

Algorithms and software for the solution and the regularization of large structured linear systems

G. Rodriguez

Dipartimento di Matematica e Informatica, Università di Cagliari
viale Merello 92, 09123 Cagliari, Italy

Workshop on Numerical Linear Algebra,
Internet and Large Scale Applications
Monopoli, 9–14 settembre 2007

smt: a Matlab structured matrices toolbox

in collab. with Michela Redivo Zaglia and Antonio Aricò

The idea behind `smt`

Extend Matlab with a computational framework for structured matrices, with the aim of making it

- easy to use and *Matlab-like*;
- transparent for the user;
- highly optimized for what concerns storage and complexity;
- easily extensible.

What's inside smt

- `circulant` and `Toeplitz` matrices
- matrix arithmetics (`BLAS`)
- `direct solver` for Toeplitz (square and LS)
- circulant `preconditioners` for iterative methods (`pcg`, etc.)
- test matrices (`smtgallery`)
- online documentation (`help`)
- integration with `sparse` and `single` data types
- mixed types operations and errors detection
- devices for changing standard behaviour and assign constants
- coded in `matlab` and `C` language (MEX interface)

Storage

Circulant matrices (`smcirc`)

- dimension n
- first column $\mathbf{c} = (c_0, \dots, c_{n-1})^T$
- eigenvalues $\delta = \text{fft}(\mathbf{c})$

Toeplitz matrices (`smtoepl`)

- dimensions m, n
- first row/column $\mathbf{t} = (t_{-n+1}, \dots, t_{m-1})^T$
- eigenvalues of *embedding* circulant

Algorithms

- **operators** for matrix arithmetics
 - **trivial**: +, -, .*, ./, etc.
 - **fast**: *, /, ^, etc.
- **functions**: abs, real, reshape, etc.
- **visualization and subindexing**
- the *eigenvalues* field is created at allocation
- it is updated when possible, otherwise it is recomputed

Tutorial

```
>> T=smtorp([5:-1:1],[5:9])
```

```
T =
```

5	6	7	8	9
4	5	6	7	8
3	4	5	6	7
2	3	4	5	6
1	2	3	4	5

Tutorial

```
>> smtconfig display compact  
>> T=smttoep([5:-1:1],[5:9])
```

T =

smttoep object with fields:

```
type: 'toeplitz'  
t: [9x1 double]  
dim1: 5  
dim2: 5  
cev: [16x1 double]
```

Tutorial

```
>> T=smtxep(rand(2000,1));  
>> A=full(T);  
>> whos A T
```

Name	Size	Bytes	Class	Attributes
A	2000x2000	32000000	double	
T	2000x2000	98180	smtxep	

Tutorial

```
>> T = smtgallery('gaussian',2000);  
>> A = full(T); b = T*ones(2000,1);  
>> tic, [x flag res iter] = pcg(T,b,1e-8,100); toc, iter  
Elapsed time is 1.800612 seconds.
```

```
iter =  
    42
```

```
>> tic, [x flag res iter] = pcg(A,b,1e-8,100); toc, iter  
Elapsed time is 11.757274 seconds.
```

```
iter =  
    42
```

Tutorial

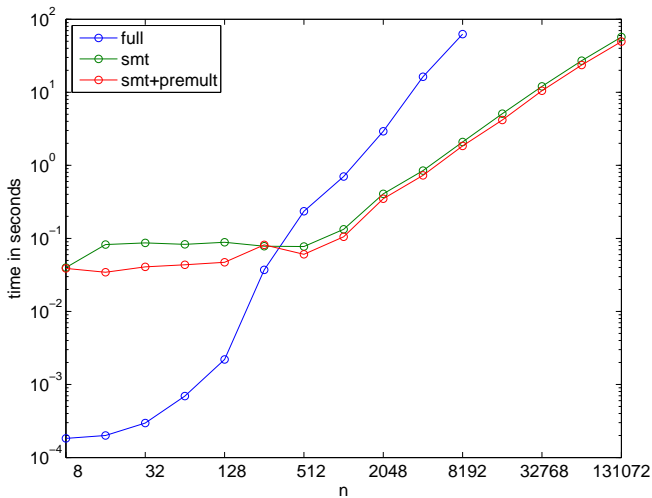
```
>> T = smtgallery('gaussian',2000);  
>> A = full(T); b = T*ones(2000,1);  
>> tic, [x flag res iter] = pcg(T,b,1e-8,100); toc, iter  
Elapsed time is 1.800612 seconds.
```

```
iter =  
    42
```

```
>> C = smtcpred('optimal',T);  
>> tic, [x flag res iter] = pcg(T,b,1e-8,100,C); toc, iter  
Elapsed time is 0.345862 seconds.
```

```
iter =  
    4
```

Speed test



time for 100 matrix-vector products

The library

Overloaded operators

<code>ctranspose</code>	<code>A'</code>	<code>plus</code>	<code>A+B</code>
<code>ldivide</code>	<code>A.\B</code>	<code>power</code>	<code>A.^2</code>
<code>minus</code>	<code>A-B</code>	<code>rdivide</code>	<code>A./B</code>
<code>mldivide</code>	<code>A\B</code>	<code>times</code>	<code>A.*B</code>
<code>mpower</code>	<code>A^2</code>	<code>transpose</code>	<code>A.'</code>
<code>mrdivide</code>	<code>A/B</code>	<code>uminus</code>	<code>-A</code>
<code>mtimes</code>	<code>A*B</code>	<code>uplus</code>	<code>+A</code>

The library

Overloaded functions

<code>abs</code>	absolute value	<code>isequal</code>	array comparison
<code>angle</code>	phase angle	<code>isfloat</code>	true for FP arrays
<code>ceil</code>	round towards ∞	<code>isreal</code>	true for real arrays
<code>conj</code>	complex conjugate	<code>length</code>	length of vector
<code>det</code>	determinant	<code>prod</code>	product of elements
<code>diag</code>	diagonal (of a) matrix	<code>real</code>	real part
<code>double</code>	convert to double	<code>reshape</code>	change size
<code>eig</code>	eigenvalues/vectors	<code>round</code>	round argument
<code>fix</code>	round towards zero	<code>sign</code>	signum function
<code>floor</code>	round towards $-\infty$	<code>single</code>	convert to single
<code>full</code>	convert to full matrix	<code>size</code>	size of array
<code>imag</code>	imaginary part	<code>sum</code>	sum of elements
<code>inv</code>	matrix inverse	<code>tril</code>	lower triangular part
<code>isa</code>	check object class	<code>triu</code>	upper triangular part
<code>isempty</code>	true for empty array		

The library

Visualization and subindexing

<code>display</code>	display array	<code>subsasgn</code>	subscripted assignment
<code>end</code>	last index	<code>subsindex</code>	subscript index
<code>get</code>	get object properties	<code>subsref</code>	subscripted reference

Toeplitz solver

1. transform into a *Cauchy-like* linear system

$$T\mathbf{x} = \mathbf{b} \quad \rightarrow \quad C\mathbf{y} = \mathbf{c} \text{ (Cauchy)}$$

2. compute solution as a Schur complement

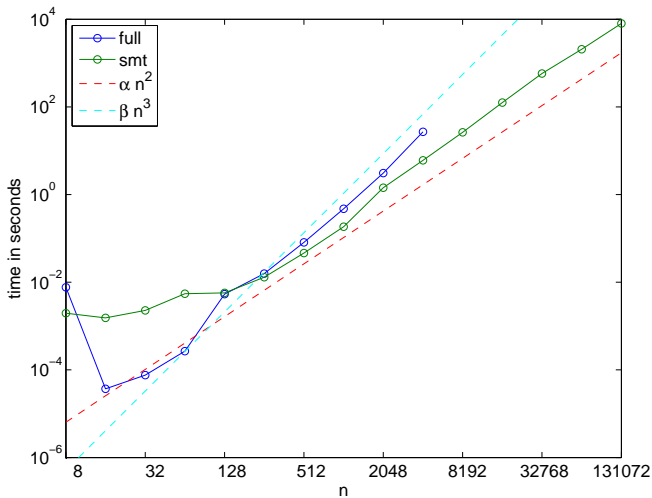
$$\mathbf{y} = \left(\left[\begin{array}{c|c} C & \mathbf{c} \\ \hline -I_n & 0 \end{array} \right] / C \right)$$

using **partial pivoting** (coded in C/MEX).

3. transform back the solution

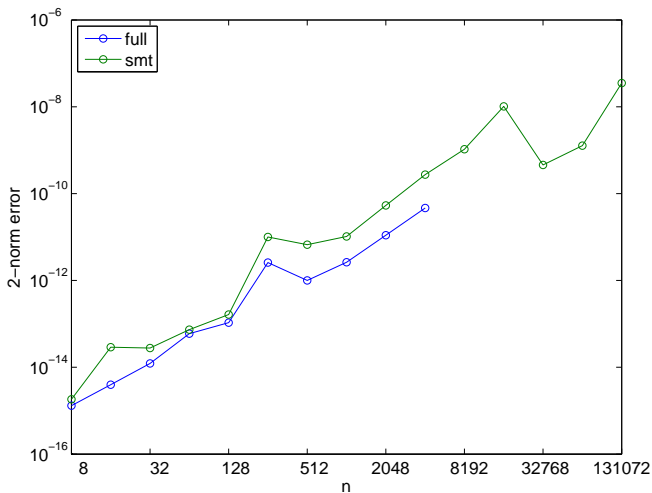
$$\mathbf{y} \quad \rightarrow \quad \mathbf{x}$$

Speed test



time for solution of 1 Toeplitz linear systems

Accuracy



solution of random Toeplitz linear systems

Toeplitz least-squares solver

The method is based on the computation of a Schur complement

$$\mathbf{x}_{LS} = \left(\left[\begin{array}{cc|c} I_m & A & 0 \\ A^* & 0 & A^* \mathbf{b} \\ \hline 0 & I_n & 0 \end{array} \right] / \left[\begin{array}{cc} I_m & A \\ A^* & 0 \end{array} \right] \right)$$

by an optimized algorithm [SIMAX 2006].

It works on full rank matrices, and has been translated into C/MEX.

An extension for rank-deficient matrices is *under construction*.

Error estimates for linear systems with applications to regularization

in collab. with Claude Brezinski and Sebastiano Seatzu

Error estimates for linear systems

Consider the linear system

$$A\mathbf{x} = \mathbf{b}$$

and let $\mathbf{e} = \mathbf{x} - \mathbf{x}^*$ and $\mathbf{r} = \mathbf{b} - A\mathbf{x}^*$, being \mathbf{x}^* an approx. of \mathbf{x} .

Then

$$\frac{\|\mathbf{r}\|}{\|A\|} \leq \|\mathbf{e}\| \leq \|A^{-1}\| \cdot \|\mathbf{r}\|.$$

Error estimates for linear systems

Consider the linear system

$$A\mathbf{x} = \mathbf{b}$$

and let $\mathbf{e} = \mathbf{x} - \mathbf{x}^*$ and $\mathbf{r} = \mathbf{b} - A\mathbf{x}^*$, being \mathbf{x}^* an approx. of \mathbf{x} .

Auchmuty [1992] showed that

$$\frac{\|\mathbf{r}\|^2}{\|A^T \mathbf{r}\|} \lesssim \|\mathbf{e}\|.$$

This estimated was further studied and extended by Brezinski [1999] and Galantai [2001].

Error estimates for linear systems

Let us compute

$$c_0 = \|\mathbf{r}\|^2$$

$$c_1 = \mathbf{r}^T \mathbf{A} \mathbf{r}$$

$$c_2 = \|\mathbf{A} \mathbf{r}\|^2$$

$$\tilde{c}_2 = \|\mathbf{A}^T \mathbf{r}\|^2$$

Then, for any $\nu \in \mathbb{R}$,

$$\|\mathbf{e}\|^2 \simeq e_\nu^2 = c_0^{\nu-1} (c_1^2)^{3-\nu} c_2^{\nu-4}$$

$$\|\mathbf{e}\|^2 \simeq \tilde{e}_\nu^2 = c_0^{\nu-1} (c_1^2)^{3-\nu} \tilde{c}_2^{\nu-4}$$

Error estimates for linear systems

Let us compute

$$c_0 = \|\mathbf{r}\|^2$$

$$c_1 = \mathbf{r}^T \mathbf{A} \mathbf{r}$$

$$c_2 = \|\mathbf{A} \mathbf{r}\|^2$$

$$\tilde{c}_2 = \|\mathbf{A}^T \mathbf{r}\|^2$$

Then, for any $\nu \in \mathbb{R}$,

$$\|\mathbf{e}\|^2 \simeq e_\nu^2 = c_0^{\nu-1} (c_1^2)^{3-\nu} c_2^{\nu-4}$$

$$\|\mathbf{e}\|^2 \simeq \tilde{e}_\nu^2 = c_0^{\nu-1} (c_1^2)^{3-\nu} \tilde{c}_2^{\nu-4}$$

For example

$$e_3 = \frac{\|\mathbf{r}\|^2}{\|\mathbf{A} \mathbf{r}\|}, \quad \tilde{e}_3 = \frac{\|\mathbf{r}\|^2}{\|\mathbf{A}^T \mathbf{r}\|} \quad [\text{Auchmuty}]$$

Error estimates for linear systems

To obtain the estimates

$$e_\nu^2 = c_0^{\nu-1} (c_1^2)^{3-\nu} c_2^{\nu-4}$$

$$\tilde{e}_\nu^2 = c_0^{\nu-1} (c_1^2)^{3-\nu} \tilde{c}_2^{\nu-4}$$

we

- express c_i by means of the **SVD** of A
- **approximate** the c_i by truncation
- compute some parameters by **interpolation**
- **extrapolate** the result

In this way we are able to obtain a family of estimates for

$$\text{the 2-norm error } \|\mathbf{e}\|_2 = \sqrt{\mathbf{e}^T \mathbf{e}} \quad (e_\nu, \tilde{e}_\nu)$$

$$\text{the } A\text{-norm error } \|\mathbf{e}\|_A = \sqrt{\mathbf{e}^T A \mathbf{e}} \quad (\hat{e}_\nu)$$

Main properties of error estimates

1. let $\rho = c_0 c_2 / c_1^2 \geq 1$, then

$$\frac{\rho^{(3-\nu)/2}}{\text{cond}(A)} \cdot e_\nu \leq \|\mathbf{e}\| \leq \text{cond}(A) \cdot \rho^{(3-\nu)/2} \cdot e_\nu$$

2. e_ν and \tilde{e}_ν are increasing functions of ν

$$e_\nu \leq e_{\nu'}, \quad \tilde{e}_\nu \leq \tilde{e}_{\nu'}, \quad \text{if } \nu \leq \nu'$$

3. generalization of Auchmuty's inequality

$$\tilde{e}_\nu^2 \leq \|\mathbf{e}\|^2, \quad \forall \nu \leq 3$$

4. there exists $\tilde{\nu} \geq 3$ such that $\tilde{e}_{\tilde{\nu}} = \|\mathbf{e}\|$

$$\tilde{\nu} = 2 \ln(\|\mathbf{e}\| / \tilde{e}_0) / \ln \tilde{\rho}$$

Large ill-conditioned linear systems

Consider the linear system

$$A\mathbf{x} = \mathbf{b} + \boldsymbol{\epsilon}$$

with $A \in \mathcal{M}_{n \times n}$ severely ill-conditioned ($\text{cond}(A) \gtrsim \|\boldsymbol{\epsilon}\|^{-1}$).

Regularization methods

- Truncated SVD
- Tikhonov regularization
- iterative methods (e.g. CG)

all depend upon a **regularization parameter**.

Large ill-conditioned linear systems

Consider the linear system

$$A\mathbf{x} = \mathbf{b} + \boldsymbol{\epsilon}$$

with $A \in \mathcal{M}_{n \times n}$ severely ill-conditioned ($\text{cond}(A) \gtrsim \|\boldsymbol{\epsilon}\|^{-1}$).

Regularization methods for large (structured) systems

- Truncated SVD
- Tikhonov regularization with fast or iterative solver
- iterative methods (e.g. CG) with fast matrix-vector product

all depend upon a regularization parameter.

Large ill-conditioned linear systems

Regularization methods for large (structured) systems

- Truncated SVD
- Tikhonov regularization with fast or iterative solver
- iterative methods (e.g. CG) with fast matrix-vector product

all depend upon a regularization parameter.

Parameter estimation techniques

- Morozov, Mallows (require info on $\|\epsilon\|$)
- Generalized Cross Validation (GCV)
- L-curve
- error estimates e_ν, \tilde{e}_ν

Tikhonov regularization

$$J(\mathbf{x}, \lambda) = \|\mathbf{Ax} - \mathbf{b}\|^2 + \lambda^2 \|\mathbf{Hx}\|^2 \quad \rightarrow \quad \mathbf{x}_\lambda = \arg \min_{\mathbf{x} \in \mathbb{R}^n} J(\mathbf{x}, \lambda)$$

The regularized vector \mathbf{x}_λ is the solution of the system

$$(\mathbf{C} + \lambda^2 \mathbf{E})\mathbf{x}_\lambda = \mathbf{A}^T \mathbf{b}$$

where $\mathbf{C} = \mathbf{A}^T \mathbf{A}$ and $\mathbf{E} = \mathbf{H}^T \mathbf{H}$.

1. **Diagonalize** $J(\mathbf{x}, \lambda)$ by employing $\text{svd}(A)$ or $\text{gsvd}(A, H)$
(this simplifies also the GCV function)

Tikhonov regularization

$$J(\mathbf{x}, \lambda) = \|\mathbf{Ax} - \mathbf{b}\|^2 + \lambda^2 \|\mathbf{Hx}\|^2 \quad \rightarrow \quad \mathbf{x}_\lambda = \arg \min_{\mathbf{x} \in \mathbb{R}^n} J(\mathbf{x}, \lambda)$$

The regularized vector \mathbf{x}_λ is the solution of the system

$$(\mathbf{C} + \lambda^2 \mathbf{E})\mathbf{x}_\lambda = \mathbf{A}^T \mathbf{b}$$

where $\mathbf{C} = \mathbf{A}^T \mathbf{A}$ and $\mathbf{E} = \mathbf{H}^T \mathbf{H}$.

2. Solve **normal system** by CG for each λ

Tikhonov regularization

$$J(\mathbf{x}, \lambda) = \|\mathbf{Ax} - \mathbf{b}\|^2 + \lambda^2 \|\mathbf{Hx}\|^2 \quad \rightarrow \quad \mathbf{x}_\lambda = \arg \min_{\mathbf{x} \in \mathbb{R}^n} J(\mathbf{x}, \lambda)$$

The regularized vector \mathbf{x}_λ is the solution of the system

$$(\mathbf{C} + \lambda^2 \mathbf{E})\mathbf{x}_\lambda = \mathbf{A}^T \mathbf{b}$$

where $\mathbf{C} = \mathbf{A}^T \mathbf{A}$ and $\mathbf{E} = \mathbf{H}^T \mathbf{H}$.

3. Compute \mathbf{x}_λ by a Schur complement (if displacement struct.)

$$\mathbf{x}_\lambda = \left(\left[\begin{array}{ccc|c} I_m & 0 & \mathbf{A} & 0 \\ 0 & I_p & \lambda \mathbf{H} & 0 \\ \hline \mathbf{A}^* & \lambda \mathbf{H}^* & 0 & \mathbf{A}^* \\ 0 & 0 & I_n & 0 \end{array} \right] / \left[\begin{array}{ccc} I_m & 0 & \mathbf{A} \\ 0 & I_p & \lambda \mathbf{H} \\ \mathbf{A}^* & \lambda \mathbf{H}^* & 0 \end{array} \right] \right)$$

(this approach is useful for GCV also)

Displacement structure and GCV

Generalized cross validation (GCV) consists of minimizing the function

$$V(\lambda) = \frac{\frac{1}{m} \|(I - A(\lambda))\mathbf{b}\|^2}{\left[\frac{1}{m} \text{Trace}(I - A(\lambda))\right]^2}$$

where the *influence matrix* is given by

$$A(\lambda) = A(A^*A + \lambda^2 H^*H)^{-1}A^*.$$

The GCV function is usually computed by the SVD (GSVD). An alternative algorithm could be constructed by expressing

$$A(\lambda) = \left(\left[\begin{array}{ccc|c} I_m & 0 & A & 0 \\ 0 & I_p & \lambda H & 0 \\ \hline A^* & \lambda H^* & 0 & A^* \\ 0 & 0 & A & 0 \end{array} \right] / \left[\begin{array}{ccc} I_m & 0 & A \\ 0 & I_p & \lambda H \\ A^* & \lambda H^* & 0 \end{array} \right] \right)$$

Tikhonov regularization

The regularized vector \mathbf{x}_λ is the solution of the system

$$(C + \lambda^2 E)\mathbf{x}_\lambda = A^T \mathbf{b}$$

where $C = A^T A$ and $E = H^T H$.

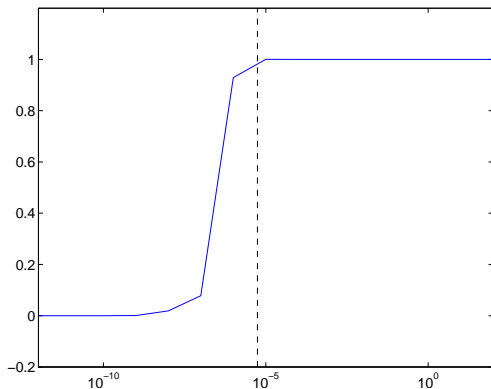
This implies

$$A^T \mathbf{r}_\lambda = \lambda^2 E \mathbf{x}_\lambda,$$

with $\mathbf{r}_\lambda = \mathbf{b} - A\mathbf{x}_\lambda$, so that

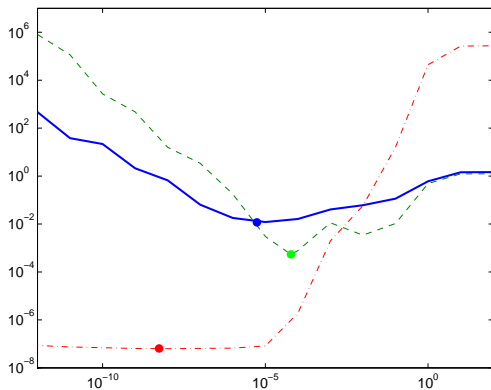
- $\frac{\lambda^2 \|E\mathbf{x}_\lambda\|}{\|A^T \mathbf{r}_\lambda\|} = 1$
- $\tilde{\epsilon}_\nu^2 = \|\mathbf{r}_\lambda\|^{2\nu-2} (\mathbf{r}_\lambda^T E \mathbf{x}_\lambda)^{6-2\nu} \|E\mathbf{x}_\lambda\|^{2\nu-8} \lambda^{-4}$

Is x_λ well computed?



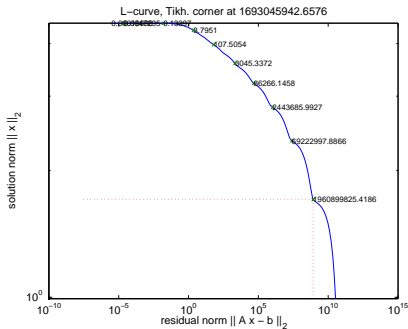
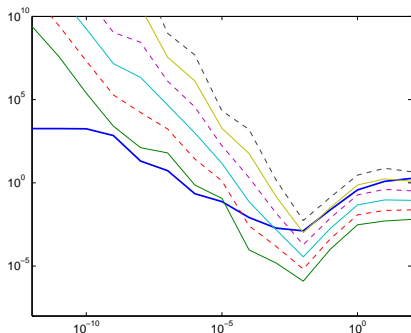
Ratio $\lambda^2 \|E x_\lambda\| / \|A^T r_\lambda\|$ vs. λ
Baart matrix, $n = 20$, $\mathbf{x} = (1, \dots, 1)^T$, $\sigma = 10^{-8}$

Comparison with GCV



Graph of $\|e\|$, \tilde{e}_3 and $10^{10} * GCV$ vs. λ
Baart matrix, $n = 20$, $\mathbf{x} = (1, \dots, 1)^T$, $\sigma = 10^{-8}$

Comparison with L-curve



Graph of $\|e\|$, \tilde{e}_1 , \tilde{e}_2 , \dots , \tilde{e}_6 and L-curve
 Pascal matrix, $n = 20$, $\mathbf{x} = (1, \dots, 1)^T$, $\sigma = 10^{-8}$

Massive experimentation: 5544 problems

- 12 matrices
- 7 test solutions
- 2 dimensions $n = 20, 100$
- 3 regularization matrices $H = I, D_1, D_2$
- 3 noise level $\sigma = 0, 10^{-8}, 10^{-4}$
- 5 realizations of the noise

Quality index:

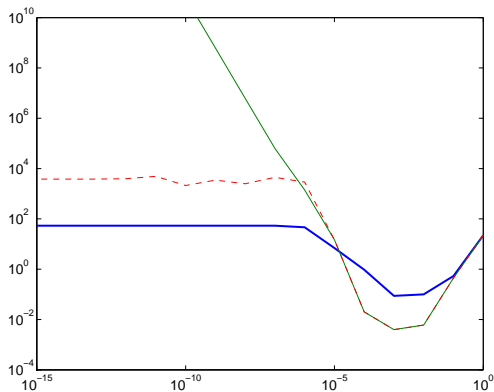
$$Q(\lambda) = \frac{\text{error}_{\text{estimated } \lambda}}{\text{error}_{\text{optimal } \lambda}} \geq 1$$

small failure: $Q(\lambda) \in [10^2, 10^4)$ *big failure:* $Q(\lambda) \geq 10^4$

Massive experimentation: 5544 problems

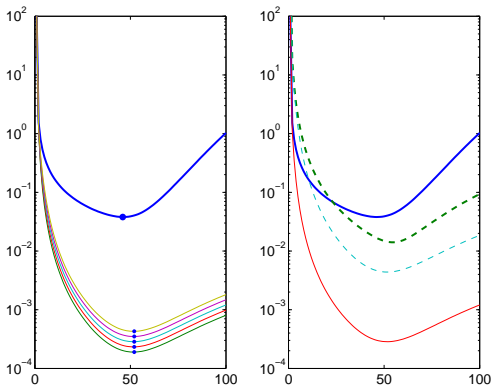
Matrix	Cond.	\tilde{e}_3		L-curve		GCV	
		$\in / > 10^4$	$\in / > 10^4$	$\in / > 10^4$	$\in / > 10^4$	$\in / > 10^4$	$\in / > 10^4$
Baart	$2.0 \cdot 10^{17}$	4	0	13	50	30	24
Heat(1)	$1.0 \cdot 10^{20}$	22	27	48	58	26	29
Hilbert	$3.1 \cdot 10^{18}$	4	0	17	51	2	39
Ilaplace	$9.2 \cdot 10^{30}$	12	0	13	64	16	28
Lotkin	$2.1 \cdot 10^{19}$	8	0	24	67	5	33
Moler	$1.7 \cdot 10^{13}$	134	58	131	59	106	58
Pascal	$1.2 \cdot 10^{20}$	12	77	94	263	3	154
Phillips	$4.0 \cdot 10^{03}$	85	32	120	60	68	26
Prolate	$5.6 \cdot 10^{13}$	1	0	91	53	13	43
Random	$2.8 \cdot 10^{02}$	34	2	96	144	68	141
Shaw	$9.9 \cdot 10^{15}$	5	0	46	49	40	9
Wing	$1.7 \cdot 10^{19}$	2	0	11	49	20	54
TOTAL		323	196	704	967	397	638

Tikhonov/CG



Graph of $\|e\|$, Tikhonov- \tilde{e}_2 and standard- \tilde{e}_2
Prolate matrix, $n = 2000$, $\mathbf{x} = (1, \dots, 1)^T$, $\sigma = 10^{-4}$

Regularizing CG

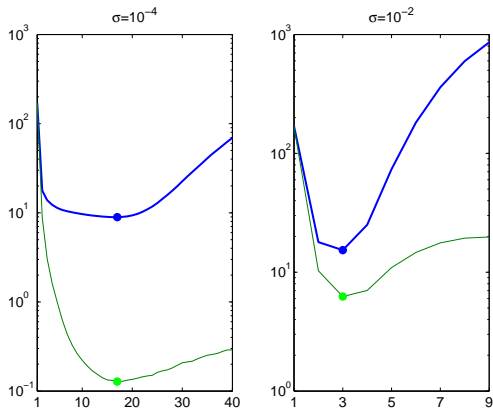


$\|\mathbf{e}\|, \tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, \dots, \tilde{\mathbf{e}}_5$

$\|\mathbf{e}\|, \tilde{\mathbf{e}}_3, \|\mathbf{e}\|_A$ and $\hat{\mathbf{e}}_3$

Gaussian matrix ($\rho = .01$), $n = 10000$, $\mathbf{x} = (1, \dots, 1)^T$, $\sigma = 10^{-4}$

Regularizing CG for an image deblurring problem



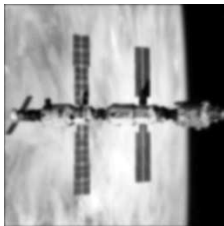
Graph of $\|e\|$ and e_3
Gaussian blur, size 256×256

Regularizing CG for an image deblurring problem

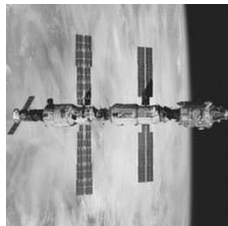
test image



blurred ($\sigma=10^{-4}$)



recovered ($k=17$)



Gaussian blur, size 256×256