) ; kron(L1,speye(n)) ];
L = qr(L,0);
options.RegMatrix = L;
[X3, IterInfo3] = IRcgls(A, bn, 1:200, options);
```



Conclusions

- We presented a recent Matlab software package IR Tools with iterative regularization methods.
- The package also includes realistic 2D test problems (please stop using Regularization Tools now).
- Very easy basic use of the iterative solvers (don't worry about parameters, stopping rules, etc.).
- Full control of all parameters and stopping rules of the iterative solvers, if needed.
- Please try the package and send bug reports to us.