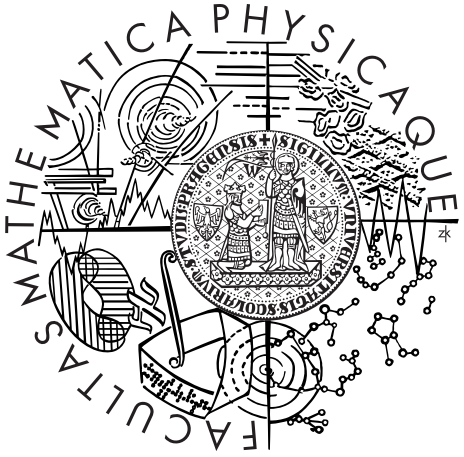
Charles University in Prague
Faculty of Mathematics and Physics

DIPLOMA THESIS



Viktor Kovtun

Regularization Based on Krylov Subspace Iterations

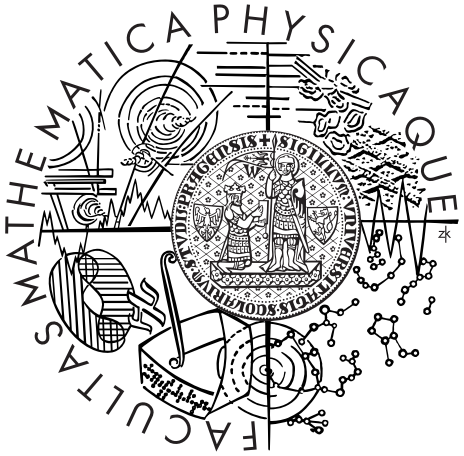
Department of Numerical Mathematics

Supervisor: RNDr. Iveta Hnětynková, Ph.D.
Study programme: Numerical and Computational Mathematics

Prague 2013

Karlova Univerzita v Praze
Matematicko-Fyzikální Fakulta

DIPLOMOVA PRÁCE



Viktor Kovtun

Regularizace založená na metodách Krylovových podprostorů

Katedra numerické matematiky

Vedoucí: RNDr. Iveta Hnětynková, Ph.D.
Studijní program: numerická a výpočtová matematika

Praha 2013

First of all, I wish to express my gratitude to my supervisor RNDr. Iveta Hnětynková, PhD. for her inspiration, guidance and patience during the entire writing process. I would also like to thank my mother for her willing to help we with the entire process of printing and binding of this theses. Special thanks to my father and to my friend Yuriy Payda, who agreed to help me to deliver the thesis to the department on time.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague Viktor Kovtun

Title: Regularization based on Krylov subspace iterations

Author: Viktor Kovtun

Department: Department of Numerical Mathematics

Supervisor: RNDr. Iveta Hnětynková, PhD.

Abstract:

In this work, we consider linear ill-posed problems $Ax \approx b$, where b is polluted by noise. These problems are difficult to solve, a direct (for example, the least squares) solution is usually useless. It is known, that combining the iterative Golub-Kahan bidiagonalization procedure with some inner regularization of the obtained bidiagonal problem forms a set of powerful regularization methods – hybrid methods. In this work, we study these methods from computational point of view. To do this we developed a MATLAB software including a user-friendly interface, and used it to run hybrid methods on various testing problems from the Regularization Toolbox [33] with different noise levels and noise colors. Noise revealing iteration detection methods are used to determine in which iteration the bidiagonalization problem becomes significantly polluted by noise. Several inner regularization and inner parameter finding methods are considered.

Keywords: ill-posed problems, regularization, iterative methods, noise in data, noise revealing

Název práce: Regularizace založená na metodách Krylovových podprostorů

Autor: Viktor Kovtun

Katedra: Katedra numerické matematiky

Vedoucí diplomové práce: RNDr. Iveta Hnětynková, PhD.

Abstrakt:

V této práci se zabýváme lineárními ill-posed problémy $Ax \approx b$, kde vektor b je zatížený šumem. Tyto problémy je komplikované řešit, neboť přímé řešení (například ve smyslu nejmenších čtverců) je zpravidla nepoužitelné. Je známo, že kombinací Golub-Kahanovy iterační bidiagonalizace s vnitřní regularizací získaného bidiagonálního problému obdržíme efektivní regularizační metody – hybridní metody. V předložené práci studujeme tyto metody z výpočetního hlediska. Proto jsme vyvinuli MATLABovskou aplikaci s uživatelsky přívětivým rozhraním a použili jsme ji k testování hybridních metod na různých testovacích problémech z Regularizačního Toolboxu [33] s různými hladinami a barvami šumu. K identifikaci iterace bidiagonalizace, ve které začíná být bidiagonální problém výrazně zatížený šumem, využíváme tzv. metody detekce noise revealing iterace. Dále uvažujeme několik metod vnitřní regularizace a metod pro nalezení vnitřního regularizačního parametru.

Klíčová slova: ill-posed úlohy, regularizace, iterativní metody, šum v datech, vyjevování šumu

Contents

Notations List

Numbers and Sets

$\mathbb{N}$	the set of natural numbers
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$	the set of nonnegative real numbers

Vectors

x	column vector
$x^T \equiv (x_1, \ldots, x_n)$	transposed vector
e_i	i-th column of the identity matrix
$(\cdot, \cdot)$	Euclidian scalar product of two vectors
x_{comp}	computed solution vector of a problem
x_{exact}	exact solution vector of a problem
x_λ	regularized solution with reg. parameter λ
b_{exact}	exact data in the right-hand side
b_{noise}	noise in the right-hand side

Matrices

A	matrix
$A \equiv [a_1, \ldots, a_m]$	representation of a matrix by columns
A^T	transposed matrix
A^{-1}	inverted matrix
A^+	pseudoinverse of a matrix
$A^\#$	regularized inverse of a matrix
I	identity matrix
$\mathrm{rank}(A)$	rank of a matrix
$\mathrm{trace}(A)$	trace of a matrix

Eigenvalues, Singular Values and Norms

λ_i, μ_i	eigenvalues of a matrix
σ_i	singular values of a matrix
$\|x\|$	norm of a vector
$\|x\|_2 \equiv \sqrt{\sum x_i^2}$	Euclidean norm of a vector
$\|A\|$	norm of a matrix
$\mathrm{cond}(A) \equiv \max(\sigma_i)/\min(\sigma_i)$	condition number of a matrix

Probability and Statistics

X	random variable (or random vector)

X_k	random process
$E(X)$	expected value of a random variable (vector)
σ_X	variance of a random variable (vector)
$R_X(s,t)$	autocovariation function of a random process
$r_X(s,t)$	autocorrelation function of a random process
$S_x(f)$	power spectral density of a process

Other Symbols

$a \propto b$	a is proportional to b
$\text{dist}(u,v) \equiv \|v-u\|_2$	Euclidean distance between 2 points u and v
$\mathcal{K}_k(A,b) \equiv \text{span}\{b, Ab, \ldots, A^{k-1}b\}$	Krylov subspace
$\delta_{noise} = \|b_{noise}\|/\|b_{exact}\|$	noise level
k_{noise}	noise revealing iteration of GKB
k_{stag}	the next iteration after k_{noise} in GKB
$k_{accepted}$	the iteration number where GKB is stopped
$g \equiv k_{accepted} - k_{noise}$	number or extra GKB iterations
$\eta \equiv \|x_{comp} - x_{exact}\|/\|x_{exact}\|$	relative approximation error

Acronyms

CG	conjugate gradient method
CGLS	CGLS method
CGNE	CGNE method
GCV	generalized cross-validation
GKB	Golub-Kahan iterative bidiagonalization
GSVD	generalized SVD
LS	least squares
LSQR	LSQR method
PSD	power spectral density
SVD	singular value decomposition
TSVD	truncated SVD

Introduction

In many fields of applications there is a need to solve linear approximation problems

$$Ax \approx b, \qquad A \in \mathbb{R}^{n\times m}, \quad x \in \mathbb{R}^m, \quad b \in \mathbb{R}^n,$$

where A represents a (discretized) mathematical model of the problem, b stands for the observed data and x is the original data to be found. These problems can sometimes be ill-posed (e.g. inverse problems – problems of converting observed measurements into information about a physical object or system) meaning that small changes to the right-hand side b can dramatically change the solution vector x. In real-world problems the right-hand side b contains except of the exact initial data b_{exact} also noise (errors) b_{noise} that can have different character. It comes from measurements errors of different kind (limited measurement precision, lens out of focus in an optical system etc.), errors in the model, discretization errors, computers finite arithmetics etc. Formally, we may assume

$$b = b_{exact} + b_{noise}, \qquad b,\ b_{exact},\ b_{noise} \in \mathbb{R}^n,$$

where only b is known. The difficulty lies in the fact that ill-posed problems cannot be solved directly, for example, using a pseudoinverse [4, §1.2.5], [22, §5.5.3]. Such a naive solution

$$x = A^+ b_{exact} + A^+ b_{noise}$$

would be meaningless due to the fact that in ill-posed problems typically $\|A^+ b_{noise}\| \gg \|A^+ b_{exact}\|$. That can be seen by substituting the singular value decomposition (SVD) of A to the last equation giving

$$x = \sum_{j=1}^{r} \frac{u_j^T b_{exact}}{\sigma_j} v_j + \sum_{j=1}^{r} \frac{u_j^T b_{noise}}{\sigma_j} v_j, \quad u_j, \in \mathbb{R}^n, v_j \in \mathbb{R}^m, \ \ \sigma_j \in \mathbb{R},$$

where $r = \mathrm{rank}(A)$, u_j and v_j are the left and the right singular vectors and σ_j are the singular values of A. In many cases, the vector b_{exact} satisfies the discrete Picard condition ($|u_j^T b_{exact}|$ decreases faster than σ_j), but the vector b_{noise} does not. For large indexes j, the value $u_j^T b_{noise}$ in the second sum above is divided by a small singular value σ_j. Therefore the norm of the second term (that equals to $\|A^+ b_{noise}\|$) can be much larger than the norm of the first one (equal to $\|A^+ b_{exact}\|$).

As we can see, it is necessary to solve ill-posed problems in a different way – by so called regularization methods [4, 31]. These methods can be divided into three categories:

- *Direct* methods usually use some decomposition of the initial problem, like QR factorization or SVD. The computational effort of these algorithms can be estimated a priori. Some widely-used methods from this category are the Tikhonov regularization [14, 24, 28, 58, 65, 67], [31, Section 5.1], TSVD [37, 68], [31, Section 3.2] etc.

- *Iterative* methods produce a sequence of vectors using only matrix-vector multiplications. These methods are preferable especially when decompositions of the problem should be avoided due to its computational cost. Iterated Tikhonov [31, Section 6.2], CGLS [6, 38], LSQR [4, 57], Kaczmarz's method [41] and others belong to this category.

- *Hybrid* methods [28], [31, Section 6.6] are a combination of the previous ones.

All regularization methods require determination of *regularization parameters*, which control the smoothness of the solution and how close the solution should fit the original data. Among the most popular methods there are GCV [19, 71], L-curve [29, 36], discrepancy principle [52] and others.

This thesis focuses on hybrid methods using the Golub-Kahan iterative bidiagonalization (GKB) as the outer regularization, and TSVD or Tikhonov as the inner regularization method. Projection onto the Krylov subspace is known to be a form of regularization with the regularization parameter k being the dimension of the projection subspace. If k is too small, there is not enough information to recreate a good approximation to the solution (over-regularization). On the other hand, if k is too large, noise of high amplitude usually gets into the approximation (under-regularization). The iteration k_{noise}, where noise starts to significantly contaminate the projection in GKB can be determined automatically using *noise revealing iteration detection methods* as described in [40, 69]. In hybrid methods, the idea is to select k slightly larger than k_{noise} and apply a (usually direct) regularization to the projected problem.

The goal of this thesis is to study the behavior of regularization based on GKB depending on the number of iterations k, the choice of different inner regularization and inner regularization parameter finding methods and other factors, for variety of the initial problems with different noise levels and noise colors. We concentrate especially on hybrid regularization and its specifics. For that purpose an application in MATLAB language has been developed. The included user interface (UI) allows the user to pick a problem, its dimension, noise level, noise color, and other parameters, and solve it using the selected hybrid regularization method. The problem and settings of the regularization method can be specified using different UI components on the left part of the application form (list-boxes, edit-boxes and scroll-bars). The result is presented in the form of graphs on the right part of the form as well as in the form of a PDF file. Also a faster application without UI has been developed to iterate over a set of parameters of interest for various problems and produce a set of results in the PDF format.

By default, white noise in considered in this thesis, however, there are sections where special colored noise is discussed. The precise arithmetic is used unless it is specified otherwise, and the computations are made in the set of real numbers $\mathbb{R}$. By default, Eucledian norms are used, although in many places this fact is emphasized by an index ($\|\cdot\|_2$). The organization of this thesis is as the following.

Chapter 1 is an introductory chapter to ill-posed problems. It briefly describes such problems and their properties. As a sample ill-posed problem the Fredholm integral equation of the first kind is considered. Noise and its colors are also reviewed.

Chapter 2 contains motivation for regularization methods. A naive solution for two examples of ill-posed problems is studied. Direct, iterative and hybrid regularization methods are summarized, focus is put on the regularization methods which are used in the computational part of this work: the GKB method, and TSVD and Tikhonov regularization.

Chapter 3 summarizes the method which allows to determine the GKB iteration, where noise starts to severely contaminate the projected problem. The Lanczos tridiagonalization algorithm is described as a starting point to further mathematical inferences. Three techniques for automatic detection of the noise revealing iteration are reviewed in this chapter, namely the stagnation-based, the closest-to-origin and the minimum-point criteria.

Chapter 4 reviews well known regularization parameter choice methods (discrepancy principle, GCV, L-curve), which are also used in numerical experiments in the last chapter to find an inner regularization parameter of the selected hybrid method.

Chapter 5 represents the computational part of this work. Our UI is described in details. Numerical experiments to study the behavior of hybrid methods for different ill-posed problems are presented. Effects of different aspects are surveyed: the choice of the number of GKB iterations (i.e. the outer regularization parameter), the noise revealing iteration detection method, the inner regularization method, the inner regularization parameter finding method etc.

Chapter 1

Ill-Posed Problems

In linear algebra the problem of finding the solution(s) of a linear system

$$Ax = b, \qquad A \in \mathbb{R}^{n\times m}, \qquad x \in \mathbb{R}^m, \qquad b \in \mathbb{R}^n \tag{1.1}$$

is considered to be the basis and fundamental part of this field of mathematics. In theory, we can always determine, when the system has one solution, when there are infinite number of them or when the system has no solution, by computing and comparing rank(A) and rank($[A|b]$). The solutions themselves can also be found using many well-known solution algorithms. However, when it comes to practical applications difficulties may appear. One of them is that the problem (1.1) can be ill-posed and the initial data can be contaminated by noise.

1.1 Introduction to Ill-Posed Problems

Consider the following linear approximation problem:

$$Ax \approx b, \qquad A \in \mathbb{R}^{n\times m}, \qquad x \in \mathbb{R}^m, \qquad b \in \mathbb{R}^n. \tag{1.2}$$

The problem is called *well-posed* if it has a solution, and that solution depends on the initial data continuously [31, 60]. If the problem does not satisfy this condition it is called *ill-posed*.

In numerical linear algebra (because the rank of a matrix can be computed only approximately) the corresponding systems of equations are classified as *stable* or *unstable* respectively. Stability of the system significantly depends on the *condition number* [60]

$$\mathrm{cond}(A) \equiv \sqrt{\frac{|\mu_1|}{|\mu_n|}}, \tag{1.3}$$

where μ_1 and μ_n are the maximal and the minimal eigenvalues of the matrix A^TA. For symmetric matrices this formula can be simplified to

$$\mathrm{cond}(A) \equiv \frac{|\lambda_1|}{|\lambda_n|}, \tag{1.4}$$

where λ_1 and λ_n are the maximal and the minimal eigenvalues of the matrix A. If the number cond(A) is "small" the matrix A is called *well-conditioned*, if cond(A) is "large" the matrix A is called *ill-conditioned*. Ill conditioning of the problem (1.2) can be caused by incorrect mathematical model that can be improved. This

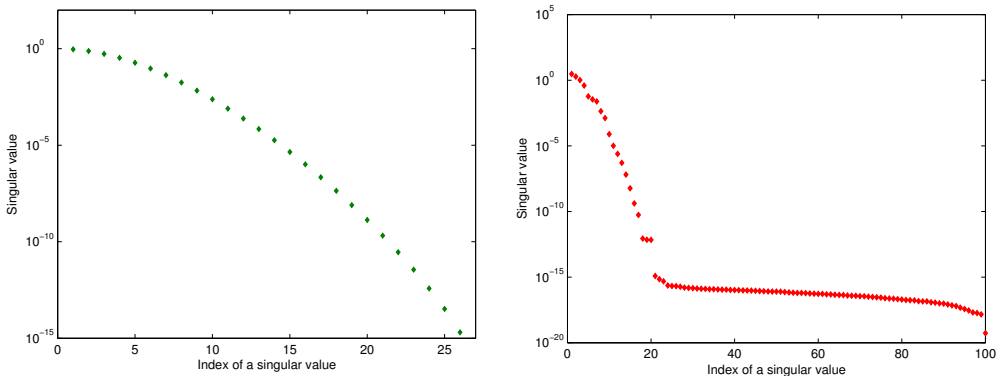


Figure 1.1: *Two examples from the Regularization Toolbox [33]: (left) discrete ill-posed problem* Parallax*, where singular values gradually decrease to zero, (right) rank-deficient problem* Shaw *with a well-determined numerical rank.*

modified system can then be solved by standard numerical techniques [4, 22]. However, this is not always possible.

Approaching these problems numerically, we should consider a *numerical* ϵ*-rank* r_ϵ of the matrix A, which can be defined by [66]

$$r_\epsilon = \min_{||A-B|| \leq \epsilon} \text{rank}(B), \tag{1.5}$$

for a given small tolerance ϵ. In other words, r_ϵ satisfies

$$\sigma_{r_\epsilon} \leq \epsilon \leq \sigma_{r_\epsilon+1} \tag{1.6}$$

When looking at the set of singular values of the matrix A, we may distinguish two classes of ill-conditioned problems [31]:

- **Discrete ill-posed problems** – numerical rank of the matrix A is not well-defined, i.e. all singular values of A decay gradually to zero. Numerical treatment of this kind of problems lies in finding a balance between the norm of the solution and the residual norm. This will be explained later.
- **Rank-deficient problems** – problems with a well-defined numerical rank r_ϵ characterized by a well-determined gap between large and small singular values of the matrix A, or as per (1.6), between r_ϵ and r_ϵ+1. This implies the existence of linearly dependent columns/rows of the matrix A and hence the presence of redundant information in the matrix. Thus, linearly dependent information needs to be extracted so that a problem with a well conditioned matrix could be solved. However, note that r_ϵ needs to be insensitive to small perturbations of ϵ and the singular values. Otherwise, the problem should be dealt by the techniques aimed at the discrete ill-posed problems.

Figure 1.1 gives examples of these two classes of ill-posed problems: a discrete ill-posed (left) and a rank-deficient problem (right) [66].

1.2 Sample Ill-Posed Problem

As a classic example of ill-posed problem let us consider the problem examined in [24, 31]. Consider the following integral equation

$$\int_0^1 K(s,t)\,f(t)\,dt = g(s), \quad 0 \le s \le 1, \tag{1.7}$$

where the functions $K(s,t)$ and $g(s)$ are a priori known, the unknown function is $f(t)$. The integral is called *Fredholm integral*, and the equation is called *Fredholm integral equation of the first kind.*

The function $K(s,t)$ is defined by the mathematical model of the problem, the function $g(s)$ is known as the result of measurements in a finite set of points $s_1, \ldots, s_n$. Thus discretization by the variable s turns the problem (1.7) into

$$\int_0^1 k_i(t)\,f(t)\,dt = b_i, \quad i = 1, \ldots, n, \tag{1.8}$$

where the functions k_i and the values b_i are known. As a result, we got an integral equation, that is continuous in one variable t. In order to solve this equation numerically it has to be discretized. Methods for integral equation discretization can be found in [2, 10, 11] and [43, Chapter 3]. In general, there are two types of discretization methods, quadrature methods and Galerkin methods.

In the *quadrature method* the integral (1.8) is approximated in the following way:

$$\int_0^1 \psi(t)\,dt \approx \sum_{j=1}^{m} w_j \psi(t_j), \tag{1.9}$$

where $0 \le t_1 < \ldots < t_m \le 1$ are discretization points with the corresponding weights $w_1, \ldots, w_m$. After applying this scheme on the problem (1.8) we obtain the discrete problem $A\tilde{f} = b$ with an $n \times m$ matrix A and a vector b of size n. Their components are defined as the following:

$$a_{ij} = w_j K(s_i, t_j), \quad b_i = g(s_i). \tag{1.10}$$

Solving the system we obtain the solution $\tilde{f} = (\tilde{f}(t_1), \ldots, \tilde{f}(t_n))^T$ that is an (discrete) approximate solution of the problem (1.8) and, in turn, an (discrete) approximation to the solution f of the initial integral equation (1.7). For more detailed information about numerical quadrature methods and principles of their construction we refer to [13, 45, 55].

In the *Galerkin method* (the name originated from the research [18]) we define two sets of basis functions ϕ_i and ψ_j. The components of the corresponding matrix A and vector b have the following representations:

$$a_{ij} = \int_0^1 K(s,t)\,\phi_i(s)\,\psi_j(t)\,ds\,dt, \quad b_i = \int_0^1 g(s)\,\phi_i(s)\,ds. \tag{1.11}$$

As a result we obtain the system of equations $Ay \approx b$. The corresponding approximate solution of the initial integral equation (1.7) is

$$f(t) \approx \tilde{f}(t) = \sum_{i=1}^{n} y_i \psi_i(t), \quad y = (y_1, \ldots, y_n)^T. \tag{1.12}$$

Broader and more detailed information about the Galerkin method and principles of its application can be found in [16, 51, 70].

In both cases, the problem (1.7) leads to a linear approximation problem that is, however, ill-posed. The difficulty lies in the fact, that the kernel K has smoothing properties, meaning that after the integration all high-frequency "effects" – jumps, cusps, shagginess – are "smoothed out". This difficulty will be explained in details in Chapter 2, see also [44, Chapter 15].

1.3 Noise

In the real world in ill-posed problems we deal with additional difficulty – noisy values. In the problem (1.2) noise can appear in the right-hand side b. It can come from measurement inaccuracies, discretization errors, rounding errors etc. However, in some cases also the mathematical model might not represent the problem accurately, and therefore the matrix A may contain inaccurate data. Here, in this section, we briefly review basic kinds (colors) of noise, that can appear in practice. Most of the information about noise is taken from [69, Chapter 3] and [59].

1.3.1 Basic Concepts

In this section we review several definitions from statistics and signal processing, which we use later.

The sequence of random values $\{X_t, t \in T\}$ is called a *random process*. In the theory of random processes t is considered to be time. This parameter, however, might not represent physical time (t can be an index of a solution vector, for example).

Let us define EX_t as a *mean value* of a random process at the time t. For processes with $E|X_t|^2 < \infty$ we can define the *autocovariation function*

$$R_X(s,t) = E[(X_s - EX_s)(\overline{X_t} - \overline{EX_y})]. \tag{1.13}$$

The *autocorrelation function* is defined as

$$r_X(s,t) = \frac{R(s,t)}{\sqrt{R_X(s,s)}\sqrt{R_X(t,t)}}, \tag{1.14}$$

where $\sigma_X^2 = R_X(t,t)$ is variance of the random process at the time t.

A random process X_t is called *uncorrelated* if

$$R_X(s,t) = \begin{cases} \sigma_{X_t}^2, & t = s, \\ 0, & t \neq s, \end{cases} \quad \text{or} \qquad r_X(s,t) = \begin{cases} 1, & t = s, \\ 0, & t \neq s. \end{cases} \tag{1.15}$$

For a random process we define the *power spectral density* (PSD):

$$S_X(f) = \sum_{k=-\infty}^{\infty} R_X(k) e^{-2\pi i f k}, \tag{1.16}$$

where f stands for a frequency. This spectral density determines the distribution of the signal by frequencies. The power of the signal can be computed by integrating the spectral density over all frequencies. The PSD is used for classification of noise by color. This way of classification is widely used in such fields as acoustics, astronomy, electrical engineering etc. In the following subsections we consider several noise colors based on the character of PSD that is described by the relation

$$S_X(f) \propto 1/f^{\alpha}, \tag{1.17}$$

where α determines the character of noise.

1.3.2 White Noise

White noise is a random process, that consists of uncorrelated random values with zero mean and finite variance σ^2 (see Figure 1.2). For this kind of process the PSD doesn't depend on frequency and can be written as the following:

$$S_X(f) = \sigma^2. \tag{1.18}$$

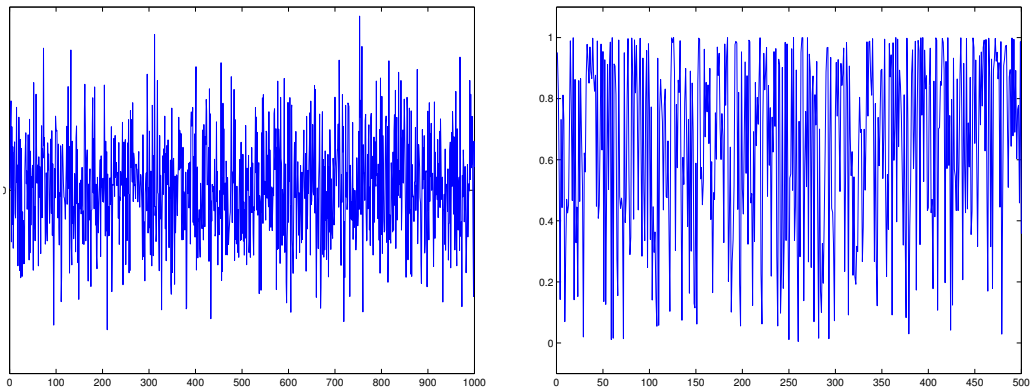


Figure 1.2: *Random white noise (left) and its Fourier coefficients (right).*

The graph of empirically computed PSD will have steady character (see Figure 1.3). This noise can be generated by the MATLAB command `randn` that creates Gaussian white noise with zero mean and unitary variance. The name white is derived from the analogy of the white noise and white light spectra.

1.3.3 Blue Noise

Noise is called *blue* if its PSD is proportional to the frequency:

$$S_X(f) \propto f. \tag{1.19}$$

In Figure 1.4 there is an illustration of a blue noise realization along with its Fourier coefficients. In Figure 1.5 we can see, that in average the PSD grows as the frequency grows.

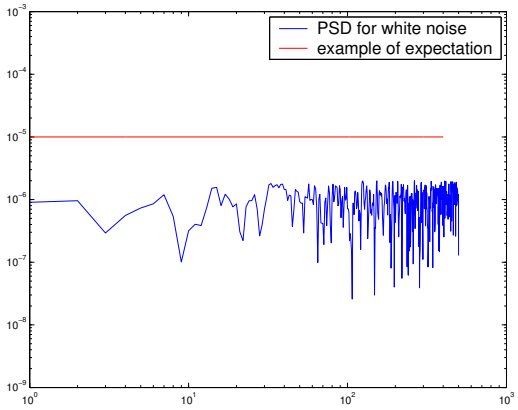


Figure 1.3: *The power spectral density of white noise and an example of the expectation of the PSD.*

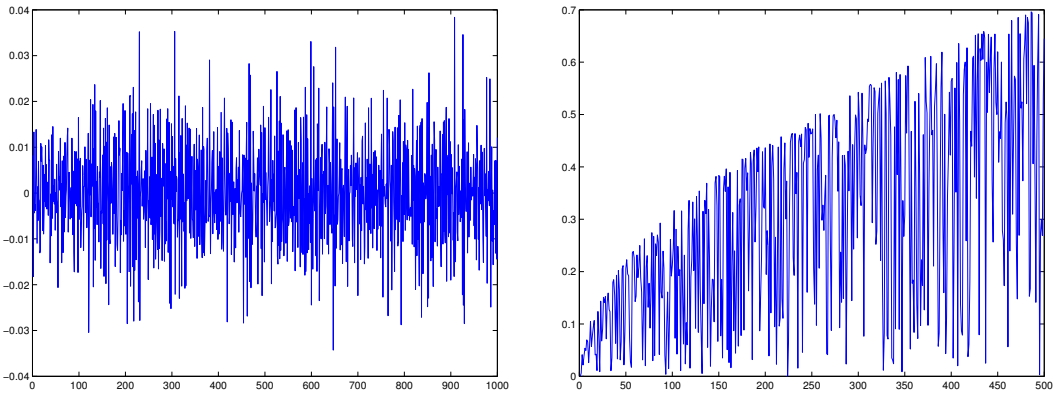


Figure 1.4: *Blue noise (left) and its Fourier coefficients (right).*

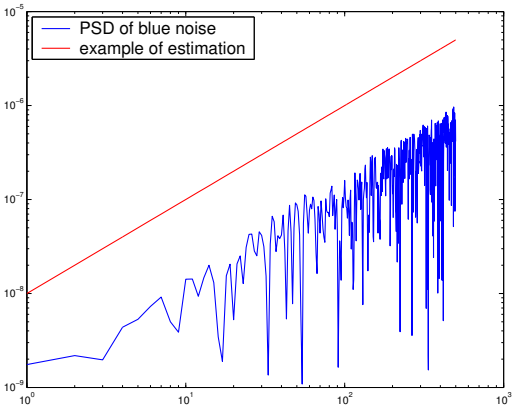


Figure 1.5: *The power spectral density of blue noise and an example of the increasing rate of the PSD.*

1.3.4 Violet Noise

For *violet noise* the PSD is proportional to the square of the frequency.

$$S_X(f) \propto f^2. \tag{1.20}$$

An example of realization of violet noise and it's Fourier coefficients are depicted in Figure 1.6. These graphs show, that this process has larger Fourier coefficients for the higher frequencies. The PSD of this random process has growing character as frequency grows (see Figure 1.7).

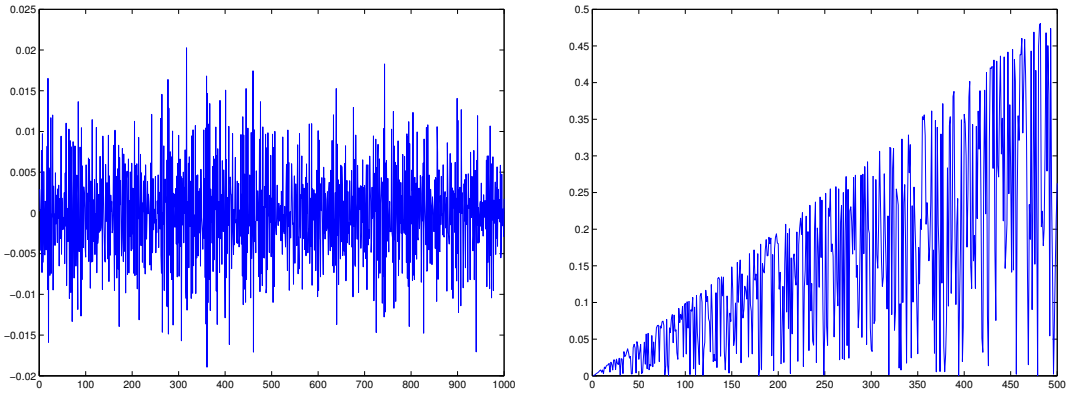


Figure 1.6: *Violet noise (left) and its Fourier coefficients (right).*

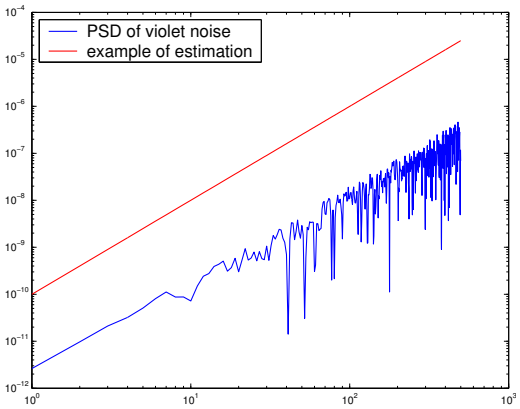


Figure 1.7: *The power spectral density of violet noise and an example of the increasing rate of the PSD.*

1.3.5 Pink Noise

A random process with PSD inversely proportional to the frequency is called *pink noise*:

$$S_X(f) \propto \frac{1}{f}. \tag{1.21}$$

In Figure 1.8 we can see a realization example of such a process and its Fourier coefficients. The PSD of this process is depicted in Figure 1.9. As we can see in the figures, as frequency grows the corresponding Fourier coefficients become smaller. Similarly, in average the PSD decreases as the frequency grows.

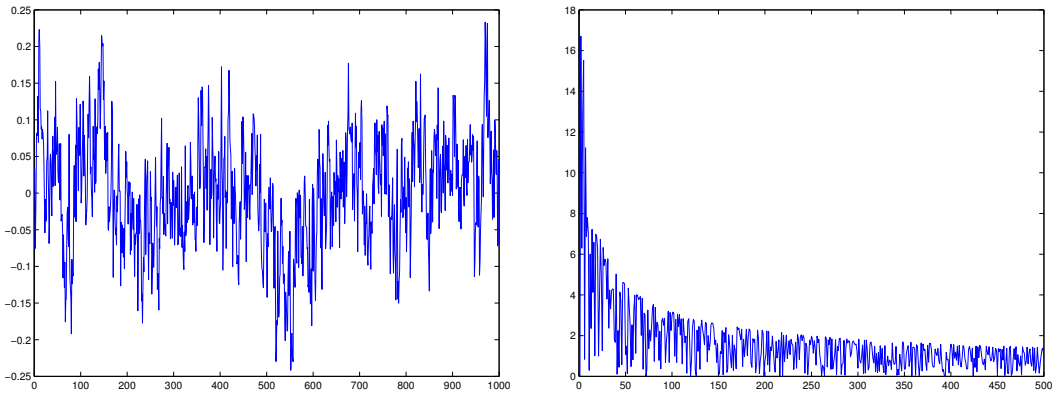


Figure 1.8: *Pink noise (left) and its Fourier coefficients (right).*

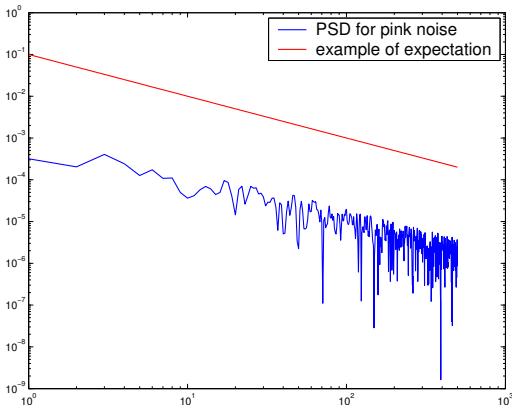


Figure 1.9: *The power spectral density of pink noise and an example of the decreasing rate of the PSD.*

1.3.6 Brownian Noise

The name of this noise type is derived from the fact that Brownian motion produces this kind of noise, see Figure 1.10. This noise has other alternative names as *Brown noise*, *red noise* or *random walk noise*. The PSD for this type of noise can be described as the following:

$$S_X(f) \propto \frac{1}{f^2}. \tag{1.22}$$

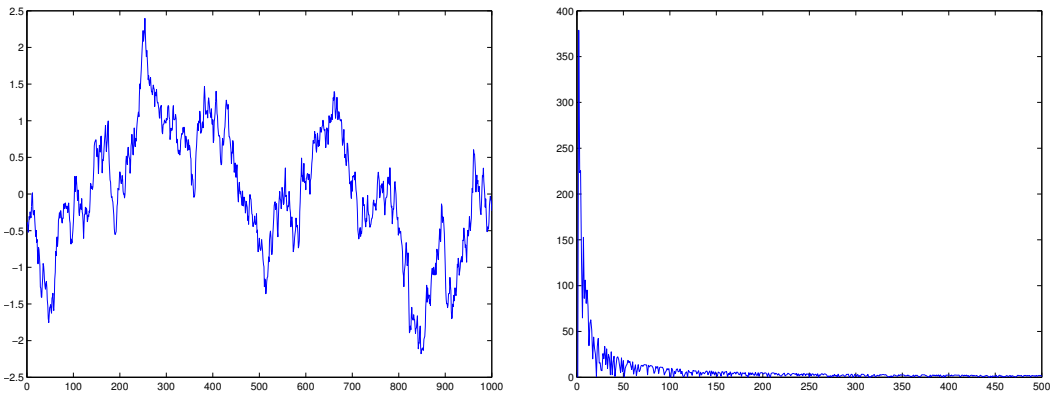


Figure 1.10: *Brownian noise (left) and its Fourier coefficients (right).*

For Brownian noise as frequency grows the PSD decreases, see Figure 1.11. Similarly Fourier coefficients are larger in the smaller frequencies domain.

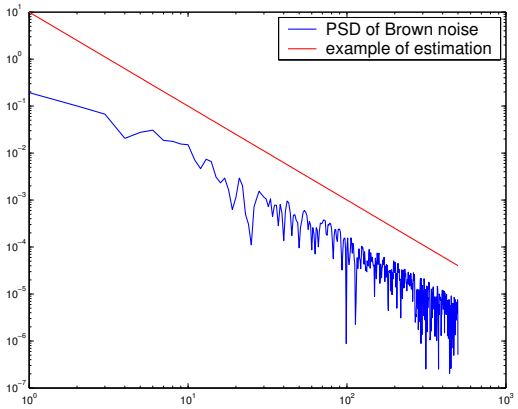


Figure 1.11: *The power spectral density of Brownian noise and an example of the decreasing rate of the PSD.*

Chapter 2

Regularization Methods

Let us show, that ill-posed problems contaminated by noise cannot be solved directly. One of the techniques how to deal with such problems is *regularization* – a powerful tool for solving ill-posed problems. The name is derived from the fact that regularization methods enforce regularity of the computed solution making sure that the solution is "smooth enough". This helps to suppress some noise components, leading usually to more stable approximate solutions (see [30, Chapter 4]).

2.1 Naive Solution

Consider an example of an ill-posed problem taken from [35]: a digital optical system of grayscale imaging with the size $k \times l$ pixels (see Figure 2.1). Each pixel of an image can be represented as a number in some range, e.g. from 0 for black to 255 for white. Vectorizing the $k \times l$ matrix of pixels (by placing the columns of the matrix below each other) we get a right-hand side vector $b \in \mathbb{R}^{kl}$ of the system (1.2).

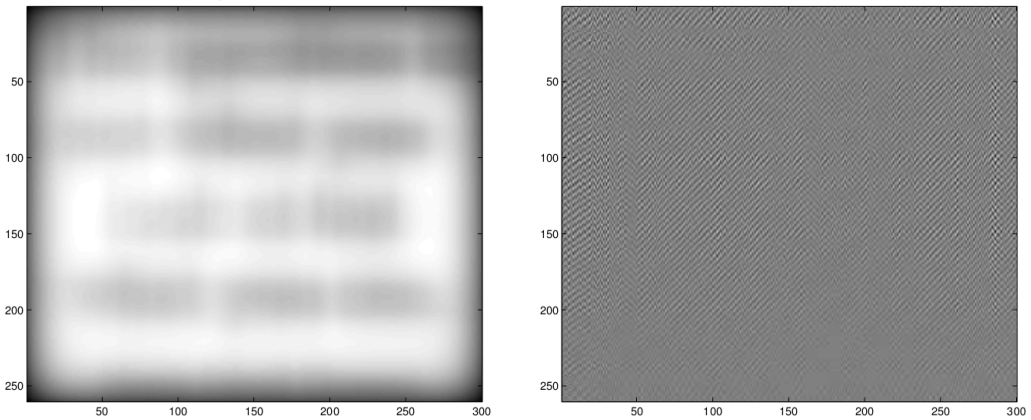


Figure 2.1: *Blurred grayscale image (left) and an attempt to obtain a (naive) solution of the blurring problem using direct algebraic methods (right).*

An optical system can be represented by an operator A: $\mathbb{R}^{kl} \to \mathbb{R}^{kl}$ that acting upon a real image x produces an image b. If A is linear this action is formulated by linear system (1.2) with $n = m = kl$. If the optical system is, for example, out of focus, the matrix A has blurring properties (sharp edges of the image are blurred) and produces a blurred image. As we will see below an attempt to solve the system (1.2) directly using standard algebraic tools does not provide a satisfactory result (see Figure 2.1 right part).

Now we describe this problem mathematically (considering general dimensions n, m in (1.2)). In equation (1.2) the matrix $A \in \mathbb{R}^{n\times m}$ with $\text{rank}(A) = r$ can be presented in an SVD form:

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_j u_j v_j^T, \tag{2.1}$$

where $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}, U^{-1} = U^T, V^{-1} = V^T$ are matrices, the columns of which are the left and right singular vectors of the matrix A, and Σ is a diagonal matrix with the corresponding singular values on the main diagonal:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0. \tag{2.2}$$

Using the psedoinverse the LS solution of the system (1.2) can be written in the form:

$$x^{-1} = A^+ b = V\Sigma^{-1}U^T b = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i. \tag{2.3}$$

Consider $b = b_{exact} + b_{noise}$, where b_{exact} represents the exact data and b_{noise} represents *white* noise in b, $\|b_{exact}\| \gg \|b_{noise}\|$. If we start solving the equation

$$Ax = b_{exact} + b_{noise}, \tag{2.4}$$

we obtain a *naive* solution

$$x_{naive} = A^+(b_{exact} + b_{noise}) = \sum_{i=1}^{r} \frac{u_i^T b_{exact}}{\sigma_i} + \sum_{i=1}^{r} \frac{u_i^T b_{noise}}{\sigma_i}. \tag{2.5}$$

Here $x_{exact} = A^+ b_{exact}$ is the exact solution and $x_{noise} = A^+ b_{noise}$ is the noise component of the naive solution. Usually the vector b_{exact} satisfies the discrete Picard condition [31], i.e. as i grows $u_i^T b_{exact}$ decreases in average faster than σ_i. On the other hand, the vector b_{noise} does not usually satisfy this condition, because the components $u_i^T b_{noise}$ are random values distributed around the same mean level[1], and for white noise it has PSD that does not depend on frequency. As a result, for x_{noise} we obtain multipliers for v_i, which grow in average as i grows. Thus for very small σ_i the second sum in (2.5) starts to dominate over the first one making the naive solution useless.

Now we return to the example in Figure 2.1. Here the vector b_{exact} is dominated by the low frequency components (see Figure 2.1 left). On the other hand, b_{noise} is basically white noise, and therefore it has no dominating frequencies. Application of the pseudoinverse A^+ on the vector $b = b_{exact} + b_{noise}$ yields amplification of the high-frequency components of b, and that means that $\|A^+ b_{noise}\| \gg \|A^+ b_{exact}\|$. And this is the reason why the solution in Figure 2.1 on the right does not represent a good approximation of the original image.

[1] In the signal processing theory this set of random values is called *mean ergodic random process*.

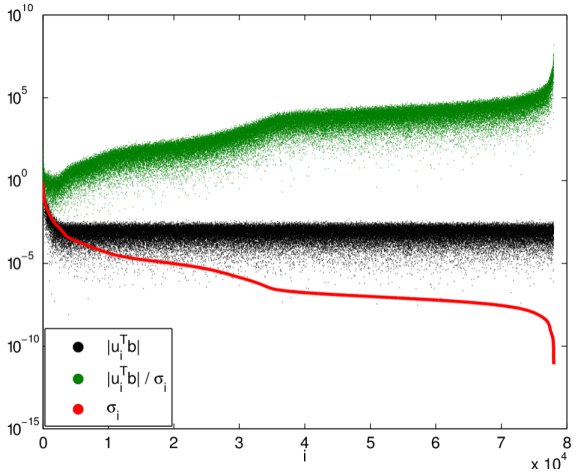


Figure 2.2: *Projections $|u_i^T b|$ (black dots), $|u_i^T b|/\sigma_i$ (green dots) and singular values σ_i of A (red line).*

Figure 2.2 shows that for this example the projections $|u_i^T b|$ (black dots) decrease with the same rate as the singular values (red line) until they reach the noise level δ_{noise}, where they start stagnating. From this point on the noisy component starts dominating in the solution, which conceals useful information completely. Green dots in the figure represent fractions $\frac{|u_i^T b|}{\sigma_i}$ to depict their growth rate.

It is known that for ill-posed problems, the singular vectors u_i and v_i in (2.1) have increasing number of sign change as i grows, i.e. as singular values σ_i decrease the frequencies of both left and right singular vectors increase see Figure 2.3. Thus the naive solution in Figure 2.1 (right) is high-frequency dominated.

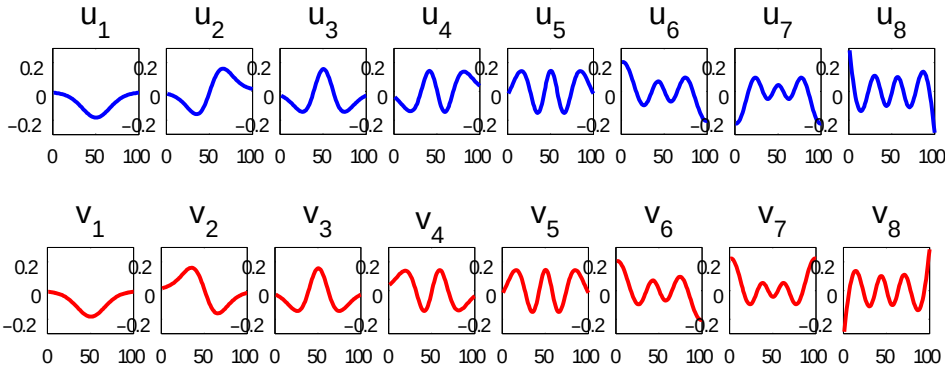


Figure 2.3: *The left u_i (top) and right v_i (bottom) singular vectors for the problem* Shaw *of the dimension 100.*

Now consider a different problem – *Shaw* from the Regularization Toolbox [33]. This problem represents a discretization of the Fredholm integral equation example described in Section 1.2. In Figure 2.4 the directly computed (naive) solution of the discrete problem is shown. Again we see, that it is dominated by amplified noise (high frequencies).

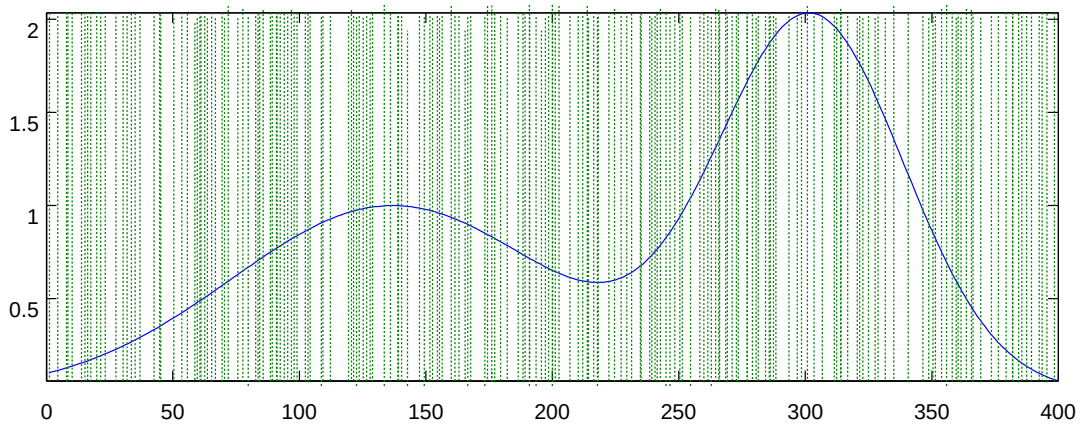


Figure 2.4: *The exact (blue solid line) and a directly computed (naive) solution (green dashed line) of the problem* Shaw *of the dimension 400 with the noise level 0.01. The variance of the naive solution is much larger than of the exact solution, therefore the green lines appear vertical.*

We see that an approximate solution of an ill-posed problem should be found in a different way. One of them is regularization – a modification of the problem in order to make the solution more stable with respect to small perturbations (noise) of the initial data. A regularized solution of the image deblurring problem from Figure 2.1 is shown in Figure 2.6, and a regularized solution of the problem *Shaw* from Figure 2.4 is depicted in Figure 2.5. Regularization methods can be divided into two categories: direct and iterative.

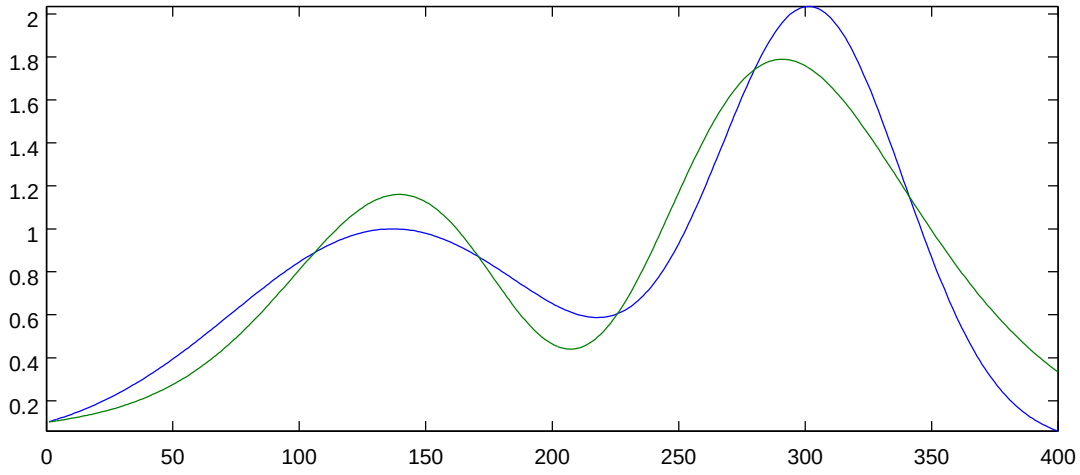


Figure 2.5: *The exact (blue line) and a regularized solution (green line) of the problem* Shaw *of the dimension 400 with the noise level 0.01.*

2.2 Direct Regularization

Direct methods use some kind of decomposition of the original matrix such as the QR factorization or SVD, they can use orthogonal transformations, matrix multiplications, bidiagonal reductions etc. Therefore computational costs for these

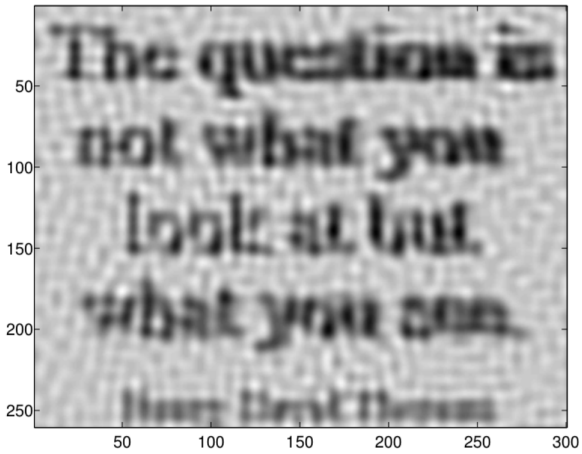


Figure 2.6: *A regularized TSVD solution of the problem in Figure 2.1 (right).*

methods can be estimated a priori. Efficient implementations of the aforementioned procedures can be found in the basic linear algebra subprograms (BLAS) and linear algebra package (LAPACK) libraries [1]. Moreover, parallel implementations of these methods exist [3, 12, 26].

2.2.1 TSVD

As it was mentioned before, extremely large errors in the naive solution are caused by components, which correspond to small singular values. Assuming that the discrete Picard condition is satisfied we try to "chop off" the SVD components dominated by noise. In this way, we get the *truncated SVD* (TSVD) solution x_k retaining only the first k components of the naive solution:

$$x_k \equiv \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i, \tag{2.6}$$

where k is the regularization parameter controlling the amount of noise in the corresponding solution. The methods for choosing regularization parameters are reviewed in Chapter 4.

Just for illustration we can view TSVD as replacing A by a regularized matrix A_k:

$$\begin{aligned} A_k &= \begin{bmatrix} | & & | \\ u_1 & \dots & u_k \\ | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} - & v_1 & - \\ & \vdots & \\ - & v_k & - \end{bmatrix} \\ &= \sum_{i=1}^{k} u_i \sigma_i v_i^T, \quad k \in \mathbb{N}. \end{aligned} \tag{2.7}$$

The condition number of this matrix is $\mathrm{cond}(A_k) = \sigma_1/\sigma_k$ that is usually much smaller than the condition number $\mathrm{cond}(A_k) = \sigma_1/\sigma_n$ of the original matrix A.

The disadvantage of the TSVD method is the requirement to compute (at least a part of) the SVD of the matrix. This can be problematic for large-scale problems. More detailed information about TSVD can be found in [30, Section 4.2] and [31, Sections 3.2, 5.3].

2.2.2 Tikhonov Regularization

The idea of the *Tikhonov regularization* method is to introduce the regularity requirement into the formulation of the problem. In other words, the modified formulation of the initial problem (1.2) is

$$\min_x \{\|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2\}, \quad \lambda \in \mathbb{R}_+, \tag{2.8}$$

or in an equivalent form

$$\min_x \left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2, \quad \lambda \in \mathbb{R}_+. \tag{2.9}$$

Here λ is the regularization parameter, that controls the weighting of terms in the sum (2.8). We comment on the components of the formula (2.8) in more detail:

- The first term $\|Ax - b\|_2^2$ defines the precision of the solution, i.e. how well the computed solution x fits the noisy data b. If this term is too big, then, obviously, x does not solve the problem. On the other hand, if this norm is too small it means that x solves the initial problem with noise almost precisely, that should be avoided as well, because we do not want to fit noise in the data (i.e. approximate the naive solution).
- The second term $\|x\|_2^2$ defines the regularity of the solution. The motivation for using this norm is that the naive solution is dominated by the high-frequency components with large amplitudes. Thereby, the idea is to suppress the high-frequency components by controlling the norm $\|x\|_2^2$.
- The balance between the terms is controlled by the factor λ^2. The larger λ is the regularity of the solution becomes more important (however, $x \to 0$ as $\lambda \to \infty$). On the other hand, the smaller λ is the more important becomes how exactly the solution x_λ solves the original system (1.2) (if $\lambda = 0$ we obtain the original system and the naive solution).

The goal is to find some optimal λ for the solution x_λ to be both regular enough and at the same time to solve the original system (1.2) with a good precision. The ways of choosing of this regularization parameter are reviewed in Chapter 4.

2.3 Iterative Regularization

The main idea of iterative methods is reducing the problem of large dimension $n \times m$ to a problem of much smaller dimension. This is achieved by projecting A on specifically constructed (often Krylov) subspaces. Such methods avoid SVD

(or GSVD) computations, creating the subspaces and the projections iteratively, using only the *matrix-vector product* Av. Iterative methods generate a sequence of vectors $x^{(k)}$, that usually converges to a solution of the initial problem (1.2). Thus these methods are preferable especially in the case, when the matrix A is too large and direct decomposition can be too time-consuming or memory-demanding.

Consider iterative methods where every vector $x^{(k)}$ is a regularized solution on the step k, where k plays the role of a regularization parameter. We are interested in iteration schemes, which initially approximate those SVD components $(u_i^T b/\sigma_i)v_i$, that correspond to the largest singular values. In the early iterations the sequence $x^{(k)}$ approaches a regularized solution but (as SVD components are approximated) in the last iterations it approaches the least squares (naive) solution $x_{LS} = A^+ b$, see Section 2.1. This phenomenon is called *semiconvergence* [54, p. 89]. Summary of many such iterative methods can be found in [31, Chapter 6]. Those are: *Landweber iteration*, *iterated Tikhonov regularization*, *ν-method*, *Kaczmar's method*, *regularizing CG iterations* etc. We concentrate on iterative methods based on the *Golub-Kahan iterative bidiagonalization* (GKB) [20].

Given a matrix A and a vector b this algorithm generates a matrix L_k (or L_{k+}) with non-zero elements only on the main diagonal and on the diagonal line directly below it. The algorithm can be written in the following form [64].

input A, b
$w_0 := 0$
$\beta_1 := \|b\|$
$s_1 := b/\beta_1$
for $k = 1, \ldots, \min(m, n)$
    $p := A^T s_k - \beta_k w_{k-1}$
    $\alpha_k := \|p\|$
    if $\alpha_k = 0$
        return
    $w_k = p/\alpha_k$
    $q := A w_k - \alpha_k s_k$
    $\beta_{k+1} := \|q\|$
    if $\beta_{k+1} = 0$
        return
    $s_{k+1} := q/\beta_{k+1}$
end

The cycle iterates until $\alpha_j = 0$ or $\beta_{j+1} = 0$ or until the dimension of the problem is reached. Let $S_k \equiv [s_1, \ldots, s_k] \in \mathbb{R}^{n \times k}$, $W_k \equiv [w_1, \ldots w_k] \in \mathbb{R}^{m \times k}$ be matrices with orthonormal columns and

$$L_k \equiv \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix}, L_{k+} \equiv \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}. \tag{2.10}$$

The first k steps of the bidiagonalization can be written in the matrix form

$$A^* S_k = W_k L_k^T, \tag{2.11}$$

$$AW_k = S_{k+1} L_{k+} \tag{2.12}$$

(see [5, 7, 20, 40]).

Consider we seek an approximation to the solution of (1.2) in the Krylov subspace $\mathcal{K}_k(A^T A, A^T b)$, i.e. we search y_k such that $AW_k y_k \approx b$, where $x_k = W_k y_k$. If we wish to minimize the norm of the residual $r_k = b - AW_k y_k$, then y_k has to satisfy

$$\|L_{k+} y_k - e_1 \beta_1\| = \min_y \|L_{k+} y - e_1 \beta_1\|. \tag{2.13}$$

In other words, y_k is the least squares solution of the projected problem

$$L_{k+} y_k \approx \beta_1 e_1, \tag{2.14}$$

see [4, 22]. This corresponds to the LSQR [57, 56] or CGLS [38] methods that are mathematically equivalent.

If the approximation of the solution is based on ensuring the orthogonality of the residual r_k to the Krylov subspace generated by the columns of S_k, then we have

$$L_k y_k = \beta_1 e_1, \tag{2.15}$$

which corresponds to the CGNE method [27]. Regularization properties of LSQR, CGLS and CGNE methods have been widely studied, see [34, 40].

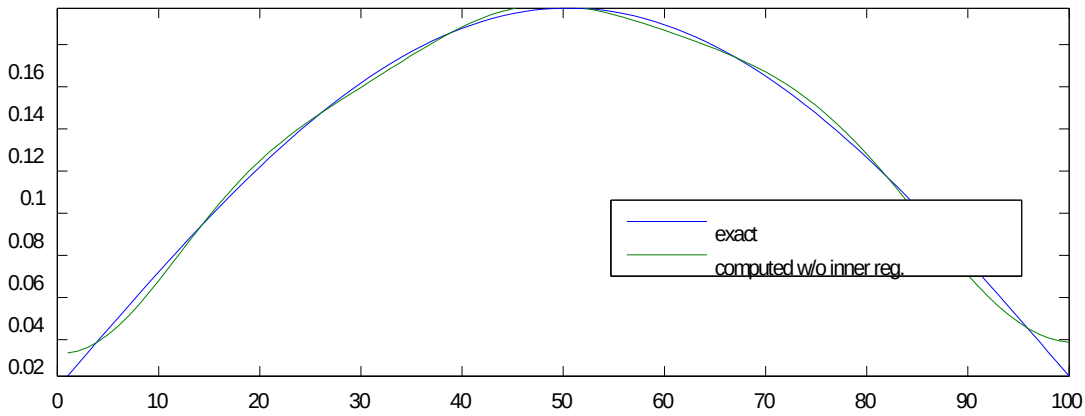


Figure 2.7: *Solution of the* Baart *problem with the noise level* 10^{-10} *for the most optimal regularization parameter* $k = 8$ *in GKB – exactly before noise contaminates the solution.*

It is well known, that the projection onto the Krylov subspace with dimension k represents by itself a form of regularization where k stands for the regularization parameter. For small k there is not enough information in the projected

problem to reconstruct the solution (over-regularization). If k gets larger, noise may transfer to the projected problem (under-regularization). Thus it is often reasonable to compute more iterations (use larger k) of GKB and then regularize the projected problem (2.14) or (2.15) using some inner (usually direct) regularization (see Section 2.4). Methods combining outer and inner regularizations are called *hybrid methods*, see [5, 7, 17, 27, 42, 69].

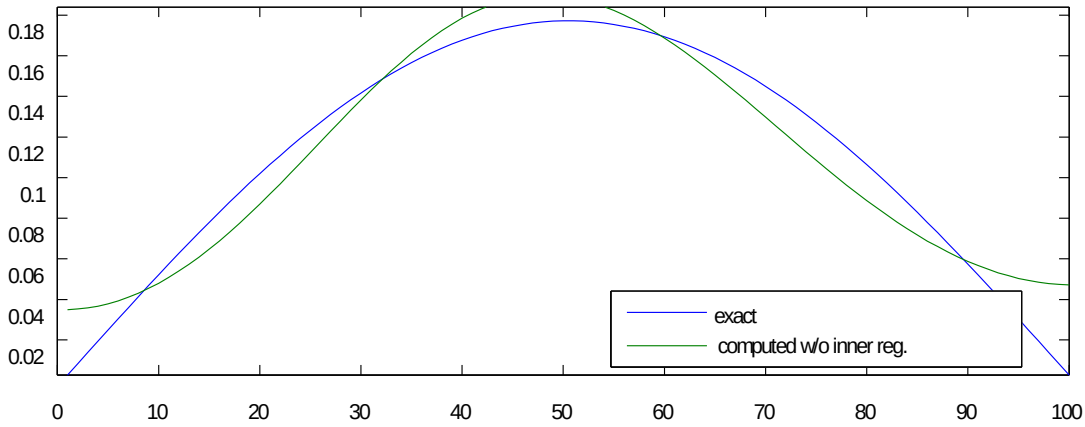


Figure 2.8: *Solution of the* Baart *problem with the noise level* 10^{-10} *for too small regularization parameter* k *(*5 *instead of* 8*) in GKB. The solution is over-regularized.*

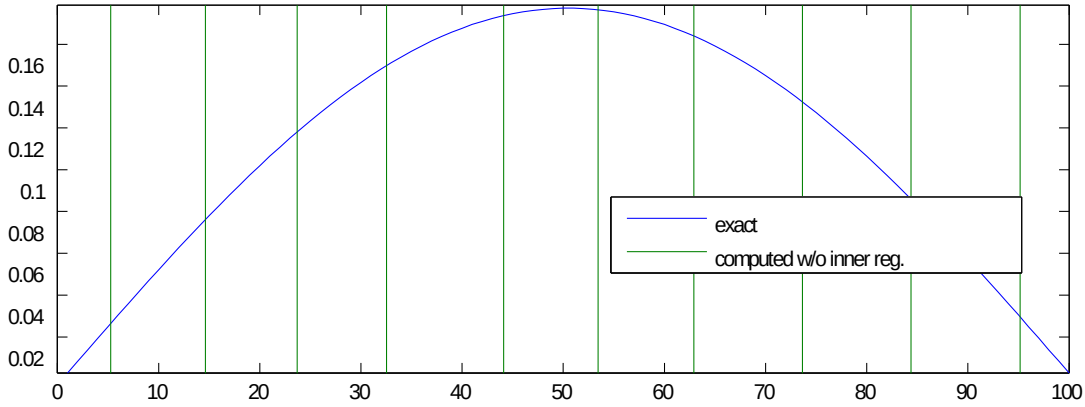


Figure 2.9: *Solution of the* Baart *problem with the noise level* 10^{-10} *for too large regularization parameter* k *(*12 *instead of* 8*) in GKB. The solution is underregularized.*

2.4 Hybrid Regularization

In this section we follow theoretical interpretation from [31]. More detailed information about hybrid methods and where this notion came from can also be found in that source.

Consider solving an ill-posed problem by GKB using (2.15) with and without inner regularization. Figures 2.7 – 2.9 show for the problem *Baart* of the dimension 400 from the Regularization Toolbox [33], with the noise level 10^{-10} how the solution without inner regularization differs from the exact one depending on the selected regularization parameter k, that is the dimension of the matrix L_k. Figure 2.7 represents the most desirable situation, when the iterative bidiagonalization process is stopped for $k = 8$ exactly before it started using components significantly contaminated by noise. Figures 2.8 and 2.9 depict the situations, when the process was stopped too early or too late. In the first case the solution is called *overregularized*. We see that is it oversmoothed. In the second case the solution is called *underregularized* and it is completely contaminated by noise. In Figure 2.10 and 2.11 we can see how inner regularization can improve the solution in case we take one or two extra iterations in the GKB, i.e. $k = 9$, $k = 10$ respectively.

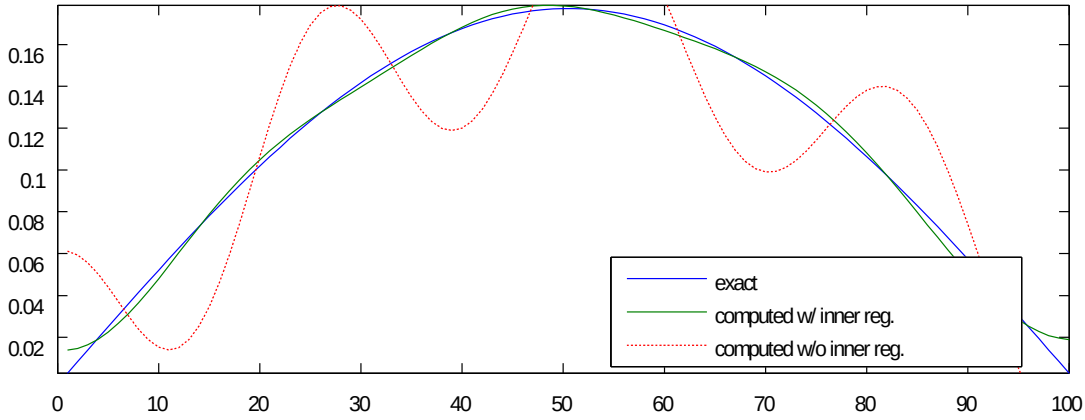


Figure 2.10: *Solution of the* Baart *problem with the noise level* 10^{-10} *computed by GKB with 1 extra iteration* ($k = 9$). *Notice how the solution with inner regularization by TSVD (solid green line) and the solution without inner regularization (dotted red line) differ and how close the first one is to the exact one.*

These numerical experiments illustrate that usage of inner regularization has one more effect. When the iteration number k (the iteration when the bidiagonalization algorithm should be stopped) is "slightly too high" (that means that not only large singular values were caught, but some smaller ones as well), the vector $x^{(k)}$, ideally, does not deteriorate. This means, that for a hybrid method the choice of correct stopping point k is not so critical as opposed to CGLS or LSQR (see Section 2.3), for example.

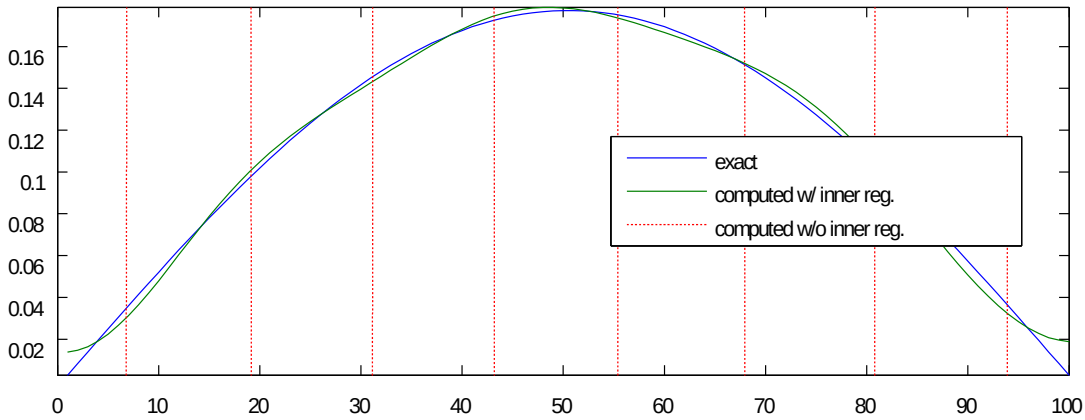


Figure 2.11: *Solution of the* Baart *problem with the noise level* 10^{-10} *computed by GKB with 2 extra iterations* ($k = 10$). *Notice that the amplitude of the solution without inner regularization is too large to fit into the area of the graph.*

Chapter 3

Noise Level Revealing Methods

In the GKB (see Section 2.3) detecting the iteration where noise begins to significantly propagate into the projected problem is advantageous for the next step of the solution improvement – inner regularization (see Section 2.4). This iteration can be used to determine the appropriate dimension of the subsequent problem for the inner regularization, and influences how much information from the original problem is used. If we stop the bidiagonalization process too early the solution is overregularized, because not enough information is used. On the other hand, as it is discussed also in this work, for hybrid methods adding several extra iterations does not degrade the solution. In most cases the opposite happens – the solution gets improved, since we have more information in the projected problem from the original problem. The noise that is necessarily present in the projection is damped by the subsequent inner regularization. In this chapter we follow the logical inferences from the papers [40, 69]. We summarize based on [40] how it is possible to identify the GKB iteration where *noise begins to contaminate the projection.*

3.1 Lanczos Tridiagonalization

The GKB (see Section 2.3) is closely related to the Lanczos tridiagonalization algorithm [46, 47]. Based on the symmetric matrix AA^T and the starting vector

$$s_1 = b/\beta_1,\ \beta_1 = \|b\| \tag{3.1}$$

this algorithm yields in k steps the symmetric tridiagonal matrix T_k such that

$$AA^T S_k = S_k T_k + \alpha_k \beta_{k+1} s_{k+1} e_k^T \tag{3.2}$$

and

$$T_k = L_k L_k^T = \begin{bmatrix} \alpha_1^2 & \alpha_1\beta_2 & & \\ \alpha_1\beta_2 & \alpha_2^2+\beta_2^2 & \ddots & \\ & \ddots & \ddots & \alpha_{k-1}\beta_k \\ & & \alpha_{k-1}\beta_k & \alpha_k^2+\beta_k^2 \end{bmatrix}, \tag{3.3}$$

where L_k is the matrix of the GKB. This relationship is described in more detail in [39] and the references given there.

The algorithm itself can be written as the following:

input A, b
$s_0 := 0$
$\beta_1 := \|b\|$
$s_1 := b/\beta_1$
for $k = 1, \ldots, m$
$v := As_k - \beta_k s_{k-1}$
$\alpha_k := \omega_k^T v$
$v := v - \alpha_k \omega_k$
$\beta_{k+1} := \|v\|$
if $\beta_{k+1} = 0$
return
$\omega_{k+1} := v/\beta_{k+1}$
end

Consider the SVD of the bidiagonal matrix of the projected problem

$$L_k = P_k \Theta_k Q_k^T, \tag{3.4}$$

where $P_k^{-1} = P_k^T$, $Q_k^{-1} = Q_k^T$, and Θ_k is a diagonal matrix with singular values of L_k on the diagonal ordered as

$$\theta_1^{(k)} < \theta_2^{(k)} < \ldots < \theta_k^{(k)}. \tag{3.5}$$

From the Lanczos tridiagonalization we get the spectral decomposition of T_k:

$$T_k \equiv L_k L_k^T = P_k \Theta_k^2 P_k^T, \tag{3.6}$$

where $(\theta_l^{(k)})^2$ are its eigenvalues and $p_l^{(k)} \equiv P_k e_l$ its eigenvectors, $l = 1, \ldots, k$. Furthermore, consider the SVD of A:

$$A = U \Sigma V^T = \sum_{j=1}^{n} \sigma_j u_j v_j^T, \tag{3.7}$$

where $U^{-1} = U^T$, $V^{-1} = V^T$, and Σ is a diagonal matrix with

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > 0 \tag{3.8}$$

on the diagonal. Then

$$AA^T = U \Sigma^2 U^T \tag{3.9}$$

is the spectral decomposition of the matrix AA^T, σ_j^2 are its nonzero eigenvalues an u_j the corresponding eigenvectors, $j = 1, \ldots, n$.

3.2 Noise Level Determination

It has been shown in [38, 40, 47, 49, 63] that the Lanczos tridiagonalization (3.2) in fact generates in the iteration k a distribution function $\omega^{(k)}$ with nodes $(\theta_i^{(k)})^2$ and weights $|(p_i^{(k)}, e_1)|$, $i = 1, \ldots k$, which approximates the distribution function ω with the nodes $\sigma_n^2, \ldots, \sigma_1^2$ and the corresponding weights $|(\frac{b}{\|b\|}, u_n)|^2, \ldots,$

$|(\frac{b}{\|b\|}, u_1)|^2$. It has been analyzed in [40] that for smaller nodes of the distribution function ω the weights are defined almost exclusively by noise, see Figure 3.1 for sizes of (b, u_j), $j = 1, \ldots, 400$ for the problem *Shaw* and different noise levels. In other words, an index j_{noise} exists, that for all $j \geq j_{noise}$ the following formula holds:

$$\left|\left(\frac{b}{\|b\|}, u_j\right)\right|^2 \approx \left|\left(\frac{b_{noise}}{\|b\|}, u_j\right)\right|^2. \tag{3.10}$$

Moreover, since b is low frequency dominated usually $j_{noise} \ll n$, see [40]. Now define the combined weight for nodes with $j \geq j_{noise}$:

$$\delta^2 \equiv \sum_{j=j_{noise}}^{n} \left|\left(\frac{b_{noise}}{\|b\|}, u_j\right)\right|^2. \tag{3.11}$$

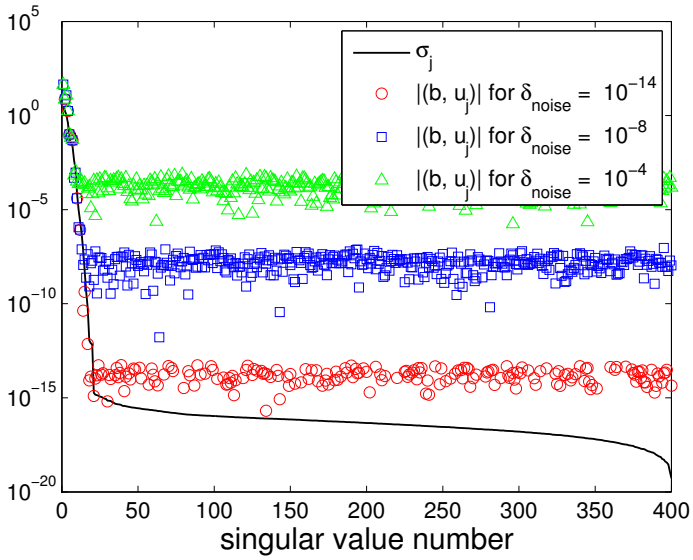


Figure 3.1: *Problem* Shaw *of the dimension 400 with the noise levels:* $10^{-14}, 10^{-8}, 10^{-4}$. *Singular values and the sizes of projections* (b, u_j), $j = 1, \ldots, 400$.

In Figure 3.2 (right) one can see that the weights for the iterations $j \geq j_{noise}$ diminish very slowly. Taken into account that usually $\|b_{exact}\| \gg \|b_{noise}\|$ we can approximate:

$$\delta^2_{noise} = \frac{\|b_{noise}\|^2}{\|b_{exact}\|^2} \approx \frac{\|b_{noise}\|^2}{\|b\|^2} = \sum_{j=1}^{n} \left|\left(\frac{b_{noise}}{\|b\|}, u_j\right)\right|^2. \tag{3.12}$$

For problems with white noise the following formula holds

$$\delta^2 \approx \frac{n - j_{noise}}{n} \delta^2_{noise} \approx \delta^2_{noise}, \tag{3.13}$$

see [40] for detailed explanation. In Figure 3.2 (left) the dependence of δ on j_{noise} is shown.

The ill-posed problems usually satisfy the discrete Picard condition [32], which states that the weights $|(\frac{b}{\|b\|}, u_j)|^2$ in average decrease faster than the corresponding singular values σ_i^2 (see [64]). Because the weights $|(\frac{b}{\|b\|}, u_j)|^2$ for small indexes

j correspond to the larger singular values $\sigma_1^2, \sigma_2^2, \ldots$, in early iterations (i.e. for small k) the singular values of L_k (Ritz values) $(\theta_i^{(k)})^2, (\theta_{i-1}^{(k)})^2, \ldots$ (large i) approximate $\sigma_1^2, \sigma_2^2, \ldots$ (see [69, Section 1.4] and [38, 49, 63]).

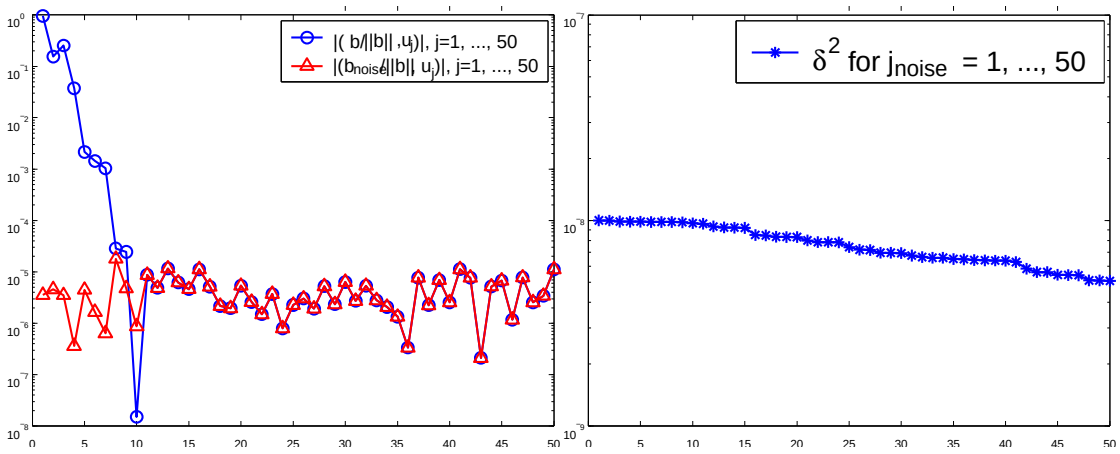


Figure 3.2: *The values* $|(\frac{b}{\|b\|}, u_j)|^2$ *and* $|(\frac{b_{noise}}{\|b\|}, u_j)|^2$ $j = 1, \ldots, 50$ *(left) and the values* δ^2 *for* $j_{noise} = 1, \ldots, 50$ *(right) for the problem* Shaw *of the dimension 400 with the noise level* $\delta_{noise} = 10^{-4}$.

As k grows the smallest Ritz value $(\theta_1^{(k)})^2$ ultimately approximates or becomes less than $\sigma_{j_{noise}}^2$ and since δ^2 satisfies (3.11), the weight corresponding to $(\theta_1^{(k)})^2$ becomes

$$\left|\left(p_1^{(k)}, e_1\right)\right|^2 \approx \delta^2 \approx \delta_{noise}^2. \tag{3.14}$$

The weight δ^2 starting from $k = j_{noise} + 1$ starts to almost stagnate (its values

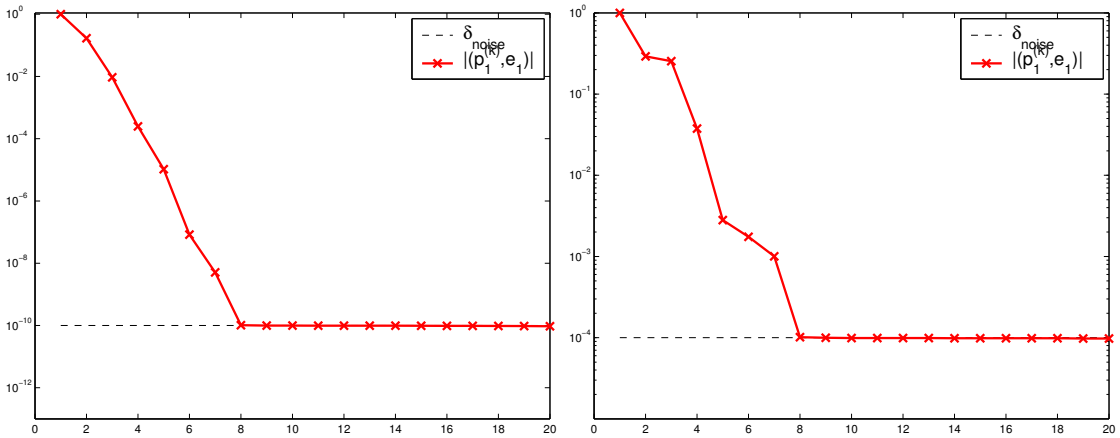


Figure 3.3: *The absolute values of the first components of left singular vectors of* L_k *–* $|(p_1^{(k)}, e_1)|$ *– which correspond to the smallest singular value for the problems* Baart *of the dimension 400 with the noise level* 10^{-10} *(left), and* Shaw *of the dimension 400 with the noise level* 10^{-4} *(right). Here* $k_{noise} = 7$ *in both examples.*

remain almost the same), and therefore the corresponding weights $|(p_1^{(k)}, e_1)|^2$ also start to almost stagnate. The iteration where it happens is denoted as k_{stag}, see

Figure 3.3.

As a conclusion we can state, that by (3.13) for the noise level estimation it is sufficient to monitor the norm of the first component $|(p_1^{(k)}, e_1)|$ of the left singular vector of the bidiagonal matrix L_k. The level of noise is estimated by the value where that norm starts to stagnate.

$$\delta_{noise} \approx \left| \left(p_1^{(k_{stag})}, e_1 \right) \right|. \tag{3.15}$$

The last iteration before this happens is called the *noise revealing iteration* ($k_{noise} = k_{stag} - 1$). It was also shown in [40], that k_{noise} is the iteration, where *noise begins to significantly propagate to the projected problem* in GKB. This iteration can be detected automatically, as studied in [69]. Noise level determination and noise revealing iteration detection can be used for building efficient stopping procedures.

3.3 Noise Level Revealing Techniques

In this section we review the techniques for k_{noise} and k_{stag} determination, which are presented in [40, 69].

3.3.1 Stagnation-Based Criterion

This criterion [40, 69] determines the iteration k_{stag} as the one, where the sequence $|(p_1^{(k)}, e_1)|$ starts to stagnate with small perturbations. This point is detected using the following formula:

$$\frac{|(p_1^{(k+1)}, e_1)|}{|(p_1^{(k+1+step)}, e_1)|} < \left(\frac{|(p_1^{(k)}, e_1)|}{|(p_1^{(k+1)}, e_1)|} \right)^{\gamma}, \tag{3.16}$$

with the parameters *step* and γ that can be used to tune the accuracy of the method.

3.3.2 Closest-to-Origin Criterion

This criterion [69] determines the stagnation point as the closest point to the "virtual" origin $(0, 0)$ in the logarithmic or semi-logarithmic scale (see Figure 3.5). The point should satisfy the condition

$$\min_{k=1,\ldots,k_{max}} \operatorname{dist}\left((0,0), \left(k, |(p_1^{(k)}, e_1)| \right) \right). \tag{3.17}$$

In the work [69] the following MATLAB algorithm is suggested[1]:

[1] It is presented with a small modification.

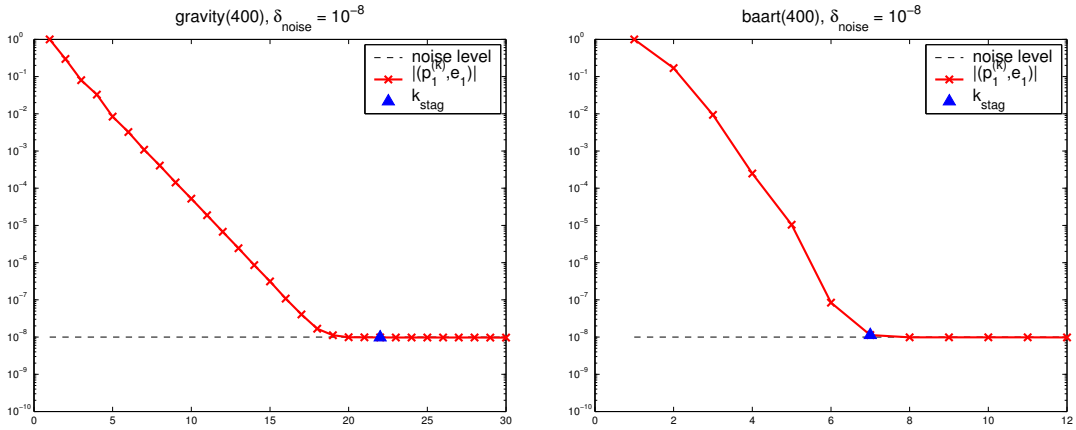


Figure 3.4: *The sequence* $|(p_1^{(k)}, e_1)|$ *and the iteration* k_{stag} *detected using the Stagnation-Based Criterion for the problems* Gravity *of the dimension 400 (left) and* Baart *of the same dimension (right) with the noise level* $\delta_{noise} = 10^{-8}$.

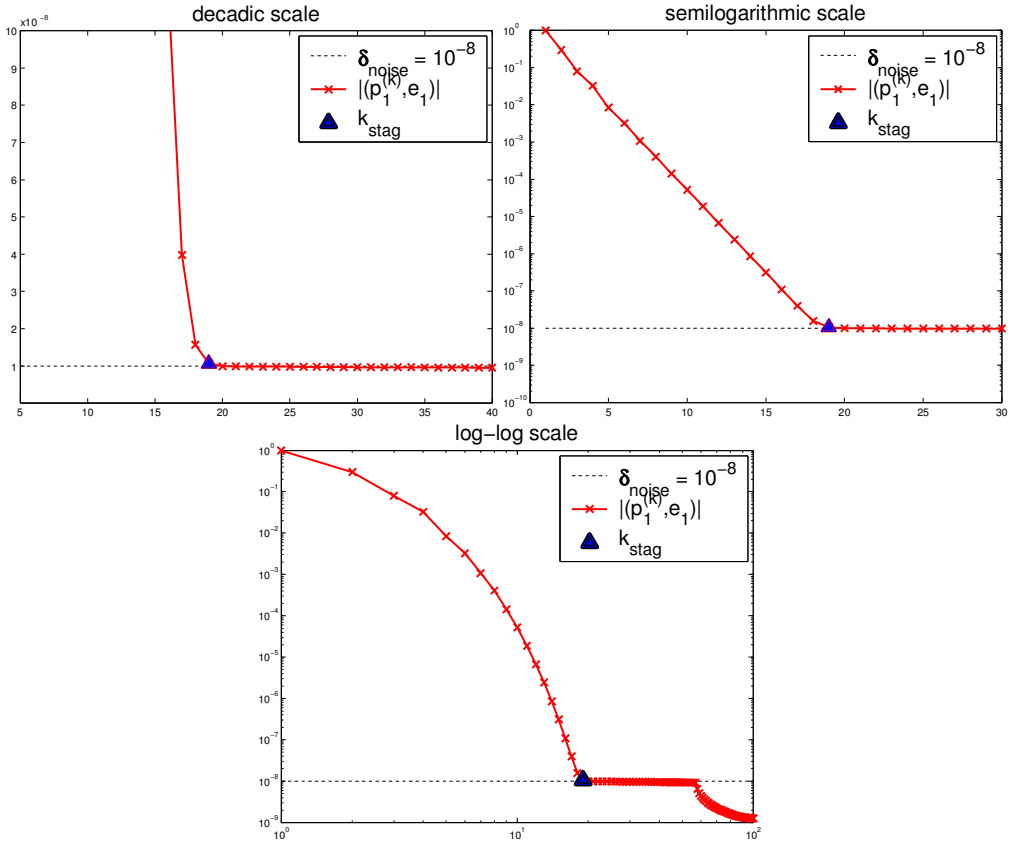


Figure 3.5: *The sequence* $|(p_1^{(k)}, e_1)|$ *in linear (left), semi-logarithmic (right) and logarithmic (bottom) scales for the problem* Gravity *of the dimension 400 with the noise level* $\delta_{noise} = 10^{-8}$.

```
% Input:
% P .......... first components of matrix P_k of left
% singular vectors
% y_center ... value close to origin, default = 0
%
% Output:
% knoise ..... corner point
%
% Variables:
% leng = max(size(P))
% minimal = 1000; default
for j = 1:leng,
  check1 = abs(P(j));
  check = [check,check1];
  lev1 = log(check1 - y_center);
  levels = [levels,lev1];
  min_lev_part = norm([levels(j),j]);
  min_lev = [min_lev,min_lev_part];
  if min_lev_part < minimal,
    minimal = min_lev;
    knoise = j;
  end;
end;
```

In this implementation there is one free parameter `y_center`, that is the coordinate of the starting point where the distances are measured from. This parameter is again used to tune the method.

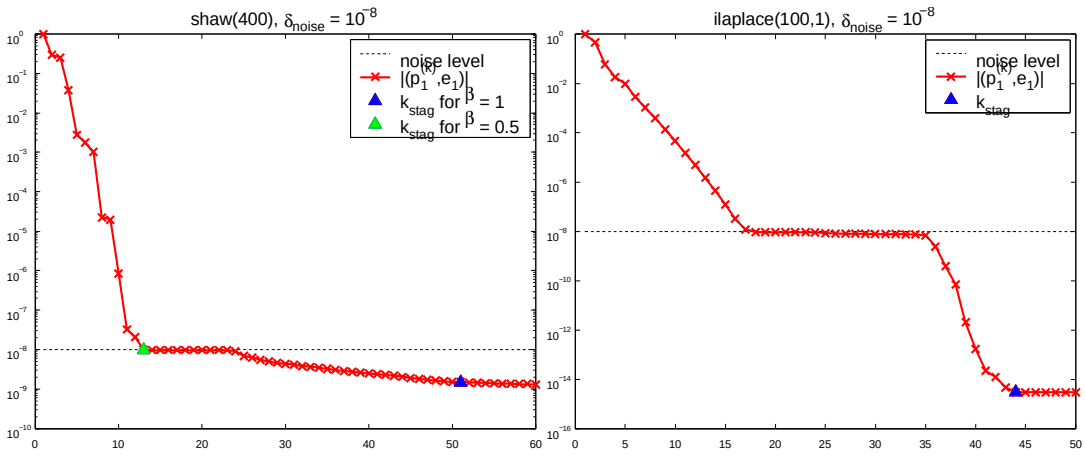


Figure 3.6: *Sequence* $|(p_1^{(k)}, e_1)|$ *and* k_{stag} *computed using the Minimum-Point criterion for the problems* Shaw *of the dimension 400 (left) and* Inverse Laplace *of the dimension 100 (right). For both problems the noise level is* $\delta_{noise} = 10^{-8}$

3.3.3 Minimum-Point Criterion

This approach [69] suggests minimizing the function:

$$\psi_\beta(k) = k \cdot |(p_1^{(k)}, e_1)|^\beta, \tag{3.18}$$

where $\beta > 0$ is a free real parameter, $k = 1, 2, \ldots$. The minimum of the function (3.18) determines k_{stag}. Two examples of how this method works are shown in Figure 3.6.

3.4 Noise Revealing in Action

In most of the cases we tested, noise iteration detection criteria give exactly the same results. However, in some ill-posed problems the "corner" of the curve cannot be easily detected. In Figure 3.7 the Closest-to-Origin criterion detected the corner at the 11^{th} iteration whereas the Minimum Point and Stagnation Based criteria detected it at the 10^{th} one. The difference can be much bigger when the curve made of values $|(p_1^{(k)}, e_1)|$ declines gradually with no "sharp enough" corner. This case is depicted in Figure 3.8. The difference between the Closest-to-Origin and the Minimum-Point criteria is 10 iterations whereas the Stagnation-Based criterion gave a completely different result. For a detailed comparison of these criteria see [69] and Chapter 4 of this thesis.

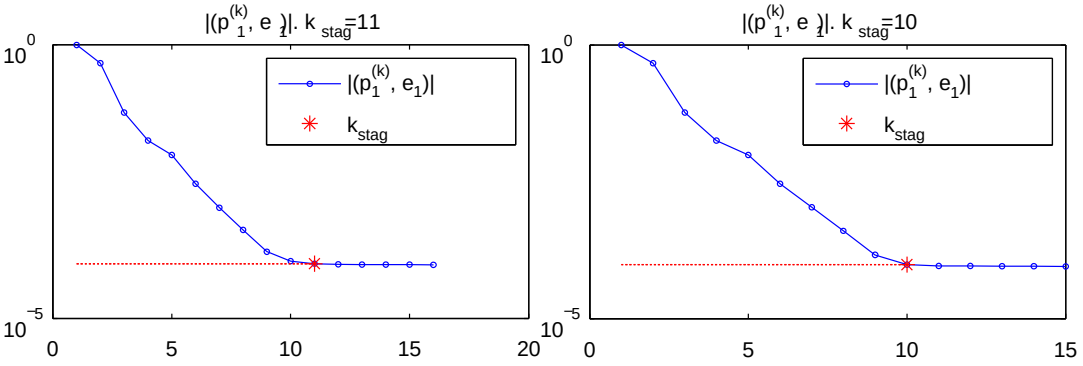


Figure 3.7: *The components* $|(p_1^{(k)}, e_1)|$ *of the* Inverse Laplace *problem with the noise level* 10^{-4}*, and the points detected by different noise iteration detection criteria: Closest-to-Origin (left), Minimum-Point and Stagnation-Based (right).*

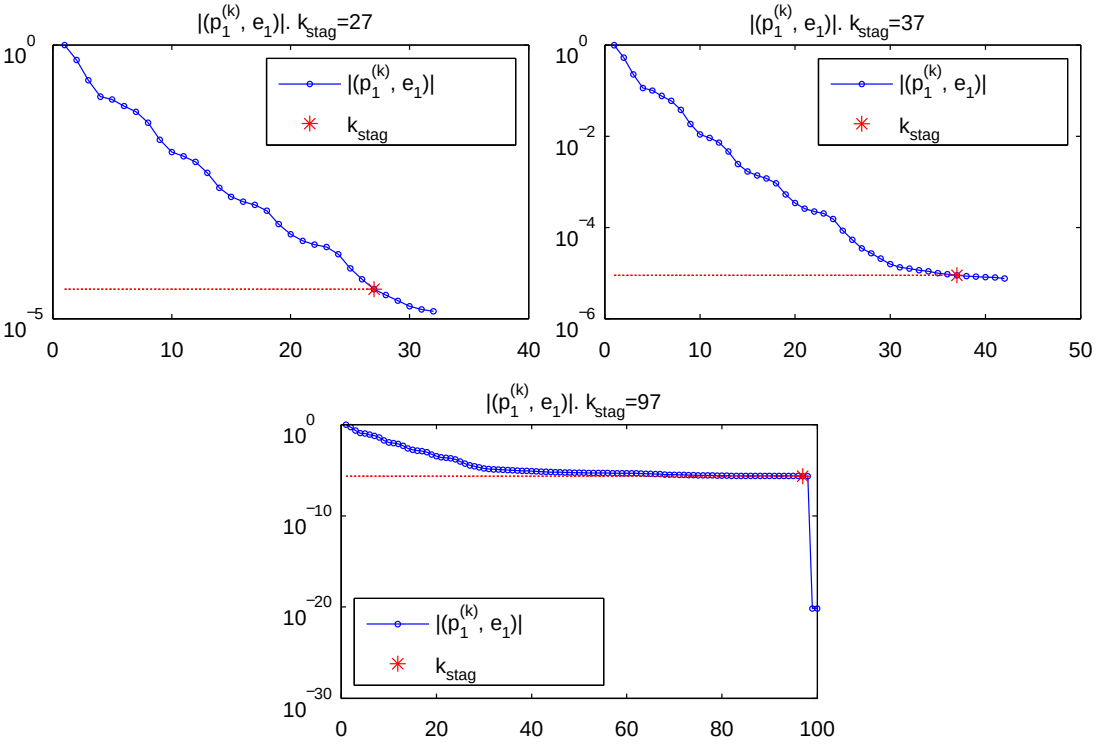


Figure 3.8: *The components* $|(p_1^{(k)}, e_1)|$ *of the* Heat *problem with the noise level* 10^{-5}*, and the points detected by different noise iteration detection criteria: Closest-to-Origin (top left), Minimum-Point (top right) and Stagnation-Based (bottom).*

Chapter 4

Regularization Parameter Choice Methods

For hybrid regularization it is important to choose also an appropriate inner regularization parameter for regularization of the projected problem obtained by GKB. This parameter can be either discrete (k) or continuous (λ) depending on the choice of the inner regularization method. From now on in this chapter we consider only the continuous case unless it is specified otherwise. For the discrete case the results can be obtained similarly. This chapter is based on the materials from [31], see also [66].

4.1 Discrepancy Principle

This principle is usually attributed to [52]. It is based on the a priori information about the required residual norm $\|e\|_2 = \|Ax - b\|_2$. In ill-posed problems usually $\|e\| = \|b_{noise}\|$, i.e. knowledge of the noise level is required. This noise level can be approximated using the formula (3.15). We seek for such a regularization parameter λ that for a defined top level of the residual norm δ_e it returns a solution x_λ satisfying:

$$\|Ax_\lambda - b\|_2 = \delta_e, \text{ where } \|e\|_2 \leq \delta_e. \tag{4.1}$$

This problem is solved using e.g. the LS method with quadratic constraint, described in [9], [53, §26] etc. In the discrete case the smallest k should be found, so that $\|Ax_\lambda - b\|_2 \leq \delta_e$.

4.2 GCV

Generalized cross-validation (GCV) [19, 71] is a method for regularization parameter choice that does not require a priori information about the noise level. It is based on the assumption that if during regularization some right-hand side entry b_i is not included into the solution, then this element should be predicted well by the regularized solution.

The method GCV minimizes the residual norm $\|Ax_\lambda - b_{exact}\|_2$, but because b_{exact} is not known the following function is used:

$$G(\lambda) = \frac{\|Ax_\lambda - b\|_2^2}{\text{trace}(I_m - AA^\#)^2}, \tag{4.2}$$

where $A^{\#}$ is a regularized inverse of A, namely:

$$x_\lambda = A^{\#} b. \tag{4.3}$$

4.3 The L-Curve Criterion

Like for GCV, for this criterion information about the noise level is not required. This method is based on the properties of the L-curve (see [31, Sections 4.6 and 7.5]). The L-function can be defined as a function of $\|Lx_{reg}\|_2$, where L is a regularization matrix (see [31, Section 1.3]) (usually we take $L \equiv I$), versus the corresponding residual norm $\|Ax_{reg} - b\|_2$, see Figure 4.1. The corner of the L-curve corresponds to the regularization parameters, which are close to optimal. They are located in the equilibrium zone of regularization errors and perturbation errors (see [31, Section 4.6] for details) in x_{reg}.

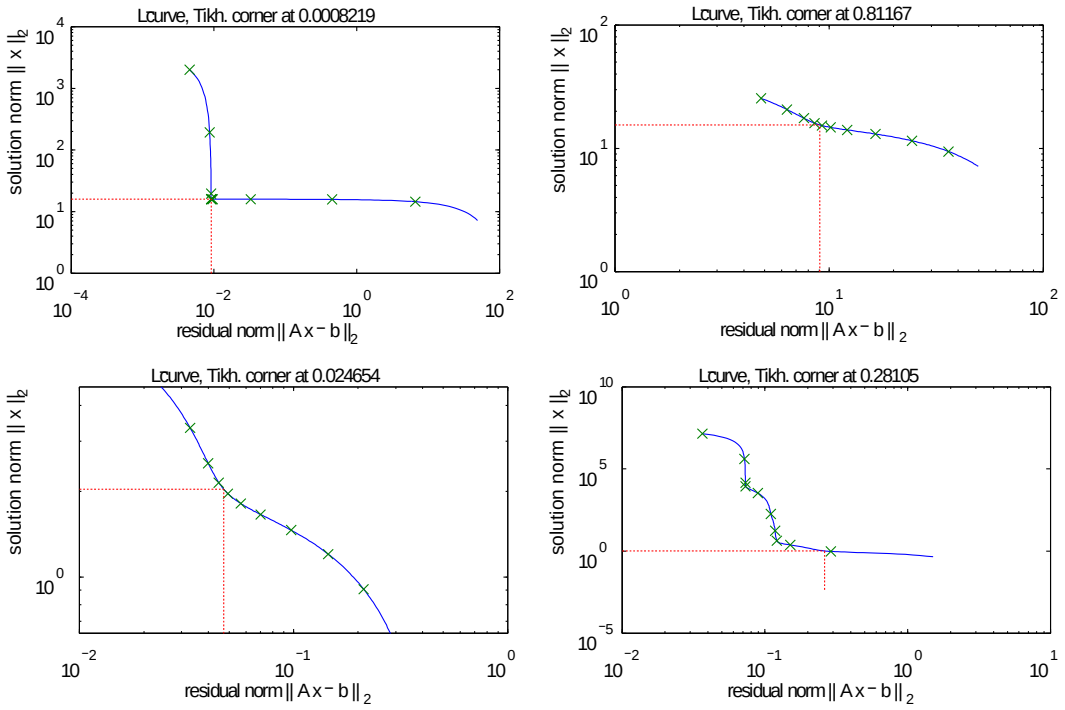


Figure 4.1: *Different forms of the L-curve in the log-log scale. The generic and the easiest form to work with is shown in the top left figure. On the top right and bottom left figures the corner is hard to determine because is is not sharp enough. On the bottom right figure there are several corners, that also makes determination of the corner problematic.*

The main idea of this method is the continuity of the dependency of $\|Lx_{reg}\|_2$ and $\|Ax_{reg} - b\|_2$ on the regularization parameter λ. This kind of dependency is typical for the Tikhonov and TSVD regularization methods. If the discrete Picard condition is satisfied, then usually L-curve consists of the vertical and adjacent horizontal parts (see Figure 4.1 top left); generally the curve can have other forms (see Figure 4.1 top right and bottom), but here we adhere to the first and the most generic form. A point on the vertical line corresponds to an

underregularized solution, i.e. a solution dominated by noise components. A point on the horizontal line corresponds to an overregularized solution, i.e. a solution with regularization errors or oversmoothed solution. Evidently, the optimal regularization parameter is located on a point of the transition curve from the horizontal line to the vertical line. The idea of the L-curve method is to find a point of equilibrium between these two error types. Note that that point is detected better on the log-log scale graph (see [31, Subsection 7.5.1]).

Chapter 5

Numerical Experiments

This chapter contains the results of numerical experiments run in the MATLAB environment v. 7.7.0 in the operating system Kubuntu 11.10, Linux kernel version 3.0.0, KDE version 4.7.4. Hardware characteristics: Intel® Core™ 2 Duo CPU T8300 @ 2.40 GHz, 4 GB of RAM, machine precision 2.2×10^{-16}.

In the program we used the iterative Golub-Kahan bidiagonalization algorithm with custom modifications developed by Iveta Hnětynková and Martin Plešinger 2005 – 2009. For details see Section 2.3 and [20]. MATLAB implementations of the test problems were taken from [69]. Colored noise was generated using the spatial pattern algorithm described in [48]. In the current work inner regularization methods (including inner regularization parameter finding methods) were added. Moreover, we developed a convenient user interface (see Section 5.1 for detailed description), that allows user to adjust parameters visually and see the results of each experiment. As an advantage, it is possible to easily extend this application by new testing problems and regularization approaches in the future.

The goal of this chapter is to show how hybrid regularization methods work depending on the choice of different noise iteration methods, number of additional GKB iterations g, inner regularization method and inner regularization parameter finding method. The experiments contain problems with a priori known solutions, so the exact and computed solutions can be compared. In the experiments the data tables contain levels of added noise in their rows, and numbers g of extra GKB iterations computed after the iteration k_{noise} used in hybrid regularization (horizontal header). The iteration index, where the GKB is stopped is defined as $k_{accepted}$. There are three important GKB iteration indexes k:

- k_{noise} represents the last iteration before the weights $|(p_1^{(k)}, e_1)|^2$ start to almost stagnate (see Section 3.2);
- k_{stag} is the first iteration where the weights $|(p_1^{(k)}, e_1)|^2$ start to almost stagnate (see Section 3.2);
- $k_{accepted}$ (or k_{accpt} in legends of figures) the iteration number in a hybrid regularization where the GKB is stopped.

The relations between these indexes are the following:

$$k_{stag} = k_{noise} + 1, \tag{5.1}$$

$$k_{accepted} = k_{noise} + g. \tag{5.2}$$

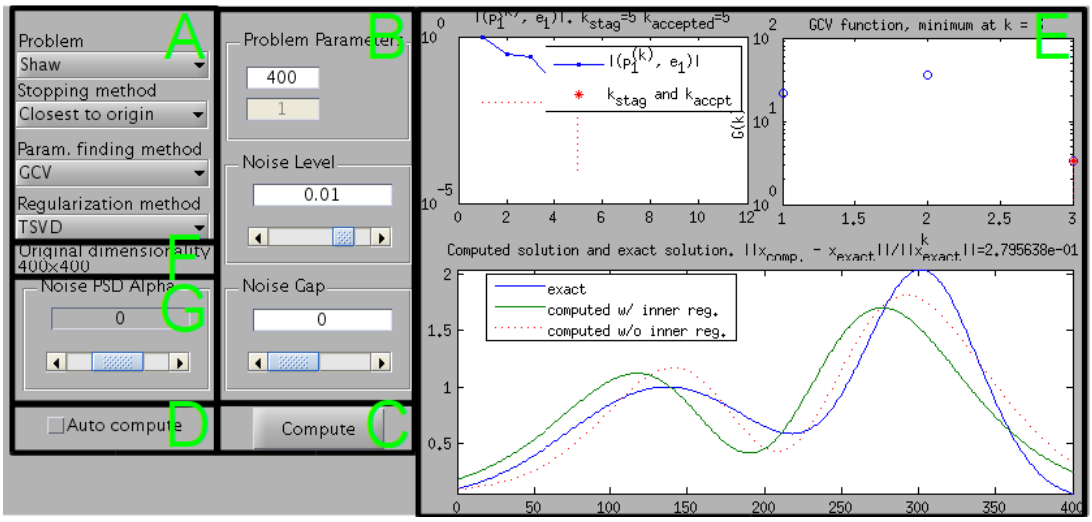


Figure 5.1: *The user interface of the application*

The data inside the cells represent relative approximation errors

$$\eta \equiv \|x_{comp} - x_{exact}\|/\|x_{exact}\|, \tag{5.3}$$

where x_{comp} and x_{exact} are the computed and the exact solution vectors respectively. In the first three experiments white noise is considered. The last experiment shows the results for different colors of noise.

The approximate solution cannot always be found, and in some cases the results may change significantly when repeating experiments, because of different noise components in b. Therefore the tables use the following markups.

- **Red number** (e.g. 1.7×10^{-1}) indicates that the result *is sometimes overregularized or underregularized* when repeating the experiment.
- **Wide red dash** (—). The result *is often or always overregularized or underregularized* when the experiment is repeated.
- **Blue number** (e.g. 1.7×10^{-1}). The solution is found, but when repeating the experiment *the error changes significantly.* In this case the biggest error (*not the average*) is put into the table.

Note that if the solution could not be found it was caused by one of two different reasons. One of them is that an incorrect inner regularization parameter was found (using GCV, L-curve or Discrepancy, see Chapter 4), the other is that the noise revealing iteration detection criterion failed (Closest-to-Origin, Minimum-Point or Stagnation-Based, see Chapter 3).

5.1 User Interface

The user interface consists of the following regions, see Figure 5.1. For description of main functions used in the application see Appendix.

A. **Descriptive problem and solution parameters.**

(i) **Selected problem.** It can be Shaw, Inverse Laplace, Gravity, Baart, Heat, Fox Good from the Regularization Toolbox [33].

(ii) **Noise iteration detection method.** The method to be used to find the noise revealing iteration k_{noise}: Closest-to-Origin, Minimum-Point and Stagnation-Based, [40, 69].

(iii) **Regularization parameter finding method.** This drop-down list defines which method is used to find the regularization parameter for the inner regularization. Possible values are: GCV [19, 71], [30, Section 5.3], [31, Section 7.4], L-curve [8, 29, 36, 61, 62], [30, Section 5.2] [31, Section 7.5] and Discrepancy [52], [30, Section 5.1], [31, Section 7.2].

(iv) **Regularization method.** This control allows user to select one of two available inner regularization methods: TSVD [37, 68] [31, Section 3.2] or Tikhonov [14, 24, 28, 58, 65, 67], [15, Chapter 5], [25, §5.1], [30, Section 4.3], [31, Section 5.1].

B. **Additional noise and problem parameters.**

(i) **General problem parameters.** This section contains "native" parameters of the selected problem, the parameters, which are used for generation of the matrix A and vectors b and x (1.2). The first parameter defines dimension of the problem. The second one is only the *Heat* problem-specific parameter, that defines conditioning of the problem ranging from 1 (ill-conditioned) to 5 (well-conditioned).

(ii) **Level of noise.** The amount of noise to be added.

(iii) **Noise gap.** The number of additional GKB iterations g after k_{noise} to be taken, see Section 2.4 and 3.2.

C. **Compute button.** Performs computation and displays the results.

D. **Auto-compute check-box.** When checked, computation is performed automatically each time user changes some parameter. Very handy, when user needs to see how some parameter affects the result. For time-consuming computations this check-box slows the system down, when user wants to change several parameters.

E. **The results area.** Contains 3 graphs.

(i) $|(\mathbf{p_1^{(k)}}, \mathbf{e_1})|$ **values.** They are used for k_{noise} determination, see Section 3.2.

(ii) **Regularization parameter finding method and regularization method-specific graph.** This graph displays minimization information, such as the functional used in GCV and its minimal point.

(iii) **Graph of the exact and approximate solution vectors.** Three solutions are presented here: exact (known for the testing problems a priori, when selecting one), computed with and without inner regularization, denoted by "w/" and "w/o" in the graphs respectively.

F. **Information section.** Displays the dimension of the projected problem.

G. **Noise color section.** Allows to choose the color of noise added to the problem using the parameter α. Here $\alpha = 1$ gives pink noise, $\alpha = 2$ – Brownian noise, $\alpha = -1$ – blue noise, $\alpha = -2$ – violet noise, and $\alpha = 0$ generates white noise, see Subsection 1.3.1.

5.2 Experiment 1. Comparison of the Noise Revealing Iteration Detection Methods

This experiment shows how choice of the noise iteration detection method (see Chapter 3) affects the results of computation. In this case the *Shaw* problem of the dimension 400 was selected, the inner regularization method was TSVD with GCV as the inner regularization parameter finding method.

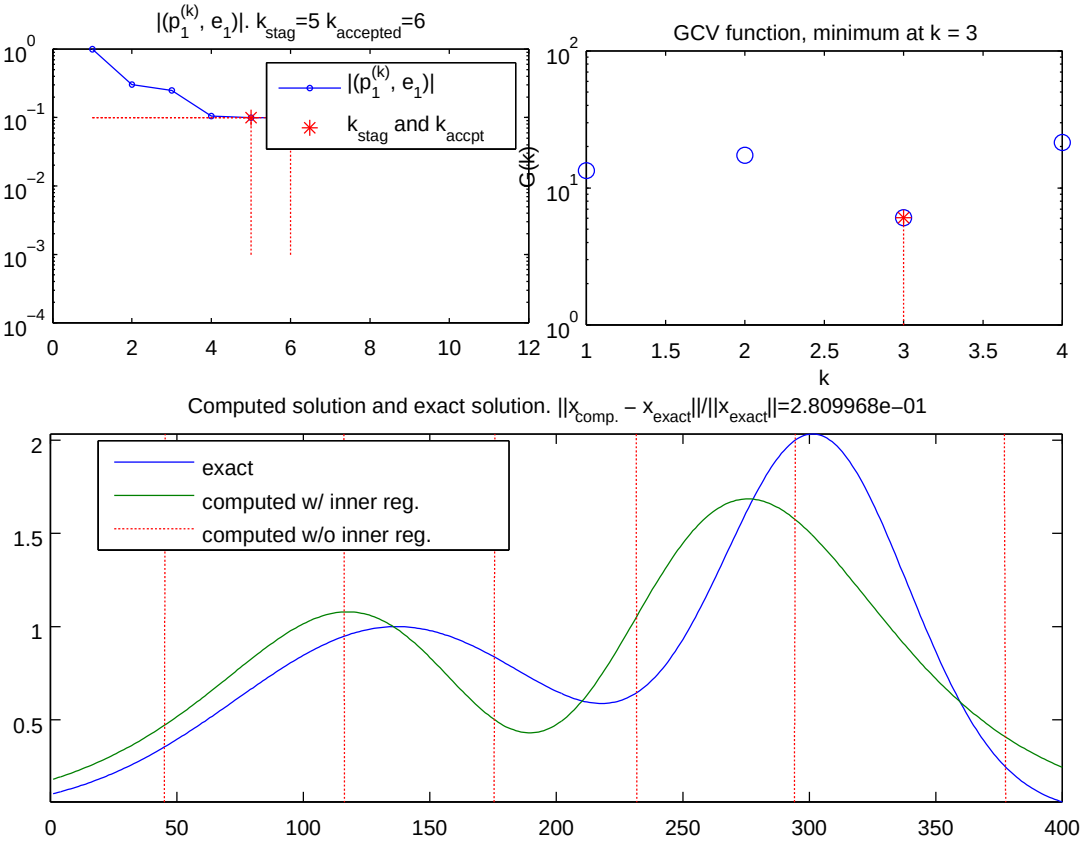


Figure 5.2: *The problem* Shaw *of the dimension 400 with* $\delta_{noise} = 0.1$ *and* $g = 1$*, the noise iteration detection criterion is Closest-to-Origin.*

	Closest-to-origin				Minimum-point				Stagnation-based			
g	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1
0	0.0321	0.147	0.28	0.67	0.321	0.147	0.28	0.663	0.0321	0.147	0.28	0.668
1	0.0321	0.0476	0.17	0.28	0.321	0.0476	0.17	0.663	0.0321	0.0476	0.17	0.668
2	0.0321	0.0476	0.17	0.28	0.321	0.0476	0.17	0.668	0.0321	0.0476	0.17	0.28
3	0.0321	0.0476	0.17	0.28	0.321	0.0476	0.17	0.668	0.0321	0.0476	0.17	0.28
4	0.0194	0.0476	0.17	0.28	0.194	0.0476	0.17	0.281	0.0321	0.0476	0.17	0.28
5	0.0194	0.0476	0.17	0.28	0.194	0.0476	0.17	0.281	0.0194	0.0476	0.17	0.28
10	0.0194	0.0476	0.17	0.28	0.194	0.0476	0.17	0.28	0.0194	0.0476	0.17	0.28
20	0.0194	0.0476	0.17	0.17	0.194	0.0476	0.17	0.17	0.0194	0.0476	0.17	0.28
30	0.0194	0.0476	0.17	—	0.194	0.0476	0.17	—	0.0194	0.0476	0.17	—

Table 5.1: *η values for the problem* Shaw *of the dimension 400 with TSVD regularization method and GCV inner regularization parameter finding method. Here and in the other tables in this chapter sets of measurements with different descriptive parameters are separated with double vertical lines. Column headers represent noise levels, row headers represent g.*

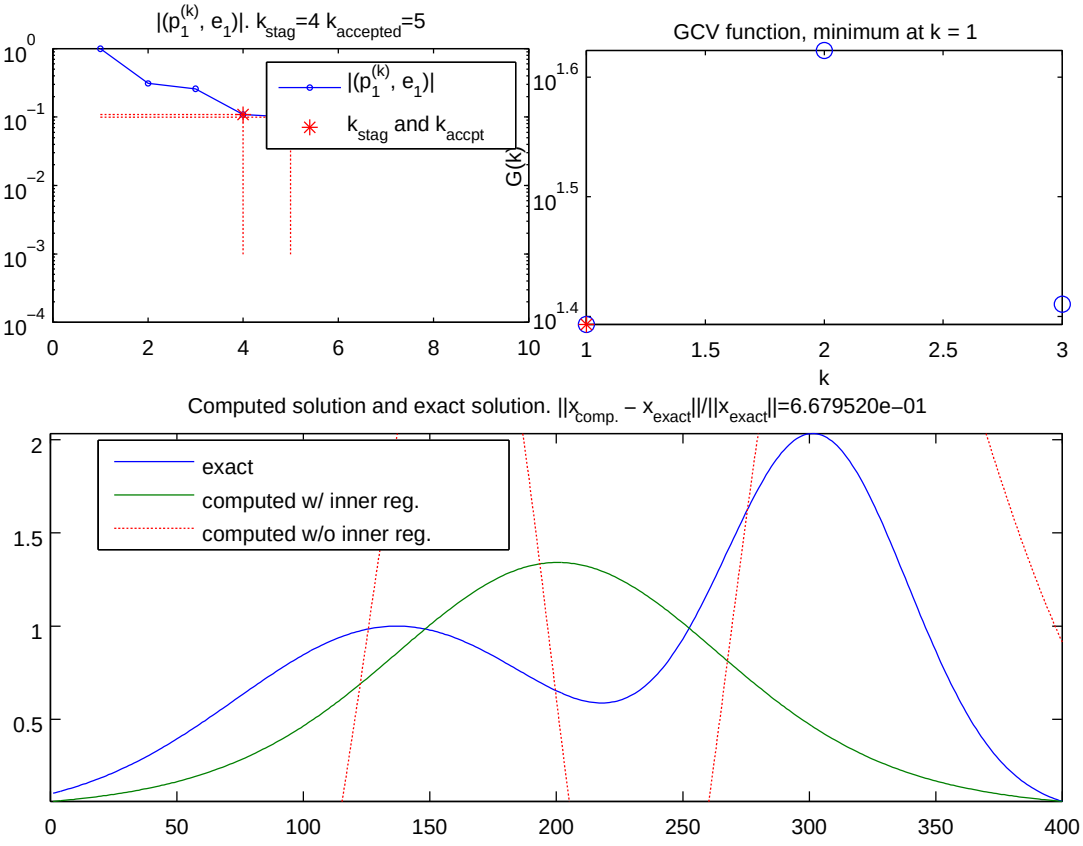


Figure 5.3: *The problem* Shaw *of the dimension 400 with $\delta_{noise} = 0.1$ and $g = 1$, the noise iteration detection criterion is Stagnation-Based.*

As we can see from Table 5.1, the values in different sets of measurements are very similar. This means that noise iteration detection methods are almost equally "good" in finding the iteration k_{noise}. It can also be seen that increasing g can improve the result. Obviously, there is no universal rule to define what exact number g is the most optimal one. In many cases taking $g = 1$ improved the solution, however, the other cases required several more iterations. The Table 5.1

is a good depiction of "obvious" cases when the solution cannot be found: when the noise level is very high and when g is relatively large. That can be seen in the bottom-right corners of each table section. This is not surprising. Taking too many additional GKB iterations (large g) causes the projected problem to be significantly contaminated by noise. In this case, inner regularization is often not able to sufficiently regularize the solution. We will see this also in the following experiments.

In Figures 5.2, 5.3 and 5.4 there is an example of the case with the noise level 0.1, where the noise iteration detection methods give different results. In this case the Closest-to-Origin method gave the most satisfactory result. Two other solutions are overregularized because the detected noise revealing iterations are too small.

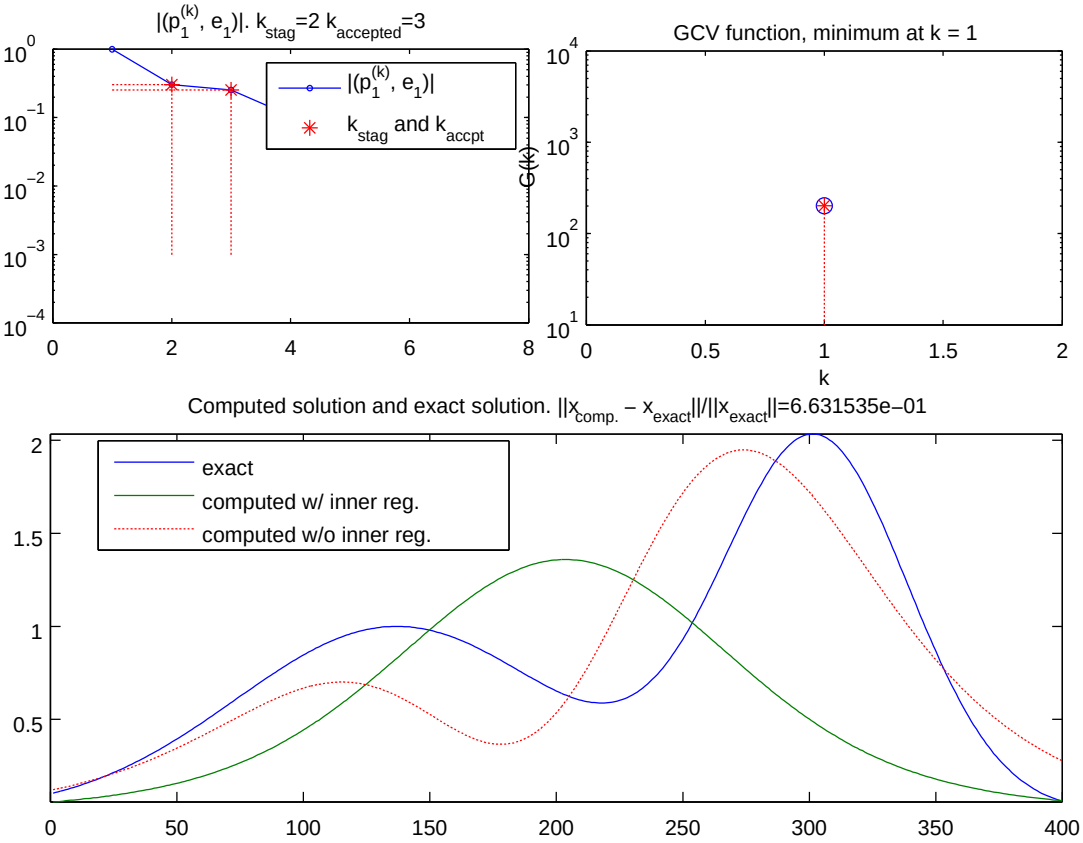


Figure 5.4: *The problem* Shaw *of the dimension 400 with* $\delta_{noise} = 0.1$ *and* $g = 1$*, the noise iteration detection criterion is Minimum-Point.*

5.3 Experiment 2. The Effect of Choosing Different Inner Regularization and Parameter Finding Methods

Inner regularization methods and inner regularization parameter finding methods are strongly interrelated, so it is better to analyze them in combinations. In this

experiment we tested the problem *Shaw* of the dimension 400 using fixed noise iteration detection criterion Closest-to-Origin, to study inner regularization in GKB separately from other parameters.

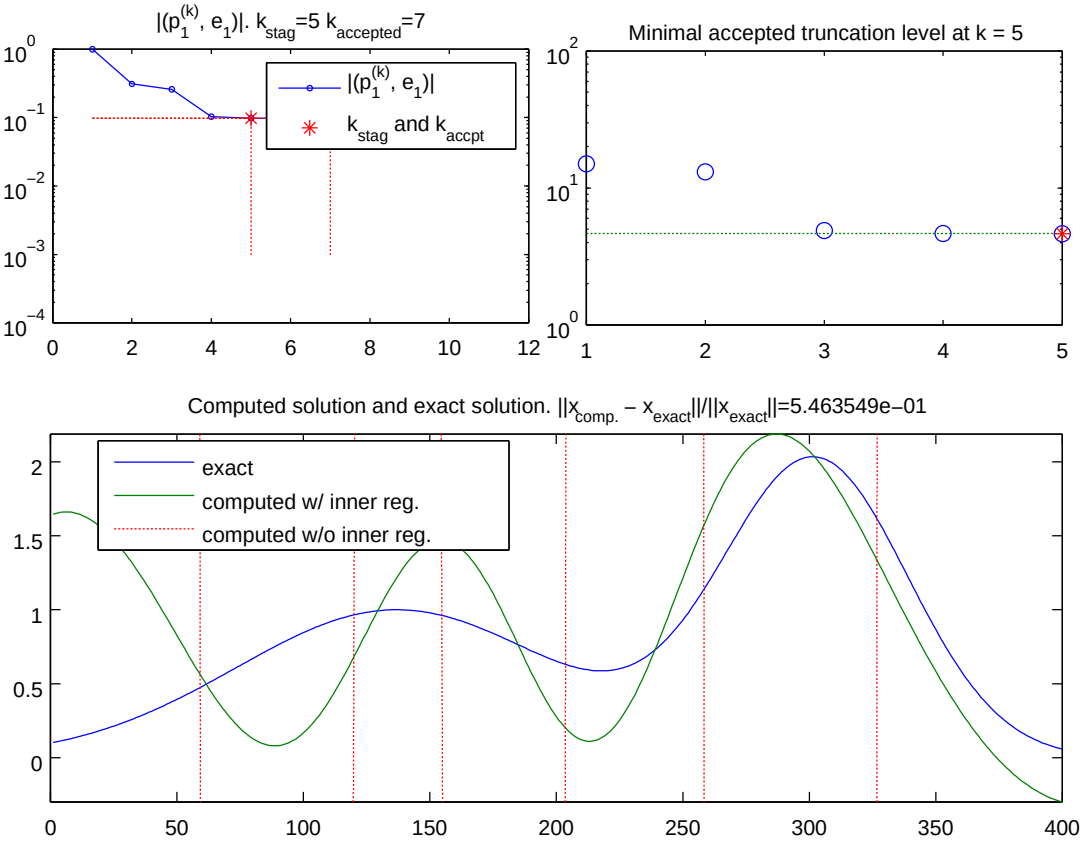


Figure 5.5: *The problem* Shaw *of the dimension 400 with* $\delta_{noise} = 0.1$ *and* $g = 2$*, the TSVD inner regularization method is used.*

	GCV				L-curve				Discrepancy			
g	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1
0	0.0321	0.147	0.28	0.67	0.1460	0.1470	0.17	—	0.039	0.0475	0.169	—
1	0.0321	0.0476	0.17	0.28	0.0321	0.1470	0.16	0.28	—	—	—	—
2	0.0321	0.0476	0.17	0.28	0.0194	0.0476	0.17	0.28	0.02	0.047	0.14	0.4
3	0.0321	0.0476	0.17	0.28	0.1	0.0476	0.17	0.28	0.02	0.047	0.15	0.4
4	0.0194	0.0476	0.17	0.28	0.1	0.2	0.06	0.28	0.03	0.047	0.15	0.4
5	0.0194	0.0476	0.17	0.28	0.3	0.2	0.1	0.28	0.03	0.047	0.15	0.5
10	0.0194	0.0476	0.17	0.28	0.09	0.2	0.1	—	0.03	0.047	0.15	0.3
20	0.0194	0.0476	0.17	0.17	0.06	0.3	0.1	—	0.03	0.048	0.15	0.4
30	0.0194	0.0476	0.17	—	0.08	0.3	0.2	—	0.03	0.048	0.15	0.4

Table 5.2: *η values for the problem* Shaw *of the dimension 400 with the Closest-to-Origin noise iteration detection criterion and the TSVD inner regularization method. Different inner regularization parameter finding methods (GCV, L-curve, discrepancy principle) are tested. For a description of the headers see Table 5.1.*

	GCV				L-curve				Discrepancy			
g	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1
0	0.02	0.076	0.21	0.4	0.026	0.076	0.21	0.2	0.02	0.05	0.17	0.3
1	0.028	0.04	0.14	0.26	0.03	0.05	0.1	0.2	0.02	0.04	0.14	0.2
2	0.027	0.046	0.16	0.23	0.02	0.03	0.1	0.2	0.02	0.04	0.15	0.2
3	0.025	0.046	0.16	0.2	0.05	0.05	0.1	0.2	0.02	0.04	0.15	0.2
4	0.023	0.046	0.16	0.2	0.06	0.04	0.1	0.2	0.02	0.04	0.14	0.2
5	0.021	0.046	0.16	0.2	0.04	0.05	0.1	0.2	0.02	0.04	0.14	0.2
10	0.02	0.04	0.15	0.2	0.08	0.04	0.1	0.2	0.02	0.04	0.15	0.2
20	0.02	0.04	0.15	0.2	0.04	0.05	0.1	0.2	0.02	0.04	0.14	0.2
30	0.02	0.04	0.1	0.2	0.05	0.05	0.1	0.2	0.02	0.04	0.14	0.2

Table 5.3: *η values for the problem* Shaw *of the dimension 400 with the Closest-to-Origin noise iteration detection criterion and the Tikhonov inner regularization method. Different inner regularization parameter finding methods (GCV, L-curve, discrepancy principle) are tested. For a description of the headers see Table 5.1.*

Looking at Tables 5.2 and 5.3 we can conclude, that the Tikhonov inner regularization gives more accurate results even with more noise added to the problem. However it is worth to note, that in many cases it gives solutions that change when repeating the experiment (i.e. taking different noise vectors). Nevertheless, the combination of Tikhonov regularization and GCV yields the best results in this experiment.

In Figure 5.5 we can see that in some cases the TSVD method can give overregularized results. Moreover, they change significantly when repeating the experiment. Figure 5.6 shows that if we change the regularization method to Tikhonov the results become more accurate.

The combination of TSVD and the discrepancy principle showed unusual behavior. When adding one extra iteration the solution could not be found, but adding more extra ones helped, see Table 5.2, Discrepancy.

In Table 5.2 for the L-curve and the discrepancy principle for $\delta_{noise} = 0.1$ the solution could not be found using TSVD inner regularization method for $g = 0$. It confirms what we expected, i.e. that using extra GKB iterations can improve the result. We can see in all tables of this chapter, that applying inner regularization for $g = 0$ (i.e. GKB is stopped before noise significantly enters the projected problem) usually does not give as satisfactory result as adding several extra iterations to GKB ($g > 0$) and then applying inner regularization, see Section 3.2 for explanation.

5.4 Experiment 3. Application on Other Ill-Posed Problems

In this section we consider different ill-posed problems using the same hybrid solution method: the Closest-to-Origin noise iteration detection criterion, the TSVD inner regularization method and the GCV inner regularization parameter finding method. The goal is to study the behavior of the method for different problems.

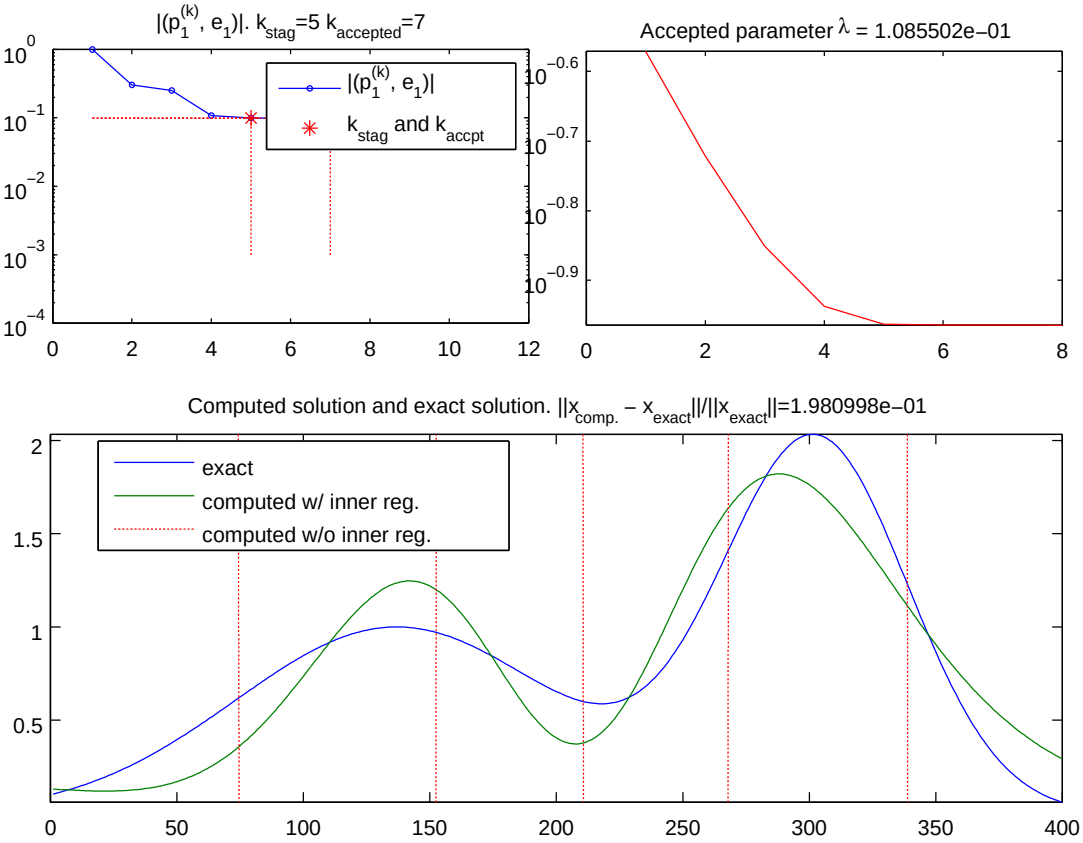


Figure 5.6: *The problem* Shaw *of the dimension 400 with* $\delta_{noise} = 0.1$ *and* $g = 2$*, the Tikhonov inner regularization method is used.*

	Shaw (400)				Inverse Laplace (100)				Gravity (100)			
g	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1
0	0.0321	0.147	0.28	0.67	0.5392	0.109	0.2	0.28	0.0031	0.0187	0.083	0.23
1	0.0321	0.0476	0.17	0.28	0.0534	0.109	0.2	0.28	0.0032	0.0193	0.088	0.25
2	0.0321	0.0476	0.17	0.28	0.042	0.094	0.2	0.28	0.0032	0.0136	0.088	0.25
3	0.0321	0.0476	0.17	0.28	0.042	0.094	0.2	0.28	0.0032	0.0136	0.088	0.25
4	0.0194	0.0476	0.17	0.28	0.042	0.094	0.2	0.28	0.0032	0.0136	0.088	0.25
5	0.0194	0.0476	0.17	0.28	0.042	0.094	0.2	0.28	0.0023	0.0136	0.061	0.17
10	0.0194	0.0476	0.17	0.28	0.042	0.09	0.2	0.23	0.0023	0.0136	0.061	0.17
20	0.0194	0.0476	0.17	0.17	0.031	0.09	0.2	0.23	0.0023	0.01	0.061	0.17
30	0.0194	0.0476	0.17	—	—	—	—	0.23	0.0023	0.01	0.061	0.09

Table 5.4: *Comparison of three problems with the Closest-to-Origin noise iteration detection criterion, TSVD inner regularization method and GCV inner regularization parameter finding method. The problems are square with the dimension given in parentheses. For the description of the headers see Table 5.1.*

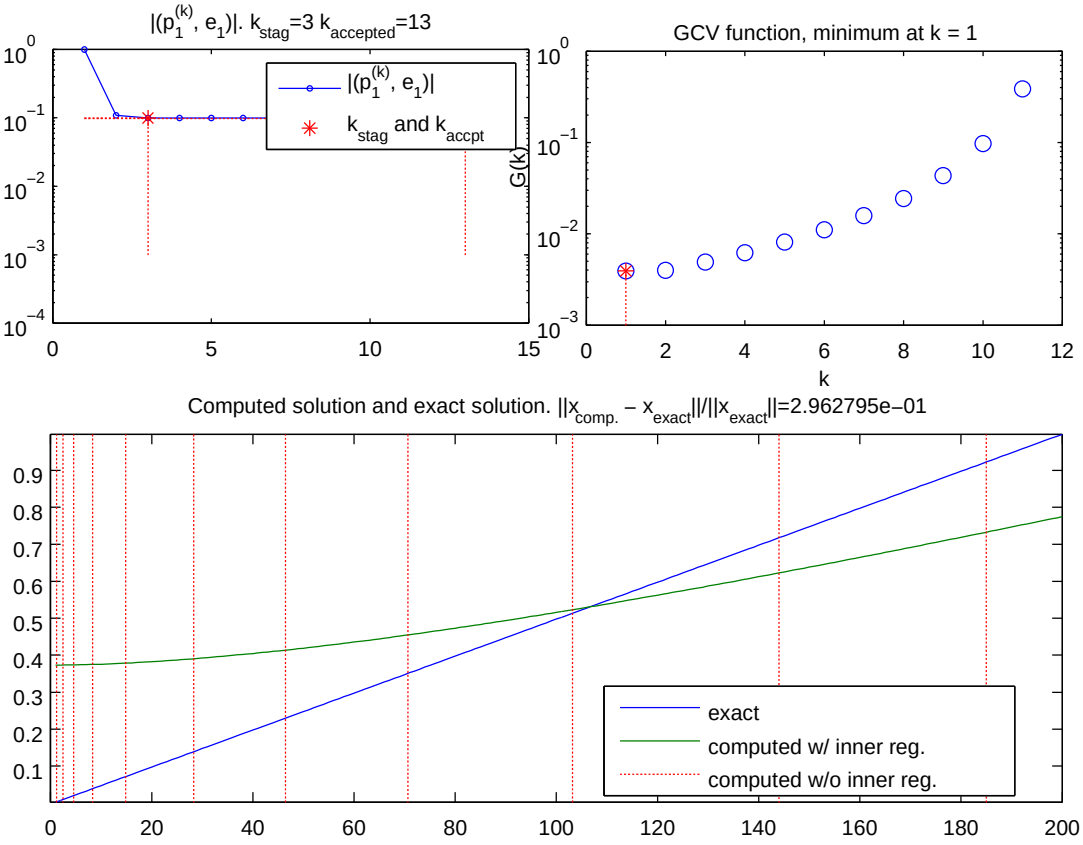


Figure 5.7: *The problem* Fox Good *of the dimension 200 with* $\delta_{noise} = 0.1$ *and* $g = 10$.

	Baart (100)				Heat (100, 1)				Fox Good (200)			
g	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1
0	0.1146	0.1662	0.345	—	0.0213	0.047	0.29	—	0.0023	0.0311	—	—
1	0.0524	0.115	0.345	—	0.0188	0.04	0.25	—	0.0023	0.0073	0.03	—
2	0.0524	0.115	0.345	—	0.0177	0.05	0.25	—	0.0023	0.0073	0.03	—
3	0.0524	0.115	0.17	—	0.0168	0.04	0.25	—	0.0023	0.0074	0.03	—
4	0.0524	0.115	0.17	—	0.0158	0.02	0.25	—	0.0023	0.0073	0.03	—
5	0.0524	0.115	0.18	—	0.0153	0.04	0.25	—	0.0023	0.0073	0.03	—
10	0.0524	0.115	0.17	—	0.015	0.02	0.25	—	0.0009	0.0073	0.03	—
20	0.0524	0.115	0.17	—	0.015	0.02	0.15	0.4	0.0009	0.0073	0.03	0.04
30	0.0524	0.115	0.17	—	0.015	0.02	0.15	0.3	0.0009	0.0074	0.03	0.05

Table 5.5: *Comparison of three problems with the Closest-to-Origin noise iteration detection criterion, TSVD inner regularization method and GCV inner regularization parameter finding method. The problem matrices are square with the dimension given in parentheses. The second parameter for the* Heat *problem influences the ill-posedness of the problem. For the description of the headers see Table 5.1.*

As we can see from Tables 5.4 and 5.5 the selected hybrid method worked quite well for other problems too. Problems only appeared for the *Inverse Laplace* problem for the larger values of g (see Table 5.4) and the *Baart* problem with very high level of noise: 0.1 (see Table 5.5), see the explanation in Section 5.2. For the *Heat* problem with the same level of noise adding more extra noisy components even helped to find a solution with a good relative error η (5.3). For the

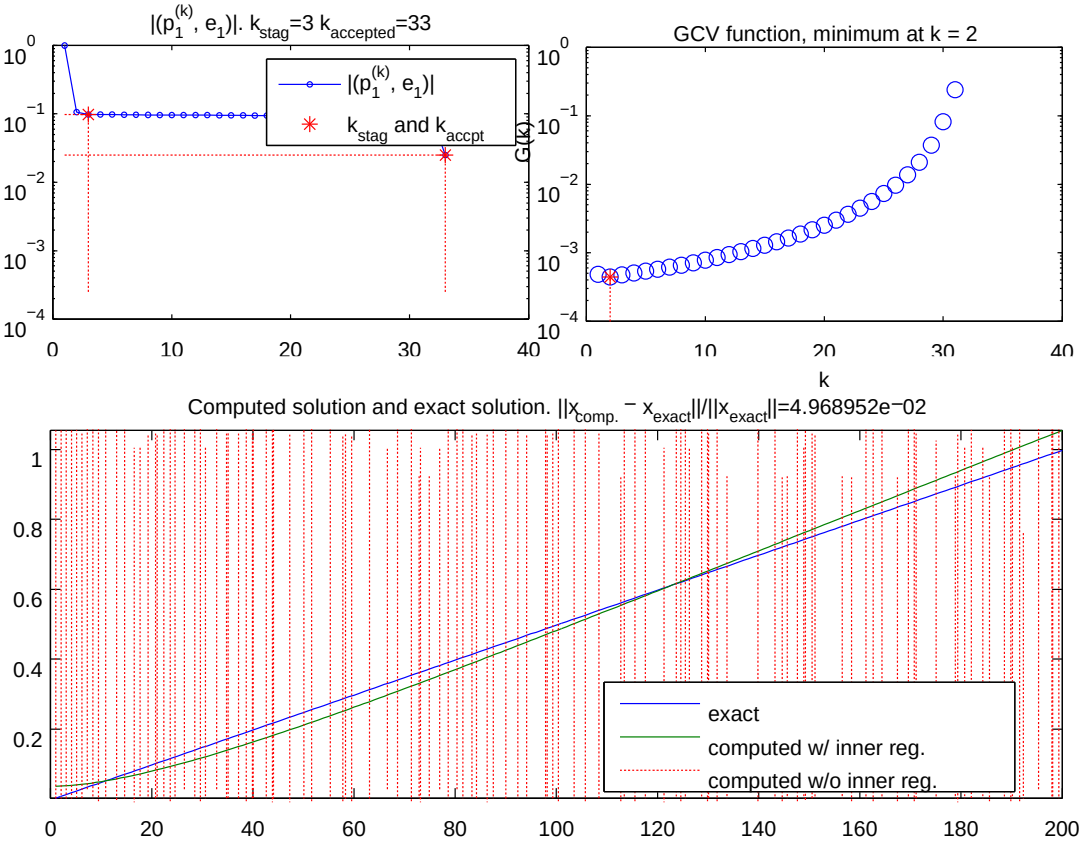


Figure 5.8: *The problem* Fox Good *of the dimension 200 with* $\delta_{noise} = 0.1$ *and* $g = 30$.

Fox Good problem the situation was similar. This is depicted in Figures 5.7 and 5.8, showing the results for 10 and 30 additional iterations, respectively.

5.5 Experiment 4. Regularization for Problems with Different Colors of Noise

Consider the problem *Baart* of the dimension 100 with the Closest-to-Origin noise iteration detection criterion, the TSVD inner regularization method and the GCV inner regularization parameter finding method. In this experiment we study the behavior of the hybrid method for different colors of noise present in the problem. Different levels of noise were tested.

	Brownian Noise, $\alpha = 2$				Pink Noise, $\alpha = 1$				White Noise, $\alpha = 0$			
g	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1
0	0.114	0.116	0.345	—	0.1146	0.116	0.345	—	0.1146	0.1662	0.345	—
1	0.167	0.12	0.345	—	0.0526	0.116	0.345	0.36	0.0524	0.115	0.345	—
2	0.997	0.3	—	—	0.525	0.118	0.345	0.37	0.0524	0.115	0.345	—
3	0.079	0.592	—	—	0.526	0.117	0.279	0.38	0.0524	0.115	0.17	—
4	0.136	0.521	—	—	0.526	0.116	0.345	0.39	0.0524	0.115	0.17	—
5	0.077	—	—	—	0.525	0.116	0.26	0.39	0.0524	0.115	0.18	—
10	—	—	—	—	0.525	0.115	0.238	—	0.0524	0.115	0.17	—
20	—	—	—	—	0.1106	—	—	—	0.0524	0.115	0.17	—
30	—	—	—	—	—	—	—	—	0.0524	0.115	0.17	—

Table 5.6: *The effect of having various noise colors in the the problem* Baart *of the dimension 100, the Closest-to-Origin noise iteration detection criterion, TSVD inner regularization method and GCV inner regularization parameter finding method. Three different colors of noise are considered (Brownian, pink and white). For the description of the headers see Table 5.1.*

	White Noise, $\alpha = 0$				Blue Noise, $\alpha = -1$				Violet Noise, $\alpha = -2$			
g	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1	10^{-6}	10^{-4}	0.01	0.1
0	0.1146	0.166	0.345	—	0.1146	0.166	0.345	—	0.1146	0.166	0.345	—
1	0.0524	0.115	0.345	—	0.0524	0.115	0.345	—	0.0524	0.115	0.345	—
2	0.0524	0.115	0.345	—	0.0524	0.115	0.345	0.345	0.0524	0.115	0.345	0.345
3	0.0524	0.115	0.17	—	0.0524	0.115	0.166	0.345	0.0524	0.115	0.166	0.345
4	0.0524	0.115	0.17	—	0.0524	0.115	0.166	0.345	0.0524	0.115	0.166	0.345
5	0.0524	0.115	0.18	—	0.0524	0.115	0.166	0.345	0.0524	0.115	0.166	0.345
10	0.0524	0.115	0.17	—	0.0524	0.115	0.166	0.345	0.0524	0.115	0.166	0.345
20	0.0524	0.115	0.17	—	0.0524	0.115	0.166	0.345	0.0524	0.115	0.166	0.345
30	0.0524	0.115	0.17	—	0.0524	0.115	0.166	0.345	0.0524	0.115	0.166	0.345

Table 5.7: *The effect of having various noise colors in the problem* Baart *of the dimension 100, the Closest-to-Origin noise iteration detection criterion, TSVD inner regularization method and GCV inner regularization parameter finding method. Three different colors of noise are considered (white, blue and violet). For the description of the headers see Table 5.1.*

As we can see from Tables 5.6 and 5.7 noise color can play significant role in the solution process (compare the values in Table 5.6). However, in some cases it may not have any impact (see Table 5.7, blue and violet noise colors). In general, colored noise can be divided into two groups (see Subsection 1.3.1 and (1.17)): with $\alpha > 0$ (Pink and Brownian) and with $\alpha < 0$ (Blue and Violet). For positive α the PSD of noise has decreasing character. As it was examined in the work [69, Section 3.4] and also [50] for $\alpha > 0$ the sequence $|(p_1^{(k)}, e_1)|$ starts to almost stagnate at the level that is lower than the level of noise. That means that extra noise iterations can be used in the solution process. For negative α the results were very similar to the results for the problems contaminated by white noise, even some improvement was observed. Let us take a more detailed look at this phenomenon.

The reason of such behavior of this sequence is as follows. In [69, Section 1.5] it is shown how the high-frequency part of noise is revealed in the vectors s_k of the GKB, see also Section 3.2 of this thesis. Since the Fourier coefficients for

noise with $\alpha < 0$ are higher for higher frequencies, the high-frequency part is amplified even more than for white noise.

For noise with $\alpha > 0$ low-level frequencies dominate, therefore the amplification of the high frequencies is not significant compared to white noise. Moreover, the values of the first projections $|(\frac{b_{noise}}{\|b\|}, u_j)|^2$ are important for noise level determination. As it was mentioned in Section 3.2 there exists an iteration j_{noise} so that for all $j \geq j_{noise}$ the noisy part of the right-hand side b becomes dominant. From (3.11) and (3.12) we obtain:

$$\delta^2_{noise} \approx \sum_{j=1}^{j_{noise}-1} \left| \frac{b_{noise}}{\|b\|}, u_j \right|^2 + \delta^2, \tag{5.4}$$

where δ_{noise} is the level of noise. In Figure 5.9 the behavior of δ^2 as a function of j_{noise} for noise of different colors is shown for the problems *Gravity* and *Shaw*. As we can see from Figure 5.9, for $\alpha < 0$ the behavior of δ^2 is similar to white noise. For noise with $\alpha > 0$ the function δ^2 decreases for the small j_{noise} values and then starts to stagnate. And therefore the information about the level of noise is located in the first iterations of $|\frac{b_{noise}}{\|b\|}, u_j|^2$. Thus if we use (3.13) in a direct way, the noise level is underestimated and often that leads to underregularized solution.

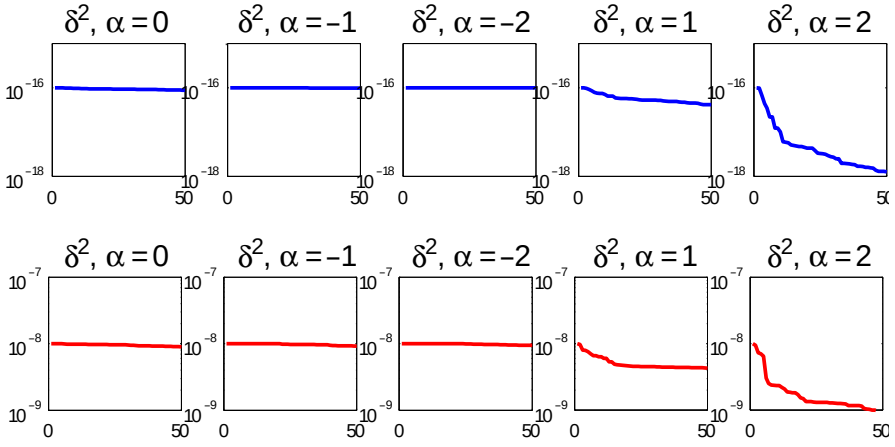


Figure 5.9: *The value δ^2 as a function of j_{noise} for the problem* Gravity *of the dimension 400 (top) with the noise level 10^{-8} and* Shaw *of the dimension 400 (bottom) with the noise level 10^{-4}, for various colors of noise (α).*

Conclusion

In this thesis, we studied hybrid regularization methods using GKB as the outer regularization algorithm with the inner regularization algorithms TSVD and Tikhonov, respectively. Methods were tested on a number of ill-posed problems from the Regularization Toolbox [33] with different noise levels and noise colors. The effects of choosing different noise revealing iteration detection methods and inner regularization parameter finding methods were tested. The hybrid regularization method proved its effectiveness. The most optimal values for g (the number of extra GKB iterations after the noise revealing iteration k_{noise}) were roughly between 2 and 5, but the choice can sometimes be very problem-specific. In some cases (like for the *Heat* and *Fox Good* problems for the noise level 0.1) unusual behavior was observed. Here, large g (between 20 and 30) was required to obtain a satisfactory approximate solution. Testing the method on different noise levels and colors showed the results similar to [69] and [50]. The method worked quite well for white noise and high-frequency dominated noise, but often failed for noise of low-frequency, because in that case noise level determination methods underestimate the noise level.

In the future, the application can be easily extended, by more testing problems and modifications of the hybrid method. Especially, the core GKB algorithm could allow choosing, except of the currently available full or no reorthogonalization, more reorthogonalization approaches (partial, selective etc). Noise level determination for low-frequency dominated noise can be further studied.

Bibliography

[1] E. Anderson, Z. Bai, C. Bishof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, O. Ostrouchow and D. Sorensen, *LAPACK User's Guide*, Second Edition, SIAM, Philadelphia, 1995.

[2] C. T. H. Baker, *The Numerical Treatment of Integral Equations*, Clarendon Press, Oxford, UK, 1977.

[3] M. W. Berry and A. Sameh, *An Overview of Parallel Algorithms for the Singular Value and Symmetric Eigenvalue Problems*, J. Comput. Appl. Math. 27, 1989, pp. 191–213.

[4] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[5] Å. Björck, *A Bidiagonalization Algorithm for Solving Large and Sparse Ill-Posed Systems of Linear Equations*, BIT 28, 1988, pp. 659–670.

[6] Å. Björck, T. Elfving, and Z. Strakoš, *Stability of Conjugate Gradient and Lanczos methods for linear least squares problems*, SIAM J. Matrix Anal. Appl. 19(3), 1998, pp. 720–736.

[7] Å. Björck, E. Grimme and P. Van Dooren, *An Implicit Shift Bidiagonalization Algorithm for Ill-Posed Systems*, BIT 34, 1994, pp. 510–534.

[8] A. S. Carasso, *Overcomming Hölder Discontinuity in Ill-Posed Continuation Problems*, SIAM J. Numer. Anal. 31, 1994, pp. 1535–1557.

[9] T. F. Chan, J. Olkin and D. W. Cooley, *Solving Quadratically Constrained Least Squares Using Black Box Unconstrained Solvers*, BIT 32, 1992, pp. 481–495.

[10] L. M. Delves and J. L. Mohammed, *Computational Methods for Integral Equations*, Cambridge University Press, Cambridge, UK, 1985.

[11] L. M. Delves and J. Walsh (Eds.), *Numerical Solution of Integral Equations*, Clarendon Press, Oxford, 1974.

[12] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[13] H. Engels, *Numerical Quadrature and Cubature*, Academic Press, Aachen, 1980.

[14] H. W. Engl, *Regularization Methods for the Stable Solution of Inverse Problems*, Surveys Math. Indust. 3, 1993, pp. 71–143.

[15] H. W.Engl, M. Hanke and A. Neubauer, *Regularization of Inverse Problems*, Kluwer, Dordrecht, the Netherlands, 1996.

[16] A. Ern and J. L. Guermond, *Theory and practice of finite elements*, Springer, 2004.

[17] R. D. Fierro, G. H. Golub, P. Ch. Hansen and D. P. O'Leary, *Regularization by Truncated Total Least Squares*, SIAM J. Scient. Comp. 18, 1997, pp. 1223–1241.

[18] B. G. Galerkin, *Journal of Engineers* (in Russian), 19 (vol. 1), 1915, pp. 897–908.

[19] G. H. Golub, M. T. Heath and G. Wahba, *Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter*, Thechnometrics 21, 1979, pp. 215–223.

[20] G. H. Golub and W. Kahan, *Calculating the Singular Values and Pseudo-Inverse of a Matrix*, SIAM J. Num. Anal. Ser. B 2, 1965, pp. 205–224.

[21] G. H. Golub and P. Van Dooren, *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, NATO ASI Series, Springer-Verlag, Berlin, 1991.

[22] G. H. Golub and C. F. Van Loan *Matrix Computations*, Third Edition, the Johns Hopkins University Press, Baltimore, MD, 1996.

[23] R. M. Gray and L. D. Davisson *An Introduction to Statistical Signal Processing*, Cambridge University Press, 2004.

[24] C. W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Research Notes in Mathematics 105, Pitman, Boston, 1984.

[25] C. W. Groetsch, *Inverse Problems in the Mathematical Sciences*, Vieweg, Wiesbaden, Germany, 1993.

[26] S. Hammarling, *Parallel Algorithms for Singular Value Problems*, in [21], pp. 173–187.

[27] M. Hanke, *On Lanczos Based Methods for the Regularization of Discrete Ill-Posed Problems*, BIT 5 (vol. 41), 2001, pp. 1008–1018.

[28] M. Hanke and P. C. Hansen, *Regularization Methods for Large-Scale Problems*, Surveys Math. Indust. 3, 1993, pp. 253–315.

[29] P. C. Hansen, *Analysis of Discrete Ill-Posed Problems by Means of the L-Curve*, SIAM Review 34, 1992, pp. 561–580.

[30] P. C. Hansen, *Discrete Inverse Problems. Insight and Algorithms. A Tutorial with MATLAB exercises*, DTU Inromatics, Technical University of Denmark. 2008.

[31] P. C. Hansen, *Rank-deficient and Discrete Ill-posed Problems - Numerical Aspects of Linear Inversion*, Philadelphia, 1997.

[32] P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT 30, 1990, pp. 658–672.

[33] P. C. Hansen, *Regularization Tools, a Matlab Package for Analysis of Discrete Regularization Problems*, Numerical Algorithms 6, 1994, pp. 1–35.

[34] P. C. Hansen and T. K. Jensen, *Noise propagation in regularizing iterations for image deblurring*, ETNA, Vol.33, 2008, pp. 204–220.

[35] P. C. Hansen, J. G. Nagy and D. P. O'Leary, *Deblurring Images – Matrices, Spectra and Filtering*, SIAM, Philadelphia, 2006.

[36] P. C. Hansen and D. P. O'Leary, *The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems*, SIAM J. Sci. Comput. 14, 1993, pp. 1487–1503.

[37] R. J. Hanson, *A Numerical Method for Solving Fredholm Integral Equations of the First Kind Using Singular Values*, SIAM J. Numer. Anal. 8, 1971, pp. 616–622.

[38] M. R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, J. Res. Nat. Bur. Standards, 49, 1952, pp. 409–435.

[39] I. Hnětynková and Z. Strakoš, *Lanczos Tridiagonalization and Core Problems.* Linear Algebra Appl. 421, 2007, pp. 243–251.

[40] I. Hnětynková, M. Plešinger and Z. Strakoš, *The Regularizing Effect of the Golub-Kahan Iterative Bidiagonalization and Revealing the Noise Level in the Data*, BIT Numerical Analysis 49, 2009, pp. 669–696.

[41] S. Kaczmarz, *Angenäherte Auflösung von Systemen Linearer Gleichungen* (in German), Bull. Acad. Polon. Sci. Lett. A 35, 1937, pp. 355–357.

[42] M. E. Kilmer and D. P. O'Leary, *Choosing Regularization Parameters in Iterative Methods for Ill-Posed Problems*, SIAM J. Matrix Anal. Appl., Vol.22, No.4, 2001, pp. 1204–1221.

[43] A. Kirsch, *An Introduction to the Mathematical Theory of Inverse Problems*, Springer-Verlag, New-York, 1996.

[44] R. Kress, *Linear Integral Equations*, Springer-Verlag, Berlin, 1989.

[45] V. I. Krylov, *Approximate Computation of Integrals* (in Russian), 2nd issue, Moscow, 1967.

[46] C. Lanczos, *Linear Differential Operators.*, Van Nostrand, London, 1961.

[47] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators.*, J. Res. Nat. Bur. Stand. 45, 1950, pp. 255–282.

[48] J. L. Lennon, *Red-shifts and red herrings in geographical ecology*, Ecography, Vol. 23, 2000, pp. 101–113.

[49] G. Meurant and Z. Strakoš, *The Lanczos and Conjugate Gradient Algorithms in Finite Precision Arithmetics*, Acta Numerica 12, Cambridge University Press, 2006, pp. 471–542.

[50] M. Michenková *Regularization Techniques Based on the Least Squares Method*, Diploma Thesis, Prague, 2013.

[51] S. G. Michlin, *Variational Methods in Mathematical Physics* (in Russian), Moscow, 1957.

[52] V. A. Morozov, *On the Solution of Functional Equations by the Method of Regularization*, Soviet Math. Dokl. 7, 1966, pp. 414–417.

[53] V. A. Morozov, *Methods for Solving Incorrectly Posed Problems*, Springer-Verlag, New York, 1984.

[54] F. Natterer, *The Mathematics of Computerized Tomography*, Weley, New York, 1986.

[55] S. M. Nikolskiy, *Quadrature Formulas* (in Russian), 2nd issue, Moscow, 1974.

[56] C. C. Paige and M. A. Saunders, *Algorithm 583: Sparse Linear Equations and Least Square Problems*, ACM Trans. Math. Software 8, 1982, pp. 195–209.

[57] C. C. Paige and M. A. Saunders, *LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares*, ACM Trans. Math. Soft. 8, 1982, pp. 43–71.

[58] D. L. Phillips, *A Technique for the Numerical Solution of Certain Integral Equations of the First Kind*, J. Soc. Comput. Mach. 9, 1962, pp. 84–97.

[59] Z. Prášková, *Fundamentals of Random Processes II* (in Czech), Prague, Karolinum, 2007.

[60] S. Shakhno, *Numerical Methods in Linear Algebra* (in Ukrainian), Lviv, Publishing Center of Ivan Franko National University of Lviv, 2007.

[61] L. Simcik and P. Linz, *Qualitative Regularization: Resolving Non-Smooth Solutions*, Report CSE-94-12, Dept. of Computer Science, University of California, Davis, CA, 1994.

[62] R. C. Smith, K. L. Bowers and C. R. Vogel, *Numerical Recovery of Material Parameters in Euler-Bernoulli Beam Models*, ICASE Report 91-14, NASA Langley Research Center, February 1991; J. Math. Systems, Estimation, and Control, to appear.

[63] Z. Strakoš, *Model Reduction Using the Vorobyev Moment Problem*, Numerical Algorithms 51, 2009, pp. 363–376.

[64] J. D. Tebbens, I. Hnětynková, M. Plešinger, Z. Strakoš and P. Tichý, *Analysis of Methods for Matrix Computations. Basic Methods.* (in Czech), Prague, 2012.

[65] A. N. Tikhonov, *Solution of Incorrectly Formulated Problems and the Regularization Method*, Soviet Math. Dokl. 4, 1963, pp. 1035–1038; English translation of Dokl. Akad. Nauk. SSSR 151, 1963, pp. 501–504.

[66] K. Tothová, *Numerical Methods of Image Processing*, Bachelor Thesis, Prague, 2010.

[67] S. Twomey, *On the Numerical Solution of Fredholm Integral Equations of the First Kind by Inversion of the Linear System Produced by Quadrature*, J. Assoc. Comput. Mach. 10, 1963, pp. 97-101.

[68] J. M. Varah, *On the Numerical Solution of Ill-Conditioned Linear Systems with Applications to Ill-Posed Problems*, SIAM J. Numer. Anal. 10, 1973, pp. 257–267.

[69] K. Vasilík, *Linear Error-In-Variable Modeling*, Diploma Thesis, Prague, 2011.

[70] M. M. Vainberg, *Variational Method and Method of Monotone Operators in the Theory of Nonlinear Equations* (in Russian), Moscow, 1972.

[71] G. Wahba *Practical Approximate Solutions to Linear Operator Equations When the Data are Noisy*, SIAM J. Numer. Anal. 14, 1977, pp. 651–667.

Appendix

In this appendix description of the main functions of the application is summarized.

main.m

```
function main()
% MAIN Entry point.
%     Main file which runs the application that iterates through
%     all combinations of problems, noise levels, GKB stopping
%     methods, regularization methods and inner regularization
%     parameter finding methods.
```

main_gui.m

```
function varargout = main_gui(varargin)
% MAIN_GUI M-file for main_gui.fig
%    Contains the entry point to the graphical user interface
%    version of the application.
```

do_regularize.m

```
function [sol_x, r] = do_regularize(with_ui, A, b, x, ...
        noise_gap, reg_met, param_finding_method, ...
        problem_name, noise_level, stopping_method, ...
        eigen_axes, reg_axes, graph_axes)
% DO_REGULARIZE Regularizes the specified approximation problem
%     Ax = b.
%
% with_ui - whether the UI-friendly program is run.
% A, b, x - the components of the problem Ax = b.
% noise_gap - number of extra iterations taken after the noise
%     level revealing one.
% reg_met - inner regularization method. Possible value:
%     - 'tsvd' - TSVD;
%     - 'Tikh' - Tikhonov.
% param_finding_method - inner regularization parameter finding
%     method. Possible values:
%     - 'gcv' - GCV;
%     - 'L-curve' - L-curve;
%     - 'discrep' - discrepancy principle.
% problem_name - name of the problem to be displayed.
% noise_level - noise level to be displayed.
% stopping_method - noise level revealing (stopping) method.
```

```
%     Possible values:
%     - cl2orig - closest-to-origin method;
%     - minpt - minimum-point method;
%     - stag - stagnation-based method.
% eigen_axes - the subgraph on the UI-friendly application with
%     eigenvalues.
% reg_axes - the subgraph on the UI-friendly application with
%     singular vector components.
% graph_axes - the subgraph on the UI-friendly application with the
%     exact naive and comuted solutions.
```

solve_regularized.m

```
function [x] = solve_regularized(S, L, W, b, method, l)
% SOLVE_REGULARIZED Regularizes the system using the specified
%     technique and solves the system.
%
% Input:
% S, L, W - the results of the GKB process, where S^T A W = L
%     (or if we take into account iteration numbers
%     S_k^T A W_k = L_k).
% b - initial vector. Typically it is the right-hand side.
% method - which inner regularization method should be used.
%     Can be 'direct' - direct LSQR method is used, no
%     regularization.
%     'tsvd' - truncated SVD or 'Tikh' - Tikhonov regularization
%     method.
% l - inner regularization parameter.
%
% Output:
% x - the regularized solution.
```

bidiag_ex.m

```
function [U, L, V, p1, k_stag] = bidiag_ex(A, s, k, p, ...
        tol, reort, refin, method, noise_gap)
% BIDIAG_EX An extended version of the Golub-Kahan iterative
%     bidiagonalization (GKB) algorithm with the noise level
%     revealing method and the noise gap (extra GKB iterations
%     after the noise revealing one).
%
% Input:
% [U, L, V, p1] = bidiag_ex(A, s) or L = bidiag_gk(A, s) -
%     mandatory inputs, it computes either all three factors or
%     only the bidiagonal matrix
% [U, L, V, p1] = bidiag_ex(A, s, k) - number of iterations
% [U, L, V, p1] = bidiag_ex(A, s, k, p) - square/rectangular L
```

```
% [U, L, V, p1] = bidiag_ex(A, s, k, p, tol) - warning threshold
% [U, L, V, p1] = bidiag_ex(A, s, k, p, tol, reort) -
%     reorthogonalization
% [U, L, V, p1] = bidiag_ex(A, s, k, p, tol, reort, refin) -
%     iterative refinement
% [U, L, V, p1] = bidiag_ex(A, s, k, p, tol, reort, refin, method) -
%     noise level revealing (stopping) method.
% [U, L, V, p1] = bidiag_ex(A, s, k, p, tol, reort, refin, method, -
%     noise_gap) - noise gap - the number of extra iterations taken
%     after the noise level revealing one.
%
% A - matrix,
% s - starting nonzero vector (e.g. the right-hand side in Ax=b),
% k - number of iterations of the bidiagonalization process
%     (each iteration consists of the left and right half-step),
% p - proces stops with square or rectangular bidiagonal matrix:
%     if p == 0 or p == false (or p not present), then
%
%     A^T * U_k = V_k * L_k^T,    where U_k = U(:,1:k),
%     V_k = V(:,1:k) and
%
%          L_k = / a_1          0  \
%                | b_2 a_2         |
%                |     ... ...     |
%                \  0      b_k a_k /
%
%     is square lower bidiagonal matrix, if p == 1 or p == true or
%     p > 1, then
%
%      A * V_k = U_{k+1} * L_{k+},    where
%
%       L_{k+} = / a_1          0      \ = / L_k             \
%                | b_2 a_2             |   \ b_{k+1} * e_k^T /
%                |     ... ...         |
%                |         b_k a_k     |
%                \  0          b_{k+1} /
%
%     is rectangular lower bidiagonal matrix,
%
% tol - threshold/tolerance, default value is 2^-52 = 2.2204*10^-16
%     (if a_j < tol or b_j < tol for some j, then algorithm warns).
% reort - reorthogonalization
%     ==  0         without reorthogononalization,
%     ==  r > 0     reorthogonalization against last r vectors,
%     == -1         full reorthogonalization (against all vectors,
%     default),
% refin - iterative refinement (multiple reorthogonalization),
%     ==  0         without reortogonalization,
```

```
%     ==  1         only one reorthogonalization, MGS-like,
%     ==  2         double reorthogonalization, IMGS-like (default),
%     ==  t         t-times reorth
% (reort and refin default values are recommended.)
%
% Output:
% L - the bidiagonal resulting matrix.
% U, V - the corresponding multiplication matrices, see the
%     description of p.
%
% [Golub, Kahan: Calculating the singular values and pseudo-inverse
% of a matrix, SIAM J. Numer. Anal., Ser. B, 2 (1965), pp. 205--224]
```

add_noise.m

```
function b = add_noise(b, noise_level, beta)
% ADD_NOISE adds noise with the specified level and color to the
%     vector b.
%
% Input:
% b - the vector, where noise should be added.
% noise_level - the level of the noise to be added.
% beta - defines the spectral distribution (color) of noise.
% Spectral density S(f) = N f^beta = N f^(-alpha)
% (f is the frequency, N is normalisation coeff).
%     beta = 0 is random white noise.
%     beta = -1 is pink noise
%     beta = -2 is Brownian noise
%     beta = 1 is blue noise
%     beta = 2 is volet noise
%
% Output:
% b - the resulting vector with noise.
```

select_problem.m

```
function [A, b, x] = select_problem(problem, noise_level, param1, ...
        param2, beta)
% SELECT_PROBLEM selects the problem using the problem number and
% the level of noise.
%
% Input:
% problem - the number of the problem:
%     1 - shaw(param1)
%     2 - ilaplace(param1)
%     3 - gravity(param1)
%     4 - baart(param1)
```

```
%     5 - heat(param1, param2)
%     6 - foxgood(param1)
% noise_level - the level of noise added to b.
% param1 - the first parameter of the selected problem.
% param2 - the second parameter of the selected problem.
% beta - defines the spectral distribution (color) of noise.
% Spectral density S(f) = N f^beta = N f^(-alpha)
% (f is the frequency, N is normalisation coeff).
%     beta = 0 is random white noise.
%     beta = -1 is pink noise
%     beta = -2 is Brownian noise
%     beta = 1 is blue noise
%     beta = 2 is volet noise
%
% Output:
% A - the matrix of the problem.
% b - right-hand side of the problem (with noise).
% x - exact solution of the problem.
```