LSTRS 1.2: MATLAB Software for Large-Scale Trust-Regions Subproblems and Regularization

Marielba Rojas

Informatics and Mathematical Modelling
Technical University of Denmark

Computational Methods with Applications
Harrachov, Czech Republic
August 19-25, 2007

Joint work with
Sandra A. Santos, Campinas, Brazil
Danny C. Sorensen, Rice, USA

Special thanks to Wake Forest, CERFACS, and T.U. Delft.

Outline

- The Trust-Region Subproblem
- LSTRS - The basic idea
- LSTRS - The Algorithm
- LSTRS - The Software
- Comparisons
- Applications

Basic Idea

Trust-Region Subproblem

$$\begin{array}{ll} \min & \frac{1}{2}x^T H x + g^T x \\ s.t. & \|x\| \leq \Delta \end{array}$$

- $H \in \mathbb{R}^{n\times n}$, $H = H^T$, n large
- $g \in \mathbb{R}^n$, $g \neq 0$
- $\Delta > 0$

Regularization Problem

$$\begin{array}{ll} \min & \frac{1}{2}\|Ax - b\|^2 \\ s.t. & \|x\| \leq \Delta \end{array}$$

- $A \in \mathbb{R}^{m \times n}$, $m \geq n$ large, from ill-posed problem
- $b \in \mathbb{R}^m$, containing noise, and $A^T b \neq 0$
- $\Delta > 0$

Trust-Region Subproblem

Characterization of solutions. Gay 1981, Sorensen 1982.

x_* with $\|x_*\| \leq \Delta$ is a solution of TRS with Lagrange multiplier λ_*, if and only if

(i) $(H - \lambda_* I)x_* = -g$.

(ii) $H - \lambda_* I$ positive semidefinite.

(iii) $\lambda_* \leq 0$.

(iv) $\lambda_* \left(\|x_*\| - \Delta\right) = 0$.

TRS as Parameterized Eigenvalue Problem

Consider the *bordered* matrix

$$B_\alpha = \begin{pmatrix} \alpha & g^T \\ g & H \end{pmatrix}$$

Then

- Eigenvalues of H interlace eigenvalues of B_α
- $\exists\ \alpha$ such that TRS equivalent to

$$\begin{array}{ll} \min & \frac{1}{2} y^T B_\alpha y \\ s.t. & y^T y \le 1 + \Delta^2 \\ & e_1^T y = 1 \end{array}$$

LSTRS

Note $\begin{pmatrix} \alpha & g^T \\ g & H \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = \lambda \begin{pmatrix} 1 \\ x \end{pmatrix} \quad \Leftrightarrow \quad \begin{array}{rcl} \alpha - \lambda & = & -g^T x \\ (H - \lambda I)x & = & -g \end{array}$

Let $H = Q\ diag(\delta_1, \delta_2, \ldots, \delta_n)\ Q^T$ and $\gamma_i = Q^T g, \ \ i = 1, 2, \ldots, n$

Suppose $x \in \mathbb{R}^n$ such that $(H - \lambda I)x = -g$.

Define $\qquad \phi(\lambda) = -g^T x = \sum_{i=1}^{n} \frac{\gamma_i^2}{\delta_i - \lambda}$

Then $\phi'(\lambda) = x^T x$.

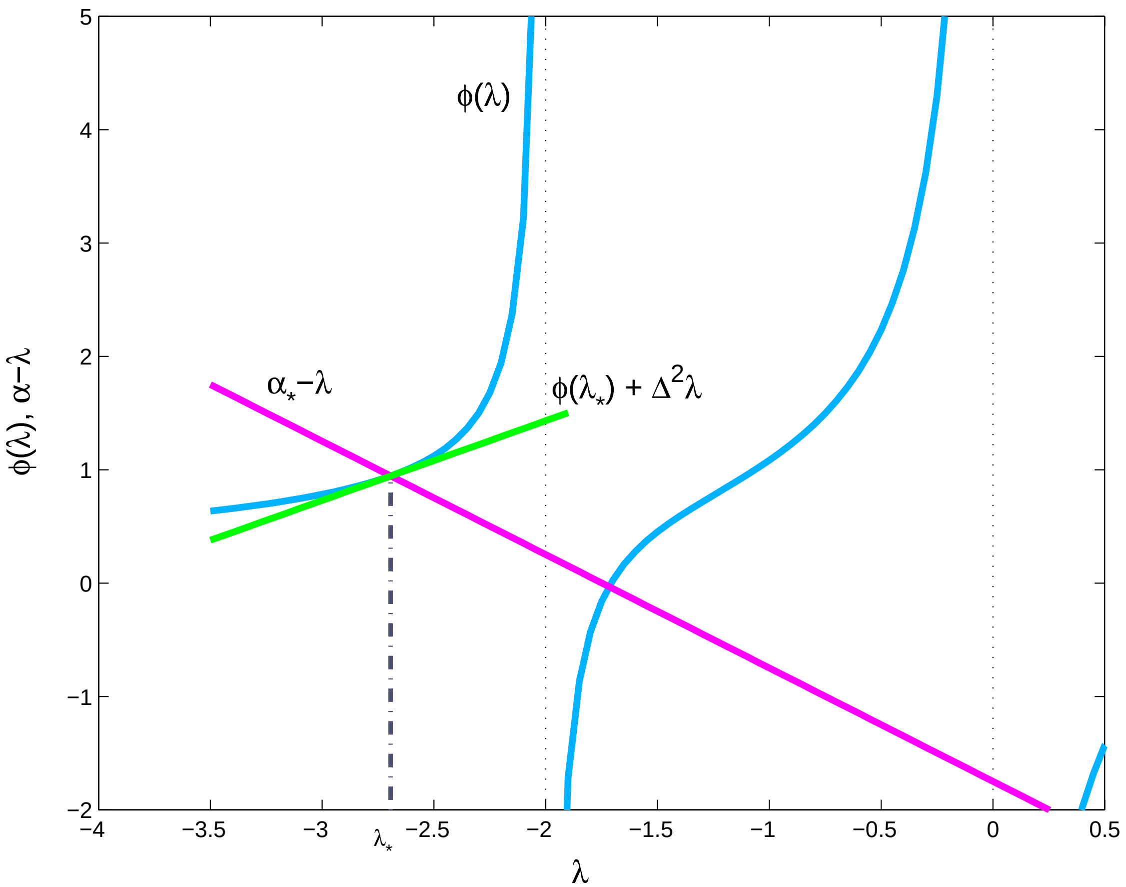
Idea: Adjust α such that $\alpha - \hat{\lambda} = \phi(\hat{\lambda})$ with $\phi'(\hat{\lambda}) = \Delta^2$.

LSTRS

- Compute rational interpolant ϕ and adjust α such that $\alpha - \hat{\lambda} = \phi(\hat{\lambda})$ with $\phi'(\hat{\lambda}) = \Delta^2$.
- Obtain interpolation points by solving large-scale eigenvalue problems for *smallest* eigenvalue of B_α.
- Solve eigenvalue problems with efficient method such as ARPACK (matrix-free, fixed storage).

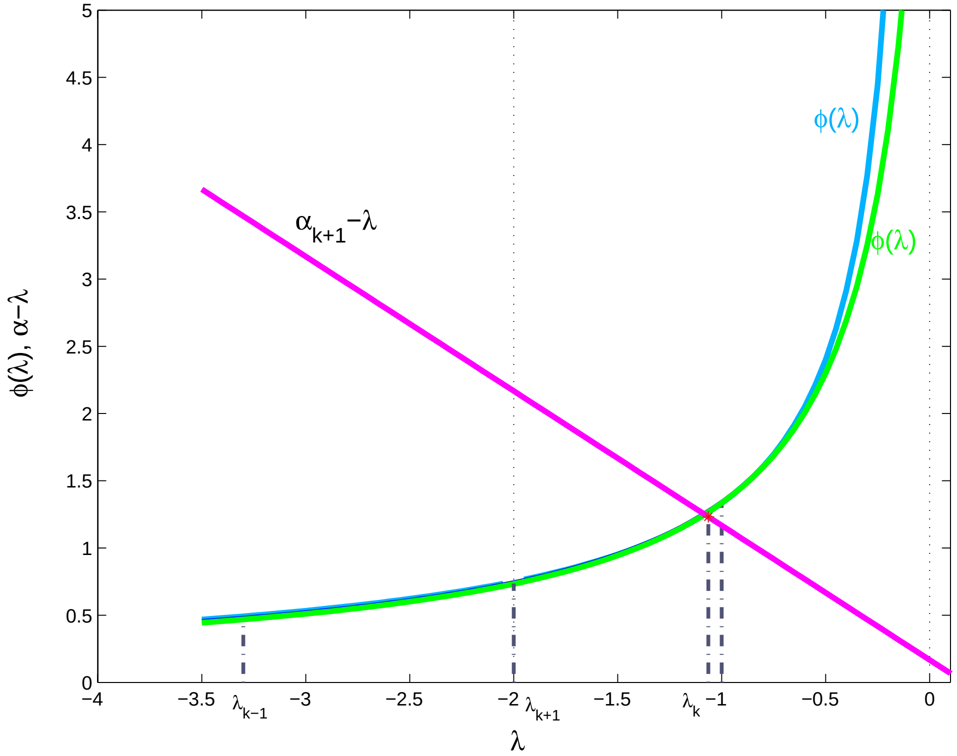
LSTRS in pictures - the standard case

$$(\mathbf{H} - \lambda \mathbf{I})\mathbf{x} = -\mathbf{g}, \ \alpha - \lambda = \phi(\lambda) \approx -\mathbf{g}^{\mathbf{T}}\mathbf{x}, \ \phi'(\lambda) = \mathbf{x}^{\mathbf{T}}\mathbf{x}.$$

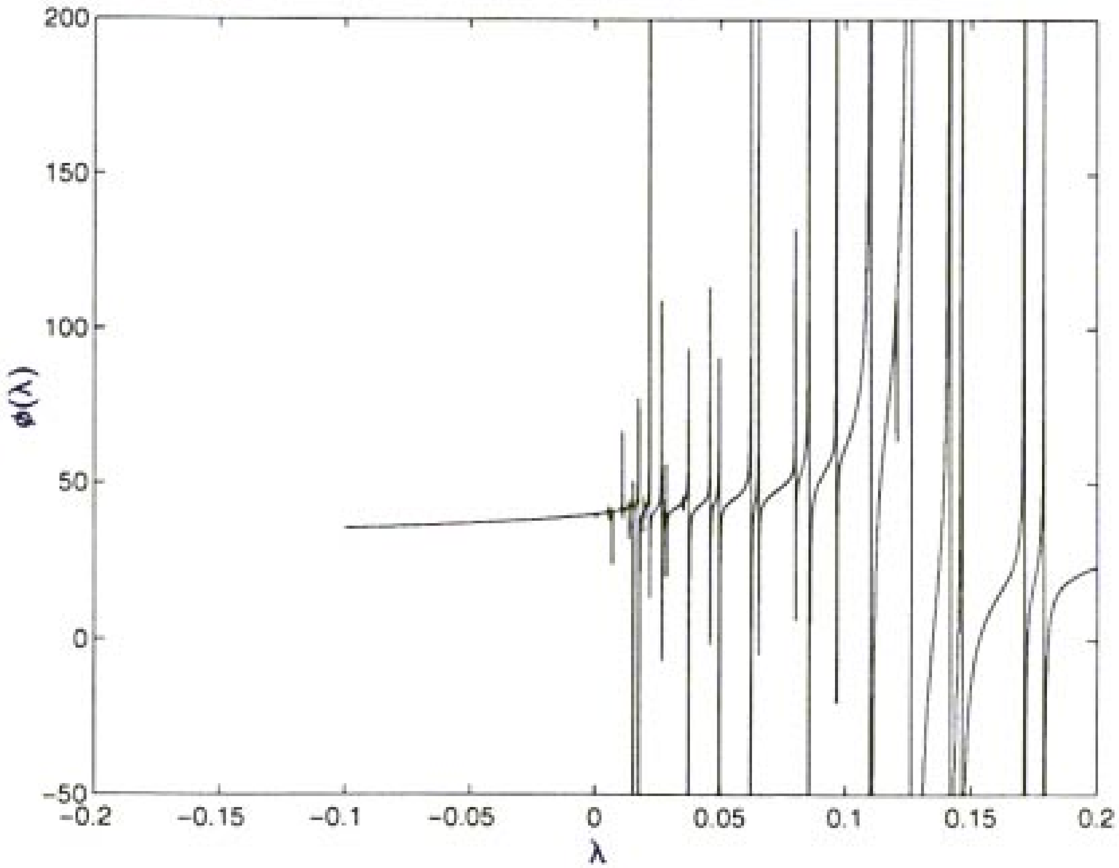


LSTRS in pictures - the (near) hard case

$$(\mathbf{H} - \lambda \mathbf{I})\mathbf{x} = -\mathbf{g}, \;\; \alpha - \lambda = \phi(\lambda) \approx -\mathbf{g^T x}, \;\; \phi'(\lambda) = \mathbf{x^T x}.$$



LSTRS in pictures - hard case in ill-posed problems



Algorithm

LSTRS - The Algorithm

Input: **H** $\in \mathbb{R}^{n\times n}$, symmetric; $g \in \mathbb{R}^n$; $\Delta > 0$; **tolerances** $(\varepsilon_\Delta, \varepsilon_\nu, \varepsilon_{HC}, \varepsilon_\alpha, \varepsilon_{Int})$.

Output: x_*, solution to TRS and Lagrange multiplier λ_*.

1. Initialization

1.1 Compute $\delta_U \geq \delta_1$, initialize α_U, initialize α_0

1.2 Compute **eigenpairs** $\{\lambda_1(\alpha_0), (\nu_1, u_1^T)^T\}$, $\{\lambda_i(\alpha_0), (\nu_2, u_2^T)^T\}$ of B_{α_0}

1.3 Initialize α_L, set $k = 0$

2. **repeat**

2.1 **Adjust** α_k (might need to compute **eigenpairs**)

2.2 **if** $\|g\||\nu_1| > \varepsilon_\nu \sqrt{1 - \nu_1{}^2}$ **then**

set $\lambda_k = \lambda_1(\alpha_k)$ and $x_k = \dfrac{u_1}{\nu_1}$, and update α_L or α_U.

else set $\lambda_k = \lambda_i(\alpha_k)$, $x_k = \dfrac{u_2}{\nu_2}$, and $\alpha_U = \alpha_k$

2.3 **Compute** α_{k+1} by 1-point $(k = 0)$ or 2-point **interpolation scheme**

2.4 **Safeguard** α_{k+1} and set $k = k + 1$

2.5 Compute **eigenpairs** $\{\lambda_1(\alpha_k), (\nu_1, u_1^T)^T\}$, $\{\lambda_i(\alpha_k), (\nu_2, u_2^T)^T\}$ of B_{α_k}

until **convergence**

LSTRS - The Algorithm

- **Update** α_k by rational interpolation
- **Adjust** α_k until eigenvector with desired structure is obtained
- **Safeguard** α_k using safeguarding interval
- **Tolerances**
- **Convergence** (Stopping Criteria)
- **H** (Hessian Matrix)
- **Eigensolver**

LSTRS - The Algorithm

Tolerances

$\varepsilon_{\boldsymbol{\Delta}}$	The desired relative accuracy in the norm of the trust-region solution. A boundary solution x satisfies $\frac{\lvert\|x\|-\Delta\rvert}{\Delta} \le \varepsilon_{\boldsymbol{\Delta}}$.
$\varepsilon_{\mathbf{HC}}$	The desired accuracy of a quasi-optimal solution in the hard case. A quasi-optimal solution $\hat{x}$ satisfies $\psi(x_*) \le \psi(\hat{x}) \le \varepsilon_{\mathbf{HC}}\psi(x_*)$, where $\psi(x) = \frac{1}{2}x^T Hx + g^T x$, and x_* is the true solution.
ε_{α}	The minimum relative length of the safeguarding interval for α. The interval is too small when $\lvert\alpha_U - \alpha_L\rvert \le \varepsilon_{\alpha} * \max\{\lvert\alpha_L\rvert, \lvert\alpha_U\rvert\}$.
$\varepsilon_{\mathbf{Int}}$	Used to declare that the smallest eigenvalue of B_α is positive in the test for an interior solution: $\lambda_1(\alpha)$ is considered positive if $\lambda_1(\alpha) > -\varepsilon_{\mathbf{Int}}$.
ε_{ν}	The minimum relative size of an eigenvector component. The component ν is small when $\lvert\nu\rvert \le \varepsilon_{\nu} \frac{\|u\|}{\|g\|}$.

LSTRS - The Algorithm

Stopping Criteria

1. Boundary Solution - $\varepsilon_{\boldsymbol{\Delta}}$
2. Interior Solution - $\varepsilon_{\mathbf{Int}}$
3. Quasi-Optimal Solution (**Near Hard Case !**) - $\varepsilon_{\mathbf{HC}}$
4. Safeguarding interval cannot be further decreased - ε_{α}
5. Maximum number of iterations reached

Software

LSTRS - The Software

Version: **1.2**

System: **MATLAB 6.0 or higher**

LSTRS - The Software

Front-end routine:	**lstrs**
LSTRS Iteration:	**lstrs_method**
Update of α_k:	**upd_param0, upd_paramk, interpol1, interpol2, inter_point**
Adjustment of α_k:	**adjust_alpha**
Safeguarding of α_k:	**safe_alpha1, safe_alphak, upd_alpha_safe**
Eigenproblems:	**b_epairs, eig_gateway, eigs_lstrs, eigs_lstrs_gateway, tcheigs_lstrs_gateway**
Stopping Criteria:	**convergence, boundary_sol, interior_sol, quasioptimal_sol**
Output:	**output**

LSTRS - The Software

General Call

```
[x,lambda,info,moreinfo] = ...
lstrs(H,g,delta,epsilon,eigensolver,lopts,Hpar,eigensolverpar)
```

Simplest Call

```
[x,lambda,info,moreinfo] = lstrs(H,g,delta);
```

LSTRS - The Software

Input Parameters

char:	**H, eigensolver**
double:	**H, g, delta**
struct:	**epsilon, lopts, Hpar, eigensolverpar**
function-handle:	**H, eigensolver**

Output Parameters

x:	solution vector
lambda:	Lagrange multiplier (Tikhonov parameter)
info:	exit condition
moreinfo:	other exit conditions, matrix-vector products, etc.

LSTRS Software: Key Features

- **Two options for Hessian Matrix:**
  - **H** (explicitly)
  - **Matrix-Vector Multiplication Routine**
- **Several options for Eigensolver:**
  - **eig**
  - **eigs_lstrs** (modified version of **eigs** that returns more information): this is MATLAB's interface to **ARPACK** (Implicitly Restarted Arnoldi Method)
  - **tcheigs_lstrs +Tchebyshev Spectral Transformation**
  - **user-provided**
- **Fixed Storage**

```
%
% File: simple.m
% A simple problem where the Hessian is the Identity matrix.
%
H = eye(50);
g = ones(50,1);
mu = -3;            % chosen arbitrarily
xexact = -ones(50,1)/(1-mu);
Delta = norm(xexact);
%
% The simplest possible calls to lstrs. Default values are used.
%
[x,lambda,info,moreinfo] = lstrs(H,g,Delta);

[x,lambda,info,moreinfo] = lstrs(@mv,g,Delta);
```

```
%
% File: mv.m
% A simple matrix-vector multiplication routine
% that computes the Identity matrix times a vector v
%

function [w] = mv(v,varargin)

w = v;
```

< M A T L A B >

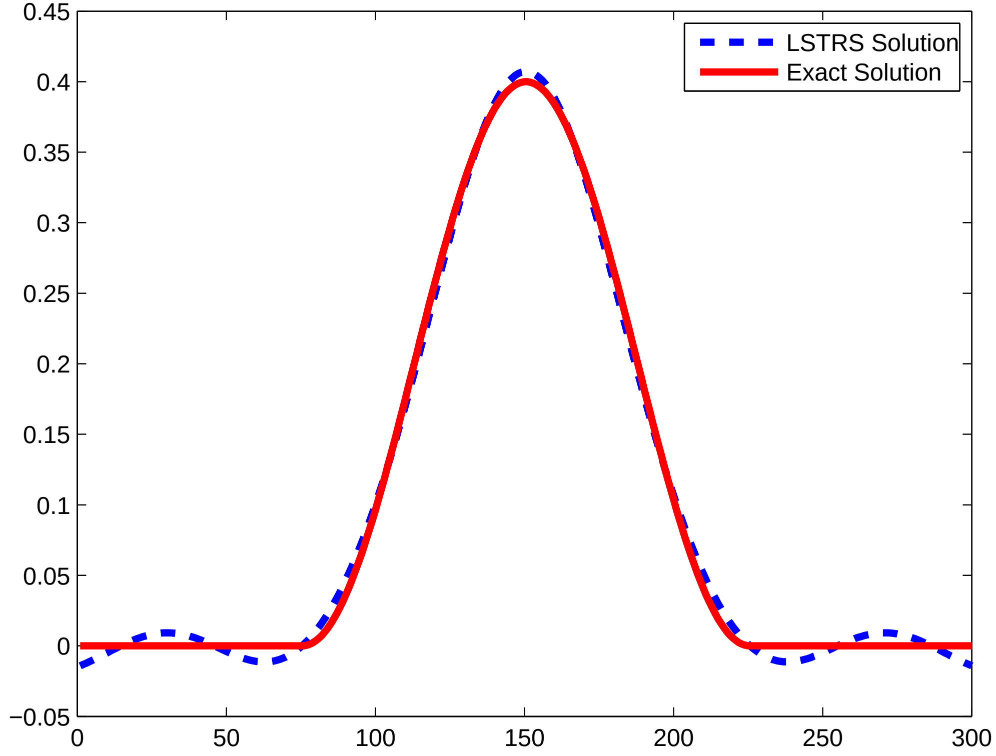
>> simple

```
Problem: no name available. Dimension: 50. Delta: 1.767767e+00

Eigensolver: eigs_lstrs_gateway

LSTRS iteration: 0
||x||: 9.317862e-01, lambda: -6.588723e+00
|||x||-Delta|/Delta: 4.729021e-01

LSTRS iteration: 1
||x||: 1.767767e+00, lambda: -3.000000e+00

|||x||-Delta|/Delta: 1.381681e-15


Number of LSTRS Iterations: 2
Number of calls to eigensolver: 2
Number of MV products: 18


(||x||-Delta)/Delta: 1.381681e-15

lambda: -3.000000e+00

||g + (H-lambda*I)x||/||g|| = 1.553836e-15

The vector x is a Boundary Solution
```

```
%
% File: regularization.m
% Computes a regularized solution to problem phillips from
% the Regularization Tools Package by P.C. Hansen
%
[A,b,xexact] = phillips(300);
atamvpar.A = A;
g = - A'*b;
Delta = norm(xexact);
lopts.name = 'phillips';   lopts.plot = 'y';  lopts.message_level = 2;
lopts.correction = 'n';     lopts.interior = 'n';
epar.k = 2;   epar.p = 7;      % for a total of 7 vectors (default)
[x,lambda,info,moreinfo] = ...
lstrs(@atamv,g,Delta,[ ],@tcheigs_lstrs_gateway,lopts,atamvpar,epar);
```

```
%
% File: atamv.m
% A matrix-vector multiplication routine
% that computes A'*A*v
% The matrix A must be a field of the structure atamvpar
%

function [w] = atamv(v,atamvpar)

w = atamvpar.A*v;
w = (w'*atamvpar.A)';
```

```
< M A T L A B >
>> regularization
Problem: phillips. Dimension: 300. Delta: 2.999927e+00
Eigensolver: tcheigs_lstrs_gateway

LSTRS iteration: 0
||x||: 8.327280e-01, lambda: -6.913002e+01
|||x||-Delta|/Delta: 7.224172e-01

LSTRS iteration: 1
||x||: 1.746167e+00, lambda: -1.768532e+01
|||x||-Delta|/Delta: 4.179302e-01

LSTRS iteration: 2
||x||: 2.935925e+00, lambda: -3.680399e-01
|||x||-Delta|/Delta: 2.133441e-02

LSTRS iteration: 3
||x||: 3.000547e+00, lambda: 1.883460e-03
|||x||-Delta|/Delta: 2.067913e-04

Number of LSTRS Iterations: 4
Number of calls to eigensolver: 5
Number of MV products: 342

(||x||-Delta)/Delta: 1.332300e-15
lambda: 1.904171e-03
||g + (H-lambda* I)x||/||g|| = 1.929542e-05
The vector x is a Quasi-optimal Solution
```



LSTRS Solution
Exact Solution
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
−0.05
0
50
100
150
200
250
300

Comparisons

Comparisons

- SSM - Sequential Subspace Method.
  Hager 2001.
- SDP - Semidefinite Programming approach.
  Rendl and Wolkowicz 1997, Fortin and Wolkowicz 2004.
- GLTR - Generalized Lanczos Trust Region method.
  Gould, Lucidi, Roma, and Toint 1999.

Average results for the 2-D Laplacian, $n = 1024$. Easy Case.

METHOD	MVP	STORAGE	$\frac{\|(H - \lambda I)x+g\|}{\|g\|}$
LSTRS	127.1	10	2.32×10^{-6}
SSM	67.3	10	9.53×10^{-7}
SSM_d	67.3	10	9.53×10^{-7}
SDP	595	10	3.17×10^{-5}
GLTR	81.6	41.3	8.56×10^{-6}

Average results for the 2-D Laplacian, $n = 1024$. Hard Case.

METHOD	MVP	STORAGE	$\frac{\|(H - \lambda I)x+g\|}{\|g\|}$
LSTRS	252.6	10	6.91×10^{-6}
SSM	377.9	10	1.42×10^{-6}
SSM_d	377.9	10	1.42×10^{-6}
SDP	2023.8	10	5.76×10^{-2}
GLTR	151.8	76.4	8.37×10^{-6}

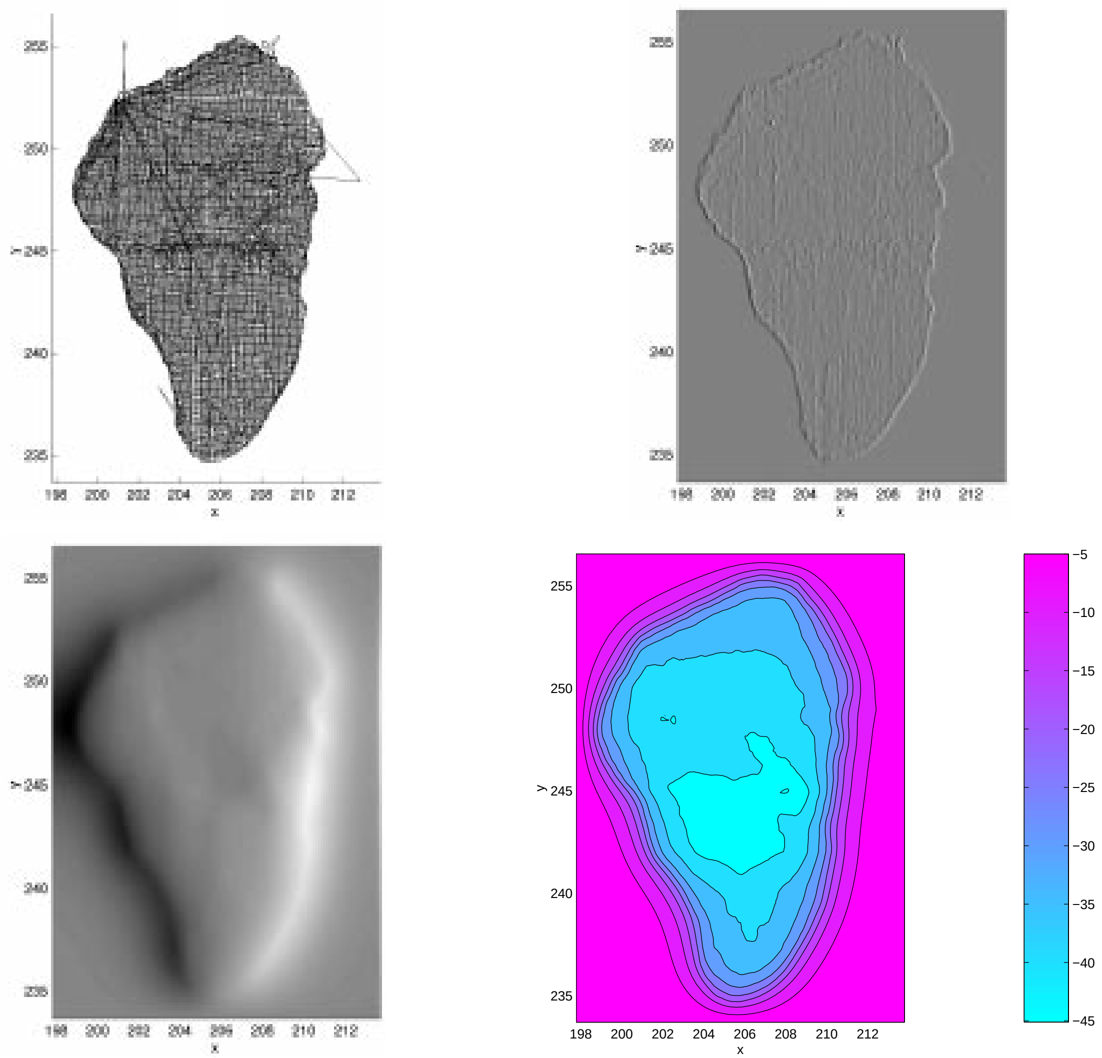
Inverse Heat Equation, $n = 1000$**. Mildly Ill-Posed.**

METHOD	MVP	STORAGE	$\frac{\|(H - \lambda I)x+g\|}{\|g\|}$	$\frac{\|x-x_{IP}\|}{\|x_{IP}\|}$
LSTRS	265	8	9.12×10^{-6}	6.13×10^{-4}
SSM	700	8	2.99×10^{-9}	2.41×10^{-4}
SSM_d	649	8	2.74×10^{-9}	4.57×10^{-4}
SDP	5700	8	2.73×10^{-7}	3.63×10^{-4}

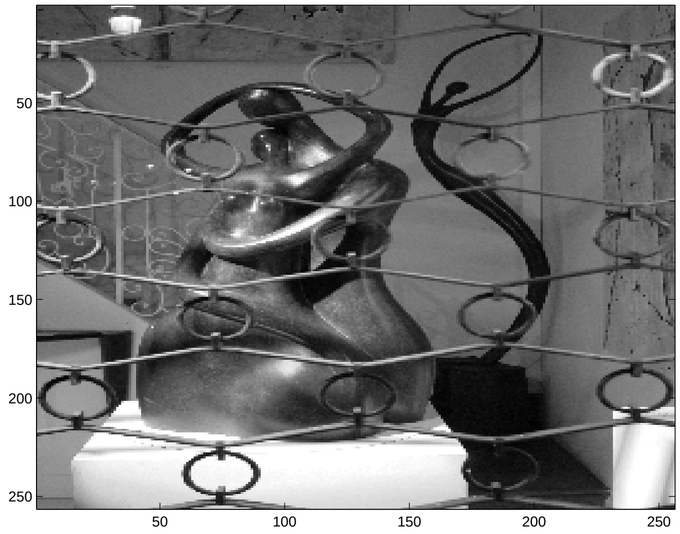
Inverse Heat Equation, $n = 1000$**. Severely Ill-Posed.**

METHOD	MVP	STORAGE	$\frac{\|(H - \lambda I)x+g\|}{\|g\|}$	$\frac{\|x-x_{IP}\|}{\|x_{IP}\|}$
LSTRS	552	8	7.05×10^{-6}	5.49×10^{-2}
SSM	512	8	1.81×10^{-7}	3.75×10^{-2}
SSM_d	215	8	2.04×10^{-7}	2.25×10^{-2}
SDP	4600	8	2.27×10^{-4}	2.08×10^{-1}

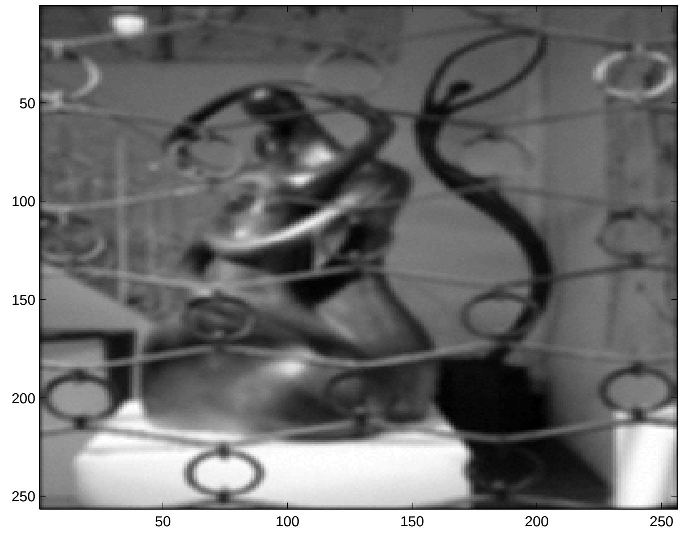
Applications



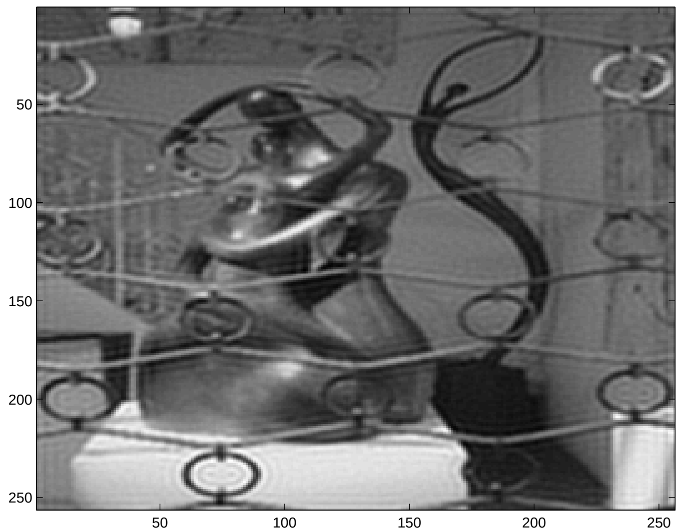
Bathymetry of the Sea of Galilee. Dimension: 40401. Vectors: 5. Matvecs: 206.



True image



Blurred and noisy image



LSTRS restoration

Dimension:	65536
Cost:	
7	Vectors
3	LSTRS iterations
201	Matvecs

REFERENCES

- M. Rojas, S. A. Santos and D. C. Sorensen. A New Matrix-Free Algorithm for the Large-Scale Trust-Region Subproblem. *SIAM Journal on Optimization*, 11(3): 611-646, 2000.
- M. Rojas, S. A. Santos and D. C. Sorensen. Algorithm xxx: LSTRS: MATLAB Software for Large-Scale Trust-Region Subproblems and Regularization. *ACM Transactions on Mathematical Software*, to appear.
- http://www.imm.dtu.dk/∼mr