Regularization Parameter Tool: User's Guide

Uno Hämarik, Reimo Palm, Toomas Raus

May 12, 2015

Contents

1 Using the tool

'Regularization Parameter Tool' is a tool intended for analysis of accuracy of regularization parameter choice rules and for solving discrete ill-posed problems by Tikhonov method. Traditionally, the most used parameter choice rule is the discrepancy principle but in some settings other rules may be preferable due to their characteristics that are better suited to a particular problem. The tool allows users to compare and test the performance of different rules on a standard set of problems or on their own problems, displaying the results in the form of graphs and tables.

The interaction with the tool is implemented as the following 4 interfaces:

1. *Solve a test problem using manually selected rules* – select a model problem and rule(s); exact solution known

2. *Solve own problem using manually selected rules* – upload own problem and select rule(s); exact solution not known

3. *Solve a test problem using approximate information about the noise level* – select a model problem, the tool solves it, choosing the parameter automatically

4. *Solve own problem using approximate information about the noise level* – upload own problem, the tool solves it, choosing the parameter automatically

Based on solving information given by the user, the tool computes and displays: a) graphs of approximate solutions, b) a table with numerical indicators and other data, and c) graphs of functions used in rules.

1.1 The problem

Every interface accepts user upload, which must be a standard Matlab `mat`-file, containing a saved variable named `A` plus either a variable named `x` (interfaces 1 and 3) or a variable named `y` (interfaces 2 and 4). Here, `A` is the system matrix, `x` is the exact solution and `y` is the right hand side. The tool solves the linear system $Ax = y$, so A and x are required for the model problem (where exact solution is known), while A and y are required for the "real" problem (where exact solution is not known).

The model problem in interfaces 1 and 3 can also be given by selecting matrix A and solution x from respective lists. The tool relies on the set of test problems from Hansen's Regularization Tools [2], complementing them with problems used by Brezinski, Rodriguez and Seatzu [1]. Tables 1 and 2 give an overview of the set of test problems. The matrix and the solution can be selected separately: it is possible to combine the matrix of one problem with the solution of another problem. Also, it is possible to change the parameters, which control the generation of the matrix or the solution, by changing the numbers in text boxes next to the names. The text boxes are prefilled with standard values of these parameters.

Each problem has the form of system of linear equation with square matrix. The size of the matrix can be given by entering the value for 'Problem size'. Default is 100; due to computing time the size cannot be too large. When the matrix A and the solution x have been generated, the exact right hand side y is computed as $y = Ax$.

It is possible to modify the generated problem in certain ways. If a nonzero value p is entered in the text box 'Additional smoothness' (interfaces 1 and 3), then the exact solution x of $Ax = y$ is replaced by $(A^*A)^{p/2}x$ (and the right hand side modified accordingly). This allows investigation of behaviour of rules when the solution of the problem is smooth.

Checking 'Normalize problem' (all interfaces) means that after A and y are generated or read, the problem $Ax = y$ is scaled in such a way that the Euclidean norms of A and y are 1, that is, the new problem $A'x' = y'$ satisfies $A' = A \,/\, \|A\|$, $y' = y \,/\, \|y\|$, and $x' = x \cdot \|A\| \,/\, \|y\|$.

Table 1: Test problems of Hansen [2] (problems 1–10) and Brezinski-Rodriguez-Seatzu [1] (problems 11–16) used by the web tool, with condition numbers of the matrix at problem size 100.

Nr	Problem	cond_{100}	selfadj	Description
1	baart	5e+17	no	(Artificial) Fredholm integral equation of the first kind
2	deriv2	1e+4	yes	Computation of the second derivative
3	foxgood	1e+19	yes	A problem that does not satisfy the discrete Picard condition
4	gravity	3e+19	yes	A gravity surveying problem
5	heat	2e+38	no	Inverse heat equation
6	ilaplace	9e+32	no	Inverse Laplace transform
7	phillips	2e+6	yes	An example problem by Phillips
8	shaw	5e+18	yes	An image reconstruction problem
9	spikes	3e+19	no	Test problem whose solution is a pulse train of spikes
10	wing	1e+20	no	Fredholm integral equation with discontinuous solution
11	gauss	6e+18	yes	Test problem with Gauss matrix $a_{ij} = \sqrt{\frac{\pi}{2\sigma}} e^{-\frac{\sigma}{2(i-j)^2}}$ with $\sigma = 0.01$
12	hilbert	4e+19	yes	Test problem with Hilbert matrix $a_{ij} = \frac{1}{i+j-1}$
13	lotkin	2e+21	no	Test problem with Lotkin matrix (same as Hilbert matrix, except $a_{1j} = 1$)
14	moler	2e+4	yes	Test problem with Moler matrix $A = B^T B$, $b_{ii} = 1$, $b_{ij} = \alpha$ for $i < j$, $b_{ij} = 0$ for $i > j$
15	pascal	1e+60	yes	Test problem with Pascal matrix $a_{ij} = \binom{i+j-2}{i-1}$
16	prolate	1e+17	yes	Test problem with a symmetric, ill-conditioned Toeplitz matrix

Table 2: Solution vectors for test problems of [1]. Here n is problem size and $i = 1, \dots, n$.

Description	$\overline{x}_i$	Description	$\overline{x}_i$
constant	1	sinusoidal	$\sin \frac{2\pi(i-1)}{n}$
linear	$\frac{i}{n}$	linear+sinusoidal	$\frac{i}{n} + \frac{1}{4} \sin \frac{2\pi(i-1)}{n}$
quadratic	$\left(\frac{i - \lfloor \frac{n}{2} \rfloor}{\lceil \frac{n}{2} \rceil}\right)^2$	step function	$\begin{cases} 0, & \text{if } i \le \lfloor \frac{n}{2} \rfloor \\ 1, & \text{if } i > \lfloor \frac{n}{2} \rfloor \end{cases}$

1.2 Noise level

In model problems (interfaces 1 and 3) the noise is added by the computer. Based on exact right hand side y and exact matrix A their noisy variants are generated by adding a random vector or a random matrix to them, whose elements have the selected distribution. Each noise level can be given as relative or absolute, and as an actual (true) noise level or a standard distribution. Noise of the right hand side can be formed as correlated noise by entering a nonzero value (between -1 and 1) for 'Correlation coefficient'. This gives the correlation coefficient of adjacent elements of the noise vector.

It is also possible to solve the problem using several noise vector / noise matrix combinations at once. The value of 'Number of noise vectors' determines how many times the noise is generated (how many times the problem is solved). If this number is larger than 1, then the output format of figures and tables is modified to accommodate the additional information.

1.3 Information about noise

In interfaces 1 and 2 three types of information about noise can be selected: 1) upper bounds δ and η of the noise vector are known, i.e. $\|y - y_*\| \leq \delta$, $\|A - A_*\| \leq \eta$; 2) variances σ^2 and σ_η^2 of the noise are known, i.e. $\mathbb{E}(y^{(i)} - y_*^{(i)})^2 = \sigma^2$, $\mathbb{E}(A^{(ij)} - A_*^{(ij)})^2 = \sigma_\eta^2$; 3) no information about noise is known. Noise information can be given as relative or absolute noise level. Note that this is what we say we *know* about noise, the actual noise level may be different.

In interfaces 3 and 4, upper bounds δ_0, η_0 and likely lower bounds δ_1 and η_1 of the noise of right hand side and matrix, respectively, should be entered. Upper bound means that the noise level is certainly not larger than this bound. Likely lower bound means that, for example, with probability 95% the noise level is larger than this bound.

1.4 Parameter choice rules

Rules which use delta (delta-rules) are on the left and rules which don't use delta (heuristic rules) are on the right. Delta-rules are classified as unstable rules (that tend to fail if the assumed noise level is less than the actual noise level) and stable rules (that allow underestimation of the noise level to some extent). If no information about the noise is known, then delta-rules produce no output. Heuristic rules don't use noise information regardless of what is entered under information about noise.

Currently the rules take into account only the noise level of the right hand side.

1.5 Technical

The search of parameters is restricted to the interval $[\alpha_M, \alpha_0]$; the default values $[10^{-18}, 1]$ are suitable for normalized problems. All computations are performed on the sequence $\{\alpha_0 q^i : i = 0, 1, \ldots\}$ of α's from this interval, starting from α_0.

1.6 Output

See the first part (theoretical aspects) of the documentation.

2 Implementation

The tool is implemented as a collection of functions in m-files. In addition to these, scripts named `progparam*.m` and `solve*.m` are available, which allow generation of output on a local computer for all four interfaces. To run the scripts locally, set the parameters in the file `progparam*.m` and run it; after setting the values of parameters, this script calls the main script `solve*.m`. The main script sends the output to the standard output in html format. This output can be redirected to a file and previewed in web browser.

The functions are intended to work with both full matrices and with matrices in diagonal form. In the latter case the "matrix" should be the column vector of eigenvalues. Where necessary, the functions check the number of columns of matrix and apply either matrix or vector operations to compute the values of expressions. All local scripts `solve*.m` compute singular value decomposition of the matrix and use these functions in vector form.

2.1 addmatrixnoise

```
A_noisy = addmatrixnoise(A, matrixnoisename,
    matrixnoiselevel, matrixnoisescale, matrixnoisetype)
```

Adds noise to the matrix.

Noise matrix elements can be either normally or uniformly distributed, and have the expectation 0. The generated noise matrix M satisfies $\|M\| = \eta$ or $\mathrm{var}(M^{(ij)}) = \sigma_\eta^2$, depending on wheteher the requested noise level is actual noise level η or standard deviation σ_η^2.

Parameters

`A` – Initial matrix.

`matrixnoisename` – Name of the noise distribution. Possible values: `"normal"` and `"uniform"`.

`matrixnoiselevel` – Numerical value of the noise level.

`matrixnoisescale` – Name of scaling of the noise level. Possible values: `"absolute"` if the noise level is to be taken as absolute noise level, `"relative"` if the noise level given is relative.

`matrixnoisetype` – Name of noise level type. Possible values: `"actual"` if actual noise level is given, `"std"` if standard deviation is given.

Returns

`A_noisy` – Matrix with noise added.

2.2 addvectornoise

```
y_noisy = addvectornoise(y, rhsnoisename, rhsnoiselevel,
    rhsnoisescale, rhsnoisetype, rhscorrcoef, rhsnoiseparam)
```

Adds noise to the vector.

The distribution of noise vector elements can be normal, uniform, Poisson (with parameter) or normal with outlier, and have the expectation 0. The generated noise vector e satisfies $\|e\| = \delta$ or $\text{var}(e^{(i)}) = \sigma^2$, depending on wheteher the requested noise level is actual noise level δ or standard deviation σ^2.

Parameters

`y` – Initial vector.

`rhsnoisename` – Name of the noise distribution. Possible values: `"normal"`, `"uniform"`, `"Poisson"`, or `"normalwithoutlier"`.

`rhsnoiselevel` – Numerical value of the noise level.

`rhsnoisescale` – Name of scaling of the noise level. Possible values: `"absolute"` if the noise level is to be taken as absolute noise level, `"relative"` if the noise level given is relative.

`rhsnoisetype` – Name of noise level type. Possible values: `"actual"` if actual noise level is given, `"std"` if standard deviation is given.

`rhscorrcoef` – Correlation coefficient of every two adjacent noise vector elements. Must be in the interval $[-1, 1]$.

`rhsnoiseparam` – (Optional.) Parameter of noise distribution, used in Poisson distribution only. Default value 1.

Returns

`y_noisy` – Vector with noise added.

2.3 alphaind

`ind = alphaind(alpha, alphaseq)`

Finds the index of α in the geometric sequence of α's. The sequence must be a geometric progression. If the possible index is fractional, then rounds it to the nearest integer.

Parameters

`alpha` – Arbitrary number.

`alphaseq` – Geometric sequence of numbers.

Returns

`ind` – Index of `alpha` within the sequence `alphaseq` of numbers.

2.4 color

`c = color(i)`

Returns the `i`-th color in the sequence used to denote the lines in graphs.

The colors are coded as: 0 = red, 1 = green, 2 = yellow, 3 = orange, 4 = light blue, 5 = brown, 6 = lilac, 7 = light green, 8 = blue.

Parameters

`i` – Number of color.

Returns

`c` – Row vector of 3 elements representing color, each element between 0 and 1.

2.5 combnoisea

```
d = combnoisea(A, y, delta, eta, alpha)
```

Returns the value of the function $\delta + \eta\|x_\alpha\|$ used on the right hand side of parameter choice rules that take into account operator noise, where x_α is the Tikhonov approximation corresponding to the regularization parameter α.

Parameters

`A` – System matrix.

`y` – Right hand side.

`delta` – Noise level of the right hand side.

`eta` – Noise level of the matrix.

`alpha` – Regularization parameter.

Returns

`d` – Value of the function.

2.6 d*

These functions compute the values of functions that are compared against delta in delta-rules.

- `d = dD(A, y, alpha)`
$$d = \|r_\alpha\|$$
- `d = dME(A, y, alpha)`
$$d = \frac{(r_\alpha, r_{1,\alpha})}{\|r_{1,\alpha}\|}$$
- `d = dRG(A, y, alpha)`
$$d = (r_\alpha, r_{1,\alpha})^{1/2}$$
- `d = dBP(A, y, q, alpha)`
$$d = \frac{\sqrt{\alpha}\sqrt{q}\|x_\alpha - x_{\alpha/q}\|}{1-q}$$

- `d = dR1(A, y, k, alpha)`

  If $k = 0$, then

$$d = (r_\alpha, r_{1,\alpha})^{1/2}$$

  If $k = 1/2,\ 3/2,\ \ldots$, then

$$d = \alpha^{1/2} \Big\| \sum_{i=0}^{k+\frac{1}{2}} (-1)^i \binom{k+\frac{1}{2}}{i} x_{i,\alpha} \Big\|$$

  If $k = 1,\ 2,\ \ldots$, then

$$d = \alpha^{1/2} \left(\sum_{i=0}^{k} (-1)^i \binom{k}{i} x_{i,\alpha}, \sum_{i=0}^{k+1} (-1)^i \binom{k+1}{i} x_{i+1,\alpha} \right)^{1/2}$$

- `d = dR2(A, y, q, k, l, alpha)`

$$d = \kappa(\alpha) \frac{S(k,1)^{\frac{q}{q-1}}}{S(l, 2q-2)^{\frac{1}{q-1}}},$$

  where $S(a, b)$ is computed as follows.

  If $a = 0$ and b is even, then

$$S(a,b) = \|r_{b/2,\alpha}\|$$

  If $a = 0$ and b is odd, then

$$S(a,b) = (r_{b/2-1/2,\alpha}, r_{b/2+1/2,\alpha})^{1/2}$$

  If $a > 0$, then

$$S(a,b) = \alpha^{1/2} \Bigg(\sum_{i=0}^{\lceil a \rceil} (-1)^i \binom{\lceil a \rceil}{i} x_{\lfloor a+b/2 \rfloor - i,\alpha}, \\ \sum_{i=0}^{\lfloor a+1 \rfloor} (-1)^i \binom{\lfloor a+1 \rfloor}{i} x_{\lceil a+b/2 \rceil - i,\alpha} \Bigg)^{1/2}$$

In these formulas, $x_{i,\alpha}$ is the approximate solution found by applying Tikhonov method additionally i times, and $x_\alpha = x_{0,\alpha}$. Correspondingly, $r_{i,\alpha} = Ax_{i,\alpha} - y$ and $r_\alpha = r_{0,\alpha}$.

Parameters

`A` – System matrix.

`y` – Right hand side.

`q`, `k`, `l` – Parameters of the rule.

`alpha` – Regularization parameter.

Returns

`d` – Value of the function.

2.7 f*

These functions compute the values of functions that are minimized in heuristic rules.

- `f = fQ(A, y, usekappa, alpha)`

$$f = \|x_\alpha - x_{1,\alpha}\|$$

- `f = fHR(A, y, rho, usekappa, alpha)`

$$f = \alpha^{-\rho}(r_\alpha, r_{1,\alpha})^{1/2}$$

- `f = fReg(A, y, tau, usekappa, alpha)`

$$f = \|r_\alpha\| \, \|x_\alpha\|^\tau$$

- `f = fMReg(A, y, usekappa, alpha)`

$$f = \alpha^{-1/2}(r_\alpha, r_{1,\alpha})^{1/2}\|x_\alpha - x_{1,\alpha}\|$$

- `f = fBRS(A, y, usekappa, alpha)`

$$f = \frac{\|r_\alpha\|^2}{\alpha\|x_\alpha\|}$$

- `f = fGCV(A, y, usekappa, alpha)`

$$f = \frac{\|r_\alpha\|^2}{(m^{-1}\operatorname{tr}(I - AA_n^{-1}))^2}$$

- `f = fRGCV(A, y, gamma, usekappa, alpha)`

$$f = \frac{\|r_\alpha\|}{(m^{-1}\operatorname{tr}(I - AA_n^{-1}))^2}(\gamma + (1-\gamma)m^{-1}\operatorname{tr}((AA_n^{-1})^2))$$

- `f = fSRGCV(A, y, gamma, usekappa, alpha)`

$$f = \frac{\|r_\alpha\|}{(m^{-1}\operatorname{tr}(I - AA_n^{-1}))^2}(\gamma + (1-\gamma)m^{-1}\operatorname{tr}((A_n^{-1}A_n^{-1})^2))$$

- `f = fMGCV(A, y, c, usekappa, alpha)`

$$f = \frac{\|r_\alpha\|}{(m^{-1}\operatorname{tr}(I - cAA_n^{-1}))^2}$$

- `f = fGML(A, y, usekappa, alpha)`

$$f = \frac{\|r_\alpha\|^2}{(\det^+(I - AA_n^{-1}))^{1/m_1}}$$

- `f = fRes(A, y, usekappa, alpha)`

$$f = \frac{\|r_\alpha\|}{(\operatorname{tr}(B^*B))^{1/4}}$$

- `f = fHRME(A, y, usekappa, alpha)`

$$f = \alpha^{-1/2}\frac{(r_\alpha, r_{1,\alpha})}{\|r_{1,\alpha}\|}$$

Here $A_n^{-1} = (\alpha I + A^*A)^{-1}A^*$, $B = A(I - AA_n^{-1})$, m = problem size, $m_1 = \operatorname{rank}(I - AA_n^{-1})$, $\det^+$ is the product of nonzero eigenvalues.

Parameters

`A` – System matrix.

`y` – Right hand side.

`rho`, `tau`, `gamma`, `c` – Parameters of the rule.

`usekappa` – Boolean indicating wheter the function is to be multiplied with the corresponding power of $\kappa(\alpha)$.

`alpha` – Regularization parameter.

Returns

`d` – Value of the function.

2.8 formatexp

```
str = formatexp(number)
```

Returns the number formatted for displaying in the table. Numbers are rounded to 3 significant digits and stripped of other unnecessary symbols. For example: 1.23, 12.3, 123, 1.23e3, 1.23e-3.

Parameters

`number` – A real number.

Returns

`str` – String containing the formatted display of the number.

2.9 genseq

```
seq = genseq(alphamin, alphamax, stepsize)
```

Returns geometric sequence of numbers based on endpoints `alphamin` and `alphamax` of the interval and common ratio `stepsize`.

The last element of the sequence is `alphamax`, the first element is the largest number that is less than equal to `alphamin`. Each element after the first is `stepsize` times larger than the previous one.

Parameters

`alphamin` – Left endpoint of the interval.

`alphamax` – Right endpoint of the interval.

`stepsize` – Common ratio of geometric sequence.

Returns

`seq` – Geometric sequence of numbers as a row vector.

2.10 getelem

```
[a b c d] = getelem(v)
```

Auxiliary function that copies up to 4 first elements of a vector into different variables. All output values after the first are optional.

Parameters

`v` – Vector of numbers.

Returns

`a`, `b`, `c`, `d` – First, second, third, fourth elements of vector.

2.11 kappa

```
k = kappa(alpha, A)
```

Returns the value of the function $\kappa(\alpha) = 1 + \alpha \,/\, \|A\|^2$.

Parameters

`alpha` – Regularization parameter.

`A` – System matrix.

Returns

`k` – Value of the function.

2.12 largest*eq

```
[alphasol fvalues1 fvalues2] =
    largesteq(_func1, data1, _func2, data2, alphaseq)

[alphasol fvalues1 fvalues2] =
    largestdeceq(_func1, data1, _func2, data2, alphaseq)
```

Finds the last α_i in the sequence of α's at which

- $f_1(\alpha_i) \leq f_2(\alpha_i)$ (`largesteq`)
- $f_1(\alpha_i) \leq f_2(\alpha_i)$ and $f_1(\alpha_{i+1}) > f_2(\alpha_{i+1})$ (`largestdeceq`)

If none exists, then returns the smallest or the largest α from the sequence, depending on wheter always $f_1(\alpha) > f_2(\alpha)$ or $f_1(\alpha) \leq f_2(\alpha)$. On request also returns vectors of all values of both functions.

This function is the discrete version of solving equations of the type $f_1(\alpha) = f_2(\alpha)$ (`largesteq`) under condition that f must decrease at the location of solution, as α decreases (`largestdeceq`).

Parameters

`_func1` – Handle of the first function.

`data1` – Cell array of additional parameters (without alpha) to the first function.

`_func2` – Handle of the second function.

`data2` – Cell array of additional parameters (without alpha) to the second function.

`alphaseq` – Vector of numbers, ordered from the smallest to the largest.

Returns

`alphasol` – Last argument in the sequence `alphaseq` at which the value of the first function is less than or equal to the value of the second function (`largesteq`) but at next alpha the value of the first function is larger than the value of the second function (`largestdeceq`).

`fvalues1` – (Optional.) Vector of all values of the first function (same size as the vector `alphaseq`).

`fvalues2` – (Optional.) Vector of all values of the second function (same size as the vector `alphaseq`).

2.13 largest*sol

```
[alphasol fvalues] = largestsol(_func, data, delta, alphaseq)
```

```
[alphasol fvalues] = largestdecsol(_func, data, delta, alphaseq)
```

Finds the last α_i in the sequence of α's at which

- $f(\alpha_i) \leq \delta$ (`largestsol`)
- $f(\alpha_i) \leq \delta$ and $f(\alpha_{i+1}) > \delta$ (`largestdecsol`)

If none exists, then returns the smallest or the largest alpha from the sequence, depending on wheter always $f(\alpha) > \delta$ or $f(\alpha) \leq \delta$. On request also returns the vector of all values of the function.

This function is the discrete version of solving equations of the type $f(\alpha) = \delta$ (`largestsol`) under condition that f must decrease at the location of solution, as α decreases (`largestsol`).

Parameters

`_func` – Handle of the function.

`data` – Cell array of additional parameters (without alpha) to the function.

`delta` – A number (the right side).

`alphaseq` – Vector of numbers, ordered from the smallest to the largest.

Returns

`alphasol` – Last element in the sequence `alphaseq` for which the value of the function is less than or equal to `delta` (`largestsol`) but at next alpha the value of the function is larger than `delta` (`largestdecsol`).

`fvalues` – (Optional.) Vector of all values of the function (same size as the vector `alphaseq`).

2.14 matrix*

These functions construct matrices of some sample test problems.

- `A = matrixgauss(n, sigma)`

  Matrix $A = (a_{ij})$, where $a_{ij} = \sqrt{\frac{\pi}{2\sigma}}\, e^{-\frac{\sigma(i-j)^2}{2}}$. Default $\sigma = 0.01$.

- `A = matrixhilbert(n)`

  Matrix $A = (a_{ij})$, where $a_{ij} = \frac{1}{i+j-1}$.

- `A = matrixlotkin(n)`

  Matrix $A = (a_{ij})$, where $a_{1j} = 1$ and $a_{ij} = \frac{1}{i+j-1}$ for $i > 1$.

- `A = matrixmoler(n, alpha)`

  Matrix $A = B^T B$, where $B = (b_{ij})$ is a matrix with $b_{ii} = 1$, $b_{ij} = \alpha$ for $i < j$, and $b_{ij} = 0$ for $i > j$. Default $\alpha = 1$.

- `A = matrixpascal(n)`

  Matrix $A = (a_{ij})$, where $a_{ij} = \binom{i+j-2}{i-1}$.

- `A = matrixprolate(n, w)`

  Toeplitz matrix with the first column (a_i), where $a_1 = 2w$, $a_i = \frac{\sin 2\pi w(i-1)}{\pi(i-1)}$ for $i > 1$. Default $w = 0.25$.

Parameters

`n` – Dimension of the matrix.

`sigma`, `alpha`, `w` – (Optional.) Parameters of the matrix.

Returns

`A` – A square matrix of size `n`.

2.15 maximizer

`[alphamax fmax fvalues] = maximizer(_func, data, alphaseq)`

Finds α in the given sequence of α's for which the value of the function $f(\alpha)$ is largest. Returns the maximizing α and maximal value of the function. On request also returns the vector of all values of the function.

Parameters

`_func` – Handle of the function to be maximized.

`data` – Cell array of additional parameters (without alpha) to the function.

`alphaseq` – Vector of numbers to be searched for the maximizer.

Returns

`alphamax` – Element of the sequence `alphaseq` which maximizes the value of the function.

`fmax` – Maximal value of the function.

`fvalues` – (Optional.) Vector of all values of the function (same size as the vector `alphaseq`).

2.16 minimizer

```
[alphamin fmin fvalues] = minimizer(_func, data, alphaseq)
```

Finds α in the given sequence of α's for which the value of the function $f(\alpha)$ is largest. Returns the minimizing α and minimal value of the function. On request also returns the vector of all values of the function.

Parameters

`_func` – Handle of the function to be minimized.

`data` – Cell array of additional parameters (without alpha) to the function.

`alphaseq` – Vector of numbers to be searched for the minimizer.

Returns

`alphamin` – Element of the sequence `alphaseq` which minimizes the value of the function.

`fmin` – Minimal value of the function.

`fvalues` – (Optional.) Vector of all values of the function (same size as the vector `alphaseq`).

2.17 minNb

```
m = minNb(A)
```

Returns the minimal eigenvalue of A^*A. This is the heuristic lower bound of the parameter interval proposed by Neubauer [3].

Parameters

`A` – System matrix.

Returns

`m` – Minimal eigenvalue of A^*A.

2.18 opquasierror

```
e = opquasierror(A, y, A_true, x_true, y_true, alpha)
```

Computes the value of operator version of the quasi error function

$$\|\alpha(\alpha I + A^*A)^{-1}x_*\| + \frac{1}{2\sqrt{\alpha}}\big(\|y - y_*\| - \|A - A_*\|\,\|x_*\|\big).$$

Parameters

`A` – System matrix.

`y` – Noisy right hand side.

`A_true` – Exact matrix of the problem.

`x_true` – Exact solution of the problem.

`y_true` – Exact right hand side.

`alpha` – Regularization parameter.

Returns

`e` – Value of the function.

2.19 oprule*

These functions compute the regularization parameter chosen by specific delta-rules, and all values of the associated function, when noise levels of both right hand side and operator are given.

- `[alpha fvalues] = opruleD(A, y, delta, eta, alphaseq, c)`
- `[alpha fvalues] = opruleME(A, y, delta, eta, alphaseq, c)`
- `[alpha fvalues] = opruleMEe(A, y, delta, eta, alphaseq, c)`
- `[alpha fvalues] = opruleMEa(A, y, delta, eta, alphaseq)`
- `[alpha fvalues] = opruleRG(A, y, delta, eta, alphaseq, c)`
- `[alpha fvalues] = opruleBP(A, y, delta, eta, alphaseq, c)`
- `[alpha fvalues] = opruleR1(A, y, k, delta, eta, alphaseq, c)`
- `[alpha fvalues] = opruleR2(A, y, q, k, l, delta, eta,`
  `alphaseq, c)`

All these rules find the (largest) solution of the equation $d(\alpha) = cb_0(\delta + \eta\|x_\alpha\|)$, where $d(\alpha)$ is the function used in the rules, δ and η are noise levels of right hand side and operator, b_0 is a constant depending on the rule, and c is a used-adjustable constant (usually 1).

Parameters

`A` – System matrix.

`y` – Right hand side.

`delta` – Noise level of the right hand side.

`eta` – Noise level of the operator.

`q`, `k`, `l` – Parameters of the rule.

`alphaseq` – Geometric sequence from which the regularization parameter will be chosen.

`c` – (Optional.) Adjusting constant.

Returns

`alpha` – Regularization parameter chosen by the rule.

`fvalues` – Column vector of values of the function associated with the rule.

2.20 quasierror

```
e = quasierror(A, y, x_true, y_true, alpha)
```

Computes the value of the quasi error function

$$\|\alpha(\alpha I + A^*A)^{-1}x_*\| + \frac{1}{2\sqrt{\alpha}}\|y - y_*\|.$$

Parameters

`A` – System matrix.

`y` – Noisy right hand side.

`x_true` – Exact solution of the problem.

`y_true` – Exact right hand side.

`alpha` – Regularization parameter.

Returns

e – Value of the function.

2.21 rule*

These functions compute the regularization parameter chosen by specific rules, and all values of the associated function.

Delta-rules

- `[alpha fvalues] = ruleD(A, y, delta, alphaseq, c)`
- `[alpha fvalues] = ruleME(A, y, delta, alphaseq, c)`
- `[alpha fvalues] = ruleMEe(A, y, gamma, delta, alphaseq, c)`
- `[alpha fvalues] = ruleMEa(A, y, delta, alphaseq)`
- `[alpha fvalues] = ruleRG(A, y, delta, alphaseq, c)`
- `[alpha fvalues] = ruleBP(A, y, delta, alphaseq, c)`
- `[alpha fvalues] = ruleR1(A, y, k, delta, alphaseq, c)`
- `[alpha fvalues] = ruleR2(A, y, q, k, l, delta, alphaseq, c)`

All delta-rules find the (largest) solution of the equation $d(\alpha) = cb_0\delta$, where $d(\alpha)$ is the function used in the rules, δ is noise level of the right hand side, b_0 is a constant depending on the rule, and c is a used-adjustable constant (usually 1).

Parameters

A – System matrix.

y – Right hand side.

delta – Noise level of the right hand side.

gamma, q, k, l – Parameters of the rule.

alphaseq – Geometric sequence from which the regularization parameter will be chosen.

c – (Optional.) Adjusting constant.

Returns

`alpha` – Regularization parameter chosen by the rule.

`fvalues` – Column vector of values of the function associated with the rule.

Heuristic rules

- `[alpha fvalues] = ruleQ(A, y, alphaseq, usekappa, useNb)`

-
  ```
  [alpha fvalues] = ruleHR(A, y, rho, alphaseq,
      usekappa, useNb)
  ```

-
  ```
  [alpha fvalues] = ruleReg(A, y, tau, alphaseq,
      usekappa, useNb)
  ```
- `[alpha fvalues] = ruleMReg(A, y, alphaseq, usekappa, useNb)`
- `[alpha fvalues] = ruleHyb(A, y, alphaseq)`
- `[alpha fvalues] = ruleBRS(A, y, alphaseq, usekappa, useNb)`
- `[alpha fvalues] = ruleGCV(A, y, alphaseq, usekappa, useNb)`

-
  ```
  [alpha fvalues] = ruleRGCV(A, y, gamma, alphaseq,
      usekappa, useNb)
  ```

-
  ```
  [alpha fvalues] = ruleSRGCV(A, y, gamma, alphaseq,
      usekappa, useNb)
  ```

-
  ```
  [alpha fvalues] = ruleMGCV(A, y, c, alphaseq,
      usekappa, useNb)
  ```
- `[alpha fvalues] = ruleGML(A, y, alphaseq, usekappa, useNb)`
- `[alpha fvalues] = ruleRes(A, y, alphaseq, usekappa, useNb)`
- `[alpha fvalues] = ruleHRME(A, y, alphaseq, usekappa, useNb)`

Parameters

`A` – System matrix.

`y` – Right hand side.

`rho`, `tau`, `gamma`, `c` – Parameters of the rule.

`alphaseq` – Geometric sequence from which the regularization parameter will be chosen.

`usekappa` – (Optional.) Boolean indicating wheter the function is to be multiplied with the corresponding power of $\kappa(\alpha)$.

`useNb` – (Optional.) Boolean indicating wheter the interval of parameters is to be bound below using Neubauer's condition.

Returns

`alpha` – Regularization parameter chosen by the rule.

`fvalues` – Column vector of values of the function associated with the rule.

Automatic rules

-
  ```
  [alpha fvalues] = ruleAuto(A, y, delta0, eta0,
      delta1, eta1, alphaseq)
  ```

-
  ```
  [alpha fvalues] = ruleAutoStd(A, y, sigma0, sigmaeta0,
      sigma1, sigmaeta1, alphaseq)
  ```

Return the parameter chosen by the automatic rules that use approximate information about the noise level, depending on whether this information is given in the form of upper and lower bounds of the noise or upper and lower bounds of the standard deviation of noise.

Parameters

`A` – System matrix.

`y` – Right hand side.

`delta0`, `eta0` – Upper bounds of the right hand side and matrix noise, respectively.

`delta1`, `eta1` – Likely lower bounds of the right hand side and matrix noise, respectively.

`sigma0`, `sigmaeta0` – Upper bounds of the standard deviation of the right hand side and matrix noise, respectively.

`sigma1`, `sigmaeta1` – Likely lower bounds of the standard deviation of the right hand side and matrix noise, respectively.

`alphaseq` – Geometric sequence from which the regularization parameter will be chosen.

Returns

`alpha` – Regularization parameter chosen by the rule.

`fvalues` – Column vector of values of the function associated with the rule.

2.22 samplesolution

```
x = samplesolution(n, i)
```

Returns the vector of length `n` of `i`-th sample solution.

Sample solutions (Table 2) are coded as: 1 = constant, 2 = linear, 3 = quadratic, 4 = sinusoidal, 5 = linear+sinusoidal, 6 = step function.

Parameters

`n` – Length of the vector.

`i` – Number of sample solution.

Returns

`x` – Column vector of length `n`, containing the `i`-th sample solution.

2.23 Tbar

```
f = Tbar(A, y, alphaH, alphamin, alphamax)
```

Computes the quantity

$$\max_{\alpha_{\min}\leq\alpha\leq\alpha_{\max}} \sqrt{\alpha}\,\frac{\|x_{\alpha_{\mathrm{H}}} - x_\alpha\|}{\|B_\alpha(Ax_\alpha - y)\|}$$

Parameters

`A` – System matrix.

`y` – Right hand side.

`alphaH` – Fixed regularization parameter.

`alphamin` – Left endpoint of the minimization interval.

`alphamax` – Right endpoint of the minimization interval.

Returns

`f` – Value of the function.

2.24 Tbarfunc

```
f = Tbarfunc(A, y, alphaH, alpha)
```

Returns the value of the function

$$\sqrt{\alpha}\,\frac{\|x_{\alpha_{\mathrm{H}}} - x_\alpha\|}{\|B_\alpha(Ax_\alpha - y)\|}$$

that is maximized in the function `Tbar`.

Parameters

`alpha` – Regularization parameter.

`A` – System matrix.

`y` – Right hand side.

`alphaH` – Fixed regularization parameter.

Returns

`f` – Value of the function.

2.25 Tikherror

```
e = Tikherror(A, y, x_true, alpha)
```

Computes the error $\|x_\alpha - x_*\|$, where x_α is Tikhonov approximation corresponding to the parameter α.

Parameters

`A` – System matrix.

`y` – Right hand side.

`alpha` – Regularization parameter.

`x_true` – Exact solution of the problem.

Returns

`e` – Value of the error.

2.26 Tikhonov

`x = Tikhonov(A, y, alpha, xbar)`

Computes the Tikhonov approximation of the problem $Ax = y$, given the regularization parameter α and the initial approximation $\bar{x}$. If the initial approximation is not given, then it is set to 0.

Parameters

`A` – System matrix.

`y` – Right hand side.

`alpha` – Regularization parameter.

`xbar` – (Optional.) Column vector containing the initial approximation.

Returns

`x` – Column vector containing the Tikhonov approximation corresponding to `alpha`.

3 Contents of standalone interface

To use the 4 interfaces of the tool (page 2) locally, the user needs to set the values of parameters in the file `progparam*.m` and then run this file, where `*` is the number of the interface. The final command in the parameter file should be a call to the program `solve*.m`, which performs the computations according to the values of parameters.

The commands in the interface files are simple assignments. The variables and possible values are described below.

3.1 General

- `session` – Session number. This number is appended to the names of generated files (graphs and files of solution) to distinguish them from files generated in other sessions.

3.2 The problem

These parameters determine the problem to be solved.

- `filename` – Name of input file, containing either matrix `A` and solution vector `x` (interfaces 1 and 3) or matrix `A` and right hand side vector `y` (interfaces 2 and 4).
- `matrixname` – (Interfaces 1, 3) Name of the matrix of the problem. Used when `filename` is missing. Possible values: `'baart'`, `'deriv2'`, `'foxgood'`, `'gravity'`, `'heat'`, `'i_laplace'`, `'phillips'`, `'shaw'`, `'spikes'`, `'wing'`, `'gauss'`, `'hilbert'`, `'lotkin'`, `'moler'`, `'pascal'`, `'prolate'`.
- `matrixparam` – (Interfaces 1, 3) A number containing the parameter of matrix. Used only when `matrixname` is `'heat'`, `'spikes'`, `'gauss'`, `'moler'`, or `'prolate'`.
- `solutionname` – (Interfaces 1, 3) Name of the solution of the problem. Used when `filename` is missing. Possible values: `'baart'`, `'deriv2'`, `'foxgood'`, `'gravity'`, `'heat'`, `'i_laplace'`, `'phillips'`, `'shaw'`, `'spikes'`, `'wing'`, `'constant'`, `'linear'`, `'quadratic'`, `'sinusoidal'`, `'linearsinusoidal'`, `'stepfunction'`.
- `matrixparam` – (Interfaces 1, 3) Number or row vector containing the parameter of the solution. Used only when `solutionname` is `'deriv2'`, `'i_laplace'`, `'spikes'`, `'wing'` (row vector for the last).
- `problemsize` – (Interfaces 1, 3) Number of rows in matrix and solution. All problems have square matrices.
- `addsmoothness` – (Interfaces 1, 3) Amount p of smoothness increase: the solution x is replaced by $(A^*A)^{p/2}x$.
- `normalizeproblem` – `true` or `false`, depending on wheter the matrix A and right hand side y are scaled in such a way that $\|A\| = \|y\| = 1$.

3.3 Noise level

These parameters apply only to interfaces 1 and 3. They are used to generete the noise that is added to the right hand side and matrix.

- `rhsnoisename` – Name noise distribution for the right hand side. Possible values: `'normal'`, `'uniform'`, `'Poisson'`, `'normalwoutlier'`.
- `rhsnoiseparam` – Parameter of noise distribution. Used only when right hand side noise distribution is `'Poisson'`.
- `rhsnoiselevel` – Right hand side noise level.
- `rhsnoisescale` – Scale of the right hand side noise level. Possible values: `'relative'`, `'absolute'`.
- `rhsnoisetype` – Type of the right hand side noise. Possible values: `'actual'`, `'std'`.
- `rhscorrcoef` – Correlation coefficient of any two adjacent elements of noise vector. Must lie between -1 and 1.
- `matrixnoisename` – Name of noise distribution for the matrix. Possible values: `'normal'`, `'uniform'`.
- `matrixnoiselevel` – Matrix noise level.
- `matrixnoisescale` – Scale of the matrix noise level. Possible values: `'relative'`, `'absolute'`.
- `matrixnoisetype` – Type of the matrix noise. Possible values: `'actual'`, `'std'`.
- `numberofruns` – Number of runs (noise generations). The problem will be solved `numberofruns` times.
- `randomseed` – Random seed for noise generation.

3.4 Information about noise

These parameters describe what we assume to be known about the noise level.

Interfaces 1 and 2

- `noiseinfotype` – Noise information type. Possible values: `'upperbound'` (upper bounds of matrix and right hand side noise are known), `'std'` (standard deviations of the noise are known), `'notknown'` (nothing is known).
- `upbnoiseinforhs` – Upper bound of the right hand side noise.
- `upbnoiseinfomatrix` – Upper bound of the matrix noise.
- `upbnoiseinfoscale` – Scale used in the upper bounds. Possible values: `'relative'`, `'absolute'`.
- `stdnoiseinforhs` – Standard deviation of the right hand side noise.
- `stdnoiseinfomatrix` – Standard deviation of the matrix noise.
- `stdnoiseinfoscale` – Scale used in the standard deviations. Possible values: `'relative'`, `'absolute'`.

Interfaces 3 and 4

- `noiseinfotype` – Noise information type. Possible values: `'upperbound'` (upper bounds of matrix and right hand side noise are known), `'std'` (standard deviations of the noise are known).
- `actualrhsupper` – Upper bound of the right hand side noise.
- `actualmatrixupper` – Upper bound of the matrix noise.
- `actualrhslower` – Likely lower bound of the right hand side noise.
- `actualmatrixlower` – Likely lower bound of the matrix noise.
- `actualscale` – Scale used in the bounds. Possible values: `'relative'`, `'absolute'`.
- `stdrhsupper` – Upper bound of the standard deviation of right hand side noise.
- `stdmatrixupper` – Upper bound of the standard deviation of matrix noise.
- `stdrhslower` – Likely lower bound of the standard deviation of right hand side noise.

- `stdmatrixlower` – Likely lower bound of the standard deviation of matrix noise.
- `stdscale` – Scale used in the bounds of standard deviation. Possible values: `'relative'`, `'absolute'`.

3.5 Parameter choice rules

Rules and their accompanying data. Used only in interfaces 1, 2.

- `rule(*).name` – Name of parameter choice rule. Possible values: `'D'`, `'ME'`, `'MEe'`, `'MEa'`, `'RG'`, `'BP'`, `'R1'`, `'R2(2,2,1/2)'`, `'R2(2,1,1/2)'`, `'Q'`, `'HR'`, `'Reg'`, `'MReg'`, `'Hyb'`, `'BRS'`, `'GCV'`, `'RGCV'`, `'SRGCV'`, `'MGCV'`, `'GML'`, `'Res'`, `'HRME'`, `'HR(1/4)'`, `'Apri'`.
- `rule(*).param` – Number or row vector of values of constants in rules.

Rule names and additional information must be numbered in a sequence, for example

```
rule(1).name = 'D';
rule(1).param = 1;
rule(2).name = 'Q';
rule(2).param = [0 1];
    ...
```

3.6 Technical

Information about the set of numbers, where regularization parameters are searched from. The set has the form $\{\alpha_0 q^i : i = 0, \ldots\}$, $q < 1$.

- `alphamin` – Left endpoint of regularization parameter interval.
- `alphamax` – Right endpoint of regularization parameter interval.
- `stepsize` – Common ratio of the geometric progression of numbers in the set. Must be less than 1.

References

[1] C. Brezinski, G. Rodriguez, and S. Seatzu. Error estimates for linear systems with applications to regularization. *Numerical Algorithms*, 49(1–4):85–104, 2008.

[2] P. C. Hansen. Regularization tools version 4.1 (for Matlab version 7.3). `http://www2.imm.dtu.dk/~pch/Regutools/`, 2014-05-23.

[3] A. Neubauer. The convergence of a new heuristic parameter selection criterion for general regularization methods. *Inverse Problems*, 24:(055005), 2008.