ABSTRACT

Title of dissertation: IMAGE ESTIMATION AND UNCERTAINTY QUANTIFICATION

Viktoria Taroudaki, Doctor of Philosophy, 2015

Dissertation directed by: Professor Dianne P. O'Leary
Department of Computer Science

Recorded images are usually contaminated by blur and noise. The restoration of such altered images is an ill-posed problem. Even if the blur is known, the unknown noise leads to uncertainty in the restored image. The naive restoration approach fails since it contains a lot of noise at high frequencies that destroys the computed restored image. To remedy this problem, this work focuses on the computation of the restored image by using spectral filters that give weight to components of the image that are not so contaminated by noise. We use different filtering methods such as the Truncated Tikhonov, Truncated SVD, and new methods that we created here and we seek to find a near optimal choice of the filter parameter which will give the best approximation of the original image. We define and compute the Picard Parameter when the problem satisfies the Discrete Picard Condition, and with that we estimate the noise properties. Also, we develop a new method to compute the near optimal solution by using statistical analysis which also gives us a way to estimate the error of the solution, a way to quantify uncertainty.

IMAGE ESTIMATION AND UNCERTAINTY QUANTIFICATION

by

Viktoria Taroudaki

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Dianne P. O'Leary, Chair/Advisor
Professor Radu Balan
Professor David Jacobs
Professor Kayo Ide
Professor Kenneth Elpus, Dean's Representative

Dedication

To the people who stood by my side during my PhD studies.

Acknowledgments

This work wouldn't have been completed without the help of many people. I appreciate everything they consciously and unconsciously did for me all the years I was working to increase my knowledge and improve my skills.

First, I would like to thank my advisor, Professor Dianne P. O'Leary who introduced me to the field of Image Processing and gave me the opportunity to work on this field for my dissertation under her supervision. I learned so much more from her as she was there for me at any time inside and outside the University walls.

Second, I would like to thank Professor Radu Balan, Professor Kayo Ide, Professor David Jacobs, and Professor Kenneth Elpus, for being members of my Dissertation Committee, for all the scheduling modifications they needed to make for the defense, and for their very good suggestions and helpful comments on my work. Also, I should add here Professor Eugenia Kalnay for being a member of my Preliminary Examination Committee and for giving me advice for improvement.

Next, I would like to thank Professor Konstantina Trivisa, the director of the AMSC Program. Her help and support since the very first day made me feel I was in the right place to complete my PhD.

Ms. Alverda McCoy, the AMSC Program coordinator, was always there whenever I needed any help with the paperwork for the smooth completion of my studies and she gave a different kind of support than the faculty or the students.

I would also like to thank my extraordinary colleagues at the AMSC, MATH, and CS programs who were sharing with me the difficulties of graduate life inside

and outside school. Knowing that I was not alone in this journey gave me motivation to continue and pursue the doctorate degree. I can only hope that I played some role in their success. Thank you to Joshua Ballew, Patrick Carlos, Brianna Cash, Marie Chau, Alex Cloninger, Stefan Doboszczak, Tim Doster, Tyler Drombowsky, Virginia Forstall Amanda Galante, Ariel Hafftka, Summer Hu, Madhura Joglekar, Anne Jorstad, Mark Lai, Ioannis Markou, Franck Ndjakou Njeunje, Dimitris Ntogkas, Enrique Otarola Pasten, Maziar Raissi, Christiana Sabett, Abhishek Sharma, Lucia Simonelli, Michelle Sobel, Christoforos Somarakis, Alexey Stepanov, Ignacio Tomas, Hana Ueda, Karamatou Yacoubou Djima, and Arseny Zakharov.

Together with the above fellow students, my friends from the Summer School at the FIELDS Institute, Toronto Canada in July 2012, and the Gene Golub SIAM Summer School, Shanghai, China in July 2013 were there to support me during conferences and not only. They are too many to mention but they know who they are and I thank them all!

I would also like to thank my friends outside the department. First of all, a huge thanks to my roommates Irene Kyza, Nisha Kurian, Alec Armstrong, Kara Johnson, Michael Di Salvo, Lejla Sarcevic, Stephanie Young, Dana Botesteanu, Wenjin Chung, Abigail Lim, and Tong Wu, who tolerated me during the most difficult years of my graduate studies and created a friendly environment for me to work in. They took care of me when I needed it and they appreciated it when I did the same. They made my life easier. Thanks to George Zoto, George Zaki, Alma Jean Zaki, Jonathan Jesurai, Khurram Shahzad, Laura Maratou Kolias, Kleoniki Vlachou, Thodoris Rekatsinas, Evripidis Paraskevas, Konstantinos Zampogiannis,

Elena Stai, Dimitris Spyropoulos, and Jonathan Braxton with whom I shared a lot of good moments that made me relax in between stressful times, and gave me strength to stay in the right track for me.

Nothing happens without faith in God, and so, I would like to thank Father Nick and Presvytera Kostoula, Father Jason and Presvytera Alexandra, Father Michael, Joe Mayes, Chrissy Mayes, Aleko Tiches, Kathy Matrakas, Ted Matrakas, Kristina-Maria Paspalis, Eleni Paspalis, Kostas Paspalis, John Alexiou, Nina Alexiou, Patrick O'Donnel, Alexia Christian, Elias Mavromatis, Maria Poulakis, Maria and Dr. Theodore Papaloizos and all the Parishioners of Saints Constantine and Helen of Washington DC.

On the friends list, I shouldn't forget my old friends from Greece and around the world, Dimitra Trobouki, Argyro Xanthaki, Dimitra Xanthaki, Theoni Agathou, Nikos Pattakos, Laurel Kanawayer, Ines Brahim, Guillaume Chappel, Sophia Tudin Karina, Maria Saitaki, Maria Christofi, George Moutsatsos, George Tzanakis, Nikos Pattakos, Manos Kamarianakis, Nikos Mitsakos, George Papageorgiou, Vasilis Noutsis, and Dimitris Antonopoulos. Even though they are far and we don't communicate much, I know that they are always there for me.

In addition, I would like to thank my teachers, mentors, and Professors at the Universities of Crete, Athens and Maryland. Especially, Mrs. Brotzaki, Mr. Orfanoudakis, Mr. Armaos, the Hellenic Mathematical Society, Prof. Dougalis, Prof. Makridakis, Prof. Chatzipantelidis, Prof. Garefalakis, Prof. Skandalis, Prof. Stratis, Prof. Papadopoulou, Prof. Kosioris, Prof. Balan, Prof. Nochetto, Prof. Levermore, Prof. Ide, Prof. Kalnay, and Ms. Kimberly Fouche who helped and encouraged me

before and during my graduate studies. I don't know where I would have been without them.

Stephen Semick, my DRP student, helped me develop as a teacher and mentor in a different way than the students I had helped as a Teaching Assistant at the Universities of Crete, Athens and Maryland and he shouldn't be left without mentioning.

Another huge thanks goes to the School of Music and the Choral Activities Program. I spent six years at the University of Maryland singing at the choirs and had a great time preparing and performing great pieces at the University, the Kennedy Center, Strathmore, and Mayerhoff Music Halls as well as other places in Washington DC and Maryland under the direction of great conductors. Thanks to Dr. Edward Maclary, Dr. Kenneth Elpus, Tim Reno, Stephen Holmes, Scot Hanna-Weir, Cindy Bauchspies, Allan Laino, Paul Heins, Rachel Carlson, Ianthe Marini, Steven Seigart, Lauri Johnson, and the members of the University Chorale and Women's chorus who let me join them in singing and take a break from work. It helped me a lot.

My family's love, guidance and continuous support was there for me whenever I needed it. I would definitely not be where I am now if they hadn't helped me get through struggles that seemed impossible to handle. My parents (Especially my father, Michael Taroudakis, who was always my example going through this process), my sisters, my grandmothers, my aunts, uncles, and cousins all showed me how much they understood that I chose a difficult path to follow and supported me in any way they could.

Together with my family, Joshua Ballew was always there for me in the best and the worst moments during the last few years.

I would like to mention that the work presented in this thesis was partially supported by the NSF Grant DMS 1016266 and that I was also supported by the Onassis Foundation Scholarship.

Lastly, I would like to thank one of the teachers in my elementary school. I don't recall his name but I can't forget the role he played in my life. He substituted for my teacher for a few days when I was in the fifth grade. There was a question about a circle in a square with side length 1 unit. We had to find the area or the circumference of the circle. It might seem like a very simple exercise but at the time I was the only one who solved it and he congratulated me. He was the first teacher who did that for math with me. In addition to that he gave us a song so that we can remember the first few digits of π. And that was when I decided that I would like to work more on math. And I didn't change my mind since then. So thank you Mr. ...

I am sure I have left out many people that I should thank and I apologize to those I missed. There were so many people who passed by my life and left a small or big mark on me that made me who I am and do what I do. And this dissertation is a result of that. The order I mentioned people above has nothing to do with the magnitude of the impact all these people had on me. So, thank you all!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

cdf	cumulative distribution function
ContTSVD	Continuous Truncated SVD
DP	Discrepancy Principle
DPC	Discrete Picard Condition
exp_std	estimated standard deviation
GCV	Generalized Cross Validation
HS	Heaviside filter
HYBR	Tikhonov Truncated SVD Hybrid filter
Matlab	Mathworks' Matrix Laboratory software
PP	Picard Parameter
PSF	Point Spread Function
SNR	Signal-to-Noise ratio
SOF	Statistical Optimal Filtering
SVD	Singular Value Decomposition
SVE	Singular Value Expansion
TAN	Tangent Filter
Tik	Tikhonov filter
Tikk	Tikhonov filter truncated using the PP
TSCM	Truncated Singular Component Method
TSCM	Truncated Singular Component Method truncated using the PP
TSVD	Truncated Singular Value Decomposition
TSVDk	Truncated Singular Value Decomposition truncated using the PP

Chapter 1: Introduction

People have always wanted to keep snapshots of their everyday life for reference at a later time or for research and educational purposes. Cavemen drew on the walls of the caves using colors made from nature. Later artists painted their houses, graves and other buildings, objects or paintings with various scenes. More recently, cameras were invented, first engraving, then analog cameras and at last digital cameras. In none of these cases is the object represented exactly in the image. But as technology progresses, the accuracy of the representation increases. Digital cameras give us very good representations of the true image, but due to the procedure that the image passes through, blurring occurs. This blurring can be caused by the machine errors in transforming the image into data in the camera and from the background and the way of taking the picture. Having clear images is not a luxury. Sometimes it is a matter of life and death, like in surgeries where the doctor needs to know exactly where to operate, or in weather forecasts.

The images that are recorded by cameras or medical imaging devices are usually contaminated by blur and noise that come from factors such as the motion of the camera or the object, the setting, the surrounding atmosphere or turbulence. The restoration of such altered images is a challenging problem since it is ill-posed.

Even if the blur is known (e.g., due to the motion or defocus), the noise is unknown and random which can lead to multiple restored images corresponding to a given noisy blurred image. We seek to better approximate the true image by estimating the noise properties such as the mean and the standard deviation.

An image is divided into pixels that have values denoting the color of that pixel. A grayscale image, which we will use for simplicity, has one value for each pixel, an integer in the interval $[0, 255]$. 0 is the black color, and 255 is the white color. See Figure 1.1

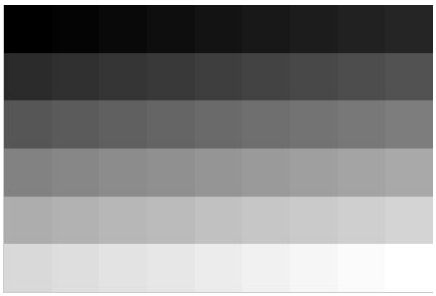

Figure 1.1: Part of the colorband of a gray-scale image, with pixel values ranging from 0 (upper left) to 255 (lower right).

Blurring occurs when the image of a pixel is affected by its neighbors. In this project, we will assume that this is caused by a linear transformation arising from the camera.

1.1 Point-Spread Function and Blurring Matrix

In general, the blurring matrix $\mathbf{A}$ can be experimentally measured using *point spread functions (PSF)* for each pixel of the original image. An easy way to do this

is by constructing an artificial image of size $m_v \times m_h$ which contains only one white pixel (of value 255) as the target pixel, say the (i, j) pixel of the image $\mathbf{X}$ (or the $(j-1) \cdot m_v + i$ element of the vector $\mathbf{x}$ formed by stacking the columns of the image) and black anywhere else (value 0). See Figure 1.2.

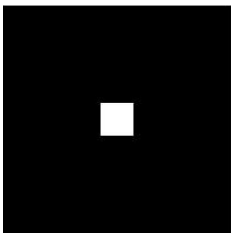

Figure 1.2: Example of an artificial image.

We consider this as the image with no blur or noise, and we blur it the same way as we would blur the original image. Then, we measure the resulting $n_v \times n_h$ blurred image $\mathbf{B}$, the point spread function. The corresponding vector $\mathbf{b}$, formed by stacking the columns of $\mathbf{B}$, is the $(j-1) \cdot m_v + i$ column of the matrix $\mathbf{A}$.

If we know that the blur is spatially invariant, then measuring only one column of the blurring matrix $\mathbf{A}$ is enough to determine the whole matrix, as the rest of the columns of $\mathbf{A}$ are simply going to be some displacement of that one column. (See Appendix A for more details.)

If the blur is spatially variant, we need to move the target point to all the pixels of the image and measure the blur to compute all the columns of the blurring matrix. For more information, someone could consult [21].

1.2 The model

We will use the notation in Table 1.1.

Table 1.1: Model Equation Symbols and Explanation

Symbol	**Size**	**Explanation**
$\mathbf{A}$	$m \times n$	Matrix defined through the point spread function (PSF)
$\mathbf{X}_{true}$		Original true image
$\mathbf{x}_{true}$	$n \times 1$	Vector containing the values corresponding to the pixels of the image $\mathbf{X}_{true}$
$\mathbf{B}$		The blurred image we measure
$\mathbf{b}$	$m \times 1$	Vector which contains the values of the pixels of the blurred image $\mathbf{B}$
$\mathbf{e}$	$m \times 1$	Noise vector

With the above notation, the discrete linear model of the blurred grayscale image is described by the equation

$$\mathbf{b} = \mathbf{A}\mathbf{x}_{true} + \mathbf{e}, \tag{1.1}$$

and we know that $0 \leq x_j \leq 255$, $j = 1, \ldots, n$.

1.3 Image Restoration

Much research has been performed on ways to restore an image (see for example [2], [7], [23]) and many different approaches and algorithms are now used to eliminate noise and blur.

My work focuses on the computation of a restored image using *spectral filters* that give weight to components of the image that are not so contaminated by noise. To do this, we estimate the properties of the noise such as the mean and the standard deviation. For each filtering method, an optimal choice of the filter parameters will give the best approximation of the original image $\mathbf{x}_{true}$.

1.4 Spectral Filters

We assume that $\mathbf{b} = \mathbf{b}_{true} + \mathbf{e}$ where $\mathbf{b}_{true} = \mathbf{A}\mathbf{x}_{true}$ is the true image, and the noise $\mathbf{e}$ consists of samples from a distribution with mean 0 and standard deviation s. The matrix $\mathbf{A}$ is $m \times n$ with $m \geq n$, full-rank (rank=n) and generally ill-conditioned, since it is a discretization of an ill-posed operator. From this point, for simplicity, we will assume that $\mathbf{A}$ is a real matrix but the generalization to complex is straight-forward.

To explore the ill-conditioning of the matrix $\mathbf{A}$, we look at its *singular value decomposition* (SVD). Let the SVD of $\mathbf{A}$ be $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The matrices $\mathbf{U}$ and $\mathbf{V}$ are orthogonal with size $m \times m$ and $n \times n$ respectively. The matrix $\mathbf{\Sigma}$ is a matrix whose main diagonal elements are the singular values of the matrix $\mathbf{A}$ in decreasing order ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$). The other entries in the matrix $\mathbf{\Sigma}$ are zero.

First consider the special case where $m = n$. Since $\mathbf{A}$ does not have zero singular values, it is nonsingular. The solution of the problem (1.1) can be found by multiplying both sides of Equation (1.1) by the inverse of the matrix $\mathbf{A}$ and taking into account that the matrices $\mathbf{U}$ and $\mathbf{V}$ in the SVD are orthogonal (i.e., $\mathbf{U}^T = \mathbf{U}^{-1}$

and $\mathbf{V}^T = \mathbf{V}^{-1}$), and the matrix $\mathbf{\Sigma}$ is diagonal. This means that

$$\begin{aligned}
\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e} \quad &\leftrightarrow \quad \mathbf{b} - \mathbf{e} = \mathbf{A}\mathbf{x} \\
&\leftrightarrow \quad \mathbf{A}^{-1}(\mathbf{b} - \mathbf{e}) = \mathbf{x} \\
&\leftrightarrow \quad \left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^{-1}(\mathbf{b} - \mathbf{e}) = \mathbf{x} \\
&\leftrightarrow \quad \mathbf{V}^T\mathbf{\Sigma}^{-1}\mathbf{U}^T(\mathbf{b} - \mathbf{e}) = \mathbf{x}.
\end{aligned}$$

So using the definition of matrix-vector multiplication a few times, the true solution to our problem is

$$\begin{aligned}
\mathbf{x}_{true} =& (\mathbf{V}^T\mathbf{\Sigma}^{-1}) \sum_{i=1}^{n} \mathbf{u}_i^T(\mathbf{b} - \mathbf{e}) \\
=& \mathbf{V}^T \sum_{i=1}^{n} \frac{\mathbf{u}_i^T(\mathbf{b} - \mathbf{e})}{\sigma_i} \\
=& \sum_{i=1}^{n} \frac{\mathbf{u}_i^T(\mathbf{b} - \mathbf{e})}{\sigma_i} \mathbf{v}_i.
\end{aligned} \tag{1.2}$$

where $\mathbf{u}_i$ is the i-th column of the matrix $\mathbf{U}$ and $\mathbf{v}_i$ is the i-th column of the matrix $\mathbf{V}$, since $\mathbf{\Sigma}$ is a diagonal matrix with singular values σ_i

Since the error vector $\mathbf{e}$ is unknown, we cannot use this to solve the problem.

One approach to estimating $\mathbf{x}_{true}$ when $m \geq n$ is to minimize the norm of the residual $\mathbf{A}\mathbf{x} - \mathbf{b}$. We use the notation $\|.\|$ to denote the 2-norm of a vector. Then

$$\begin{aligned}
\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 &= \min_{\mathbf{x}} \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{x} - \mathbf{b}\| \\
&= \min_{\mathbf{x}} \|\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{x} - \mathbf{U}^T\mathbf{b}\| \\
&= \min_{\mathbf{x}} \|\mathbf{\Sigma}\mathbf{V}^T\mathbf{x} - \boldsymbol{\beta}\| \\
&= \min_{\mathbf{x}} \|\mathbf{\Sigma}\mathbf{z} - \boldsymbol{\beta}\|,
\end{aligned} \tag{1.3}$$

where $\boldsymbol{\beta} \equiv \mathbf{U}^T\mathbf{b}$ or ($\beta_i \equiv \mathbf{u}_i^T\mathbf{b}$) and $\mathbf{z} = \mathbf{V}^T\mathbf{x}$ and since the matrix $\mathbf{U}$ is orthogonal, it satisfies $\|\mathbf{U}^T\mathbf{y}\| = \|\mathbf{y}\|$, for every vector $\mathbf{y}$.

The solution of problem (1.3) is

$$\mathbf{x}_{naive} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i. \tag{1.4}$$

This solution is naive since for small singular values, the fraction becomes huge and may surpass the numbers that a computer can handle. Clearly this is not the desired solution. For this reason we apply a *spectral filter* ϕ_λ, as

$$\mathbf{x}_{filt} = \sum_{i=1}^{n} \phi_\lambda(\sigma_i)\frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i. \tag{1.5}$$

In all of these cases, though, the solution is just an estimate of the original image since the noise is unknown.

We assume that the problem satisfies the discrete Picard condition [15], which means that there is a parameter k such that $\mathbf{u}_i^T\mathbf{b} \approx \mathbf{u}_i^T\mathbf{e}$ for $i \geq k$,

The filter is determined by one or more parameters λ. Examples of such filters are the *truncated SVD (TSVD)* filter [10],

$$\phi_i = \phi_\lambda(\sigma_i) = \begin{cases} 1 & \text{if } \sigma_i \geq \sigma_\lambda, \\ 0 & \text{otherwise,} \end{cases} \tag{1.6}$$

the *Tikhonov* Filter [22]

$$\phi_i = \phi_\lambda(\sigma_i) = \frac{\sigma_i^2}{\sigma_i^2 + \lambda}, \tag{1.7}$$

and the TSCM (*Truncated Singular Component Method*) filter [24]

$$\phi_i = \begin{cases} 1 & \text{if } |\mathbf{u}_i^T\mathbf{b}| > \lambda_2 \text{ and } i < \lambda_1, \\ 0 & \text{otherwise.} \end{cases} \tag{1.8}$$

Notice that all of these filters reduce the contributions for which the Picard condition predicts that the data is unreliable.

1.5 Generalized Cross Validation and Discrepancy Principle

Our goal is to determine the parameter $\boldsymbol{\lambda}$ for the filter so that we obtain a good solution. Here we discuss two popular methods for determining the parameters.

Generalized Cross-Validation (GCV). GCV [11] determines the parameter $\boldsymbol{\lambda}$ so that if we leave one observation b_i out of the computation, it is best predicted by the parameters chosen. It has been shown that this means that we minimize the GCV function

$$G(\lambda) = \frac{\|(\mathbf{I} - \mathbf{A}\mathbf{V}\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T)b\|_2^2}{\text{trace}(\mathbf{I} - \mathbf{A}\mathbf{V}\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\mathbf{U}^T))^2}, \tag{1.9}$$

where $\boldsymbol{\Phi}$ is the diagonal matrix of the filter factors ϕ_i. In the case of the TSVD filter, this general form of the GCV function can take the more specific form:

$$G(\lambda) = \frac{\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|_2^2}{(n-k)^2}. \tag{1.10}$$

The GCV method makes no assumptions about the error.

The Discrepancy Principle (DP). The Discrepancy Principle [19] computes the parameter $\boldsymbol{\lambda}$ for which the norm of the residual approximates the expected norm of the noise ($\delta = \mathcal{E}(\|\mathbf{e}\|_2)$):

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_{filt}\|_2 = \tau\delta, \tag{1.11}$$

where τ is a factor commonly set to $\tau \in \{2, 3, 4, 5\}$. In our work, we used the parameter $\tau = 2$. This method relies on having a good estimate of the expected norm of the noise.

1.6 Our contributions

Chapter 2. Assuming the Picard condition holds and that the error is Gaussian, we propose a method for estimating the properties of the noise, i.e., its mean and its standard deviation.

For this, we define the Picard Parameter (PP), the index beyond which the observed data, in the coordinate system of the SVD, are overwhelmed by error. Automatic estimating of this important Picard Parameter is developed (Section 2.4) and is presented in addition to its manual estimation (Section 2.3).

The Picard Parameter is helpful because it gives an estimate of when the blurred image's components are contaminated by noise. We can thus discard those from our computations to find the solution.

Chapter 3. Ideally, instead of solving (1.3), we want to determine a filter $\phi_{\boldsymbol{\lambda}}$ to minimize the norm of the error:

$$\min_{\boldsymbol{\phi}} \|\mathbf{x}_{filt} - \mathbf{x}_{true}\|. \tag{1.12}$$

This is not possible, since $\mathbf{x}_{true}$ is not known.

Using statistical analysis, D.P. O'Leary estimated the near-optimal parameter for Tikhonov filtering [22]. Here, we extend this approach and determine the near-

optimal parameter for the TSVD filter.

Chapter 4. Our approach is quite general, and next we show how near-optimal parameters can be determined for arbitrary filters, including known filters such as TSCM. We call the method SOF, for Statistically Optimal Filter. We also discuss how to quantify the uncertainty in the computed solution.

Chapter 5. We propose several new filters: a slight modification of the Tikhonov Method, the TIKk (*Truncated Tikhonov*)

$$\phi_i = \begin{cases} 0 & \text{if } i \geq \lambda_2, \\ \frac{\sigma_i^2}{\sigma_i^2+\lambda_1} & \text{otherwise,} \end{cases} \tag{1.13}$$

a new hybrid filter (HYBR)

$$\phi_i = \begin{cases} 1 & \text{if } i \leq \lambda_1, \\ \frac{\sigma_i^2}{\sigma_i^2+\lambda_2} & \text{if } \lambda_1 < i \leq \lambda_3 - 1, \\ 0 & \text{if } i \geq \lambda_3, \end{cases} \tag{1.14}$$

a continuous version of the TSVD (ContTSVD)

$$\phi_\lambda(y) = \begin{cases} 1, & \text{if } y \geq \lambda \\ \frac{y-\sigma_{i+1}}{\lambda-\sigma_{i+1}}, & \text{if } \sigma_{i+1} \leq y < \lambda \\ 0, & \text{if } y < \sigma_{i+1}, \end{cases} \tag{1.15}$$

some Heaviside filters, and a cubic spline filter with any number of knots.

Chapter 6. We evaluate our new filters and our new SOF method for determining near-optimal filter parameters by testing against state-of-the-art algorithms such as the TSVD and Tikhonov filters and use of the Discrepancy Principle and GCV for determining filter parameters.

Chapter 7. In summary, the major contributions of this work are

- Definition of the Picard parameter and development of algorithms for determining it.
- Development of an algorithm for determining near-optimal filter parameters for any spectral filtering method.
- Uncertainty quantification through an estimate for the expected error in any spectral filtering method.
- Development of several new spectral filters. Some of them are modifications of already known ones like the Truncated Tikhonov filter, combinations of two filters like the Hybrid filter, or continuous versions of the discrete ones (ContTSVD). Some of the filters we present in this work are brand new though, like the Heaviside and Tangent filters or the spline filter either with linear or logarithmic spacing of the knots.

Chapter 2: Picard Parameter Estimation

In this chapter we define the Picard Parameter and discuss its estimation that will be necessary for the following chapters. To make it easier for the reader to follow, we define notation that will be used in the rest of the thesis. Initially, in Section 2.1, we review the continuous Picard condition based on the Fredholm integral equation and in Section 2.2 the discrete Picard condition that will be used in Chapters 3 and 4. We define the Picard parameter and in Section 2.3 we show how it is estimated manually. In Section 2.4 we present two ways of automatically estimating the Picard parameter, using histograms (2.4.1) or the Lilliefors test (2.4.2). Numerical results and conclusions are given at the end of the chapter (Sections 2.5 and 2.6).

2.1 A Fredholm integral equation

Following the presentation of [20] and [13], we consider the Fredholm equation of the first kind

$$\int_{\Omega} K(s,t)f(t)dt = g(s), \tag{2.1}$$

with known functions K and g, and unknown function f. For photographic image deblurring problems, $\Omega \subset \mathbb{R}^2$ is the domain of the image, and $f : \Omega \to \mathbb{R}$. According to the *Singular Value Expansion (SVE)*, a kernel K for which $\|K\|_{L_2} < \infty$ (i.e., K

is square integrable) can be written as

$$K(s,t) = \sum_{i=1}^{\infty} \mu_i u_i(s) v_i(t), \tag{2.2}$$

where u_i and v_i are the left and right singular functions of K and μ_i are the singular values of K, $\mu_1 \geq \mu_2 \geq \cdots \geq 0$. The singular functions u_i and v_i are orthonormal and the singular values are in nonincreasing order. Because K is square integrable, $\sum_{i=1}^{\infty} \mu_i^2 < \infty$. According to the *Singular Value Expansion* theorem

$$\int_{\Omega} K(s,t) v_i(t) dt = \mu_i u_i(s),\ i = 1, 2, \cdots . \tag{2.3}$$

Multiplying (2.1) by $u_i(s)$ and integrating with respect to s we obtain

$$\int_{\Omega} u_i(s) \int_{\Omega} K(s,t) f(t) dt ds = \int_{\Omega} u_i(s) g(s) ds.$$

Using (2.2), we see that

$$\int_{\Omega} \int_{\Omega} u_i(s) \sum_{j=1}^{\infty} \mu_j u_j(s) v_j(t) f(t) dt ds = \int_{\Omega} u_i(s) g(s) ds.$$

The functions u_i are orthonormal, so

$$\begin{aligned} \int_{\Omega} u_i(s) u_i(s) ds &= 1, \\ \int_{\Omega} u_i(s) u_j(s) ds &= 0, \quad i \neq j, \end{aligned}$$

and therefore

$$\int_{\Omega} \mu_i v_i(t) f(t) dt = \int_{\Omega} u_i(s) g(s) ds. \tag{2.4}$$

Defining the *inner product* of two functions f and g defined on Ω as

$$\langle f, g \rangle \equiv \int_{\Omega} f(t) g(t) dt, \tag{2.5}$$

equation (2.4) leads to

$$\langle v_i, f\rangle = \frac{1}{\mu_i}\langle u_i, g\rangle. \tag{2.6}$$

Using that, we see that

$$f(t) = \sum_{i=1}^{\infty}\langle v_i, f\rangle v_i(t) = \sum_{i=1}^{\infty}\frac{\langle u_i, g\rangle}{\mu_i}v_i(t). \tag{2.7}$$

The *Picard Condition* [20] for problem (2.1) is

$$\sum_{i=1}^{\infty}\left(\frac{\langle u_i, g\rangle}{\mu_i}\right)^2 < \infty. \tag{2.8}$$

If the Picard condition is satisfied, then f is the solution to (2.1). The Picard condition means that the sequence of inner products $\{\langle u_i, g\rangle\}$ decays faster than the singular values, $\{\mu_i\}$. In this case the problem is well defined and there is a solution. The problem is that when g includes noise, the Picard condition is usually violated. Even if the true right-hand side satisfies the Picard condition, the noise is random and in general does not satisfy it. Thus, the sum does not satisfy the Picard condition. If that is the case, then there is the possibility that the sequence of inner products $\{\langle u_i, g\rangle\}$ does not decay faster than the sequence of singular values of K, $\{\mu_i\}$, and then there is no solution. Also that means that for small singular values, the error dominates, i.e., $\langle u_i, g\rangle \approx \langle u_i, e\rangle$ if μ_i is small. This observation will be used later on in the computation of the Picard parameter.

2.2 The discrete problem

Now, let's see how the previous Fredholm integral equation model relates to the model of the image deblurring problem as described in Chapter 1.

A recorded image is a discretized version of the original image that can be described as a function $f : \Omega \to \mathbb{R}$. The camera will only record a rectangular part of this image. This rectangular image can be discretized using pixels. Each pixel will have one value. We can simplify the problem by stacking the pixels of the image in a vector. For example if the image is $n_1 \times n_2$, then the pixel that is located in the d_1 row and d_2 column will have a new coordinate of $i = (d_2 - 1) \cdot n_1 + d_1$. The blurring matrix in the discretized case comes from values of $K(s,t)$ multiplied by some weights that come from the approximation of the integral by a quadrature method as described below.

One idea (see [20]) is to discretize (2.1) by using a quadrature method at n points $\{t_j\}_{j=1}^n \subset \Omega$, so that

$$g(s) = \int_\Omega K(s,t)f(t)dt \approx \sum_{j=1}^{n} w_j K(s,t_j)f(t_j). \tag{2.9}$$

Using m discrete points $\{s_i\}_{i=1}^m \subset \Omega$ we have m linear equations. If we define the matrix $\mathbf{A}$ to have the elements $a_{ij} \equiv w_j K(s_i,t_j)$ and define the vectors $\mathbf{b}$ and $\mathbf{x}$ with elements $b_i \equiv g(s_i)$ and $x_j \equiv f(t_j)$, then the system of m equations becomes $\mathbf{b} = \mathbf{A}\mathbf{x}$. For the rest of the discussion we will assume for simplicity that $n = m$.

We consider the case where there is added noise and so

$$\mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{b}. \tag{2.10}$$

Let $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the singular value decomposition of the matrix $\mathbf{A}$. The matrices $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and include as columns the singular vectors of the matrix $\mathbf{A}$, and $\mathbf{\Sigma}$ is a diagonal matrix whose elements are called the singular values

of $\mathbf{A}$, denoted by $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. In the noise-free case, (2.10) takes the form

$$\mathbf{A}\mathbf{x}_{true} = \mathbf{b}_{true}. \tag{2.11}$$

The vector $\mathbf{b}_{true}$ in (2.11) satisfies the *discrete Picard condition (DPC)* if the corresponding coefficients $|(\boldsymbol{\beta}_{true})_i| = |\mathbf{u}_i^T \mathbf{b}_{true}|$ decay on average faster than the σ_i (see [13]). The added noise does not necessarily abide by this condition, and the above statement is not true for (2.10). There is an index k after which the coefficients $|\mathbf{u}_i^T \mathbf{b}_{true}|$ stop decaying and become close to $|\boldsymbol{\epsilon}_i| = |\mathbf{u}_i^T \mathbf{e}|$. For a fixed length of the unknown vector n, define k to be the index for which $\beta_i \approx \epsilon_i$ for $i \geq k$.

This parameter k is important since it signifies the point after which $\boldsymbol{\beta} = \mathbf{U}^T \mathbf{b}$ is dominated by noise. If we discard the components $k, \cdots, n$, we don't lose important information and we reduce the computational cost.

We name this parameter k the *Picard Parameter*.

2.3 Manual estimation of the Picard parameter

The Picard parameter can be estimated graphically. The *Picard plots* show the natural logarithm of the $|\beta_i|$ and the singular values σ_i with respect to the index i. According to the definition of the discrete Picard condition, on average the right-hand side values $|\beta_i|$ should decay faster than the corresponding singular values σ_i.

Figure 2.1 shows the Picard plot of the upper left 64×64 part of the Barbara image blurred by separable Gaussian blur (see Appendix A) with no added noise.

In this noise free case, the problem satisfies the DPC. We can see that the elements of $\boldsymbol{\beta}$ decay faster than the singular values as expected.

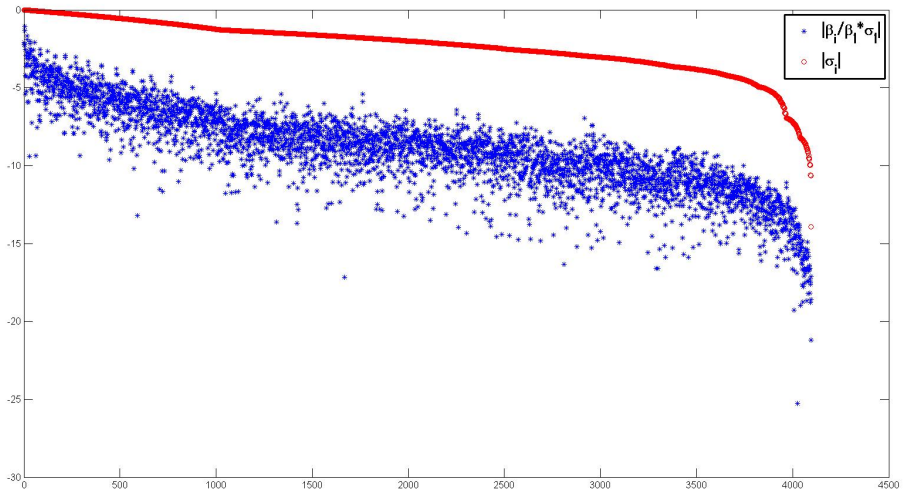

Figure 2.1: Picard plot of the true blurred image with no noise added.

Figures 2.2 and 2.3 show the Picard plots after including noise of standard deviation $s = 1$ and $s = 10$, respectively.

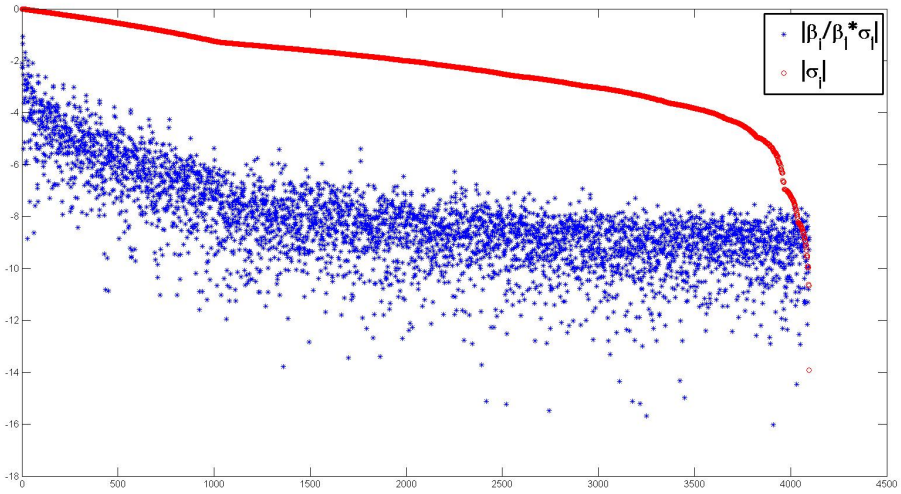

Figure 2.2: Picard plot of the known values of the blurred image with added noise with standard deviation $s = 1$.

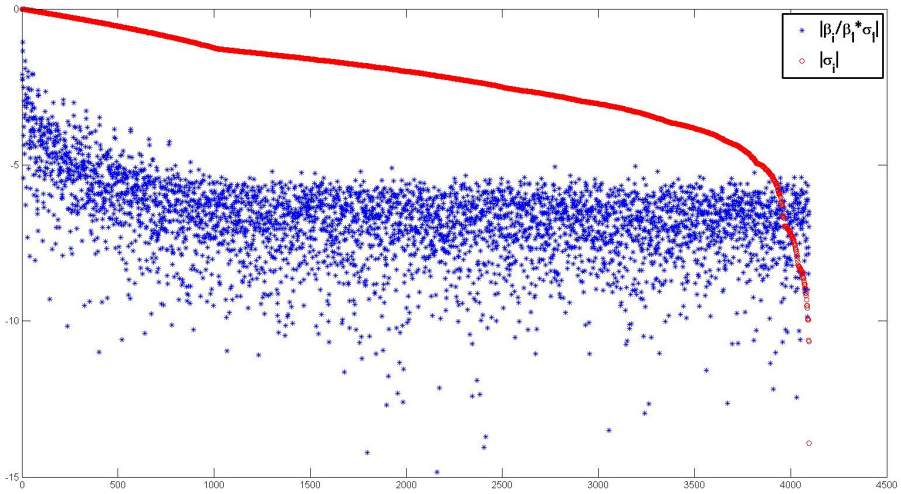

Figure 2.3: Picard plot of the known values of the blurred image with added noise with standard deviation $s = 10$.

The behavior of the components of $\boldsymbol{\beta}$ in the last two plots is different from

that of Figure 2.1. We see that $\boldsymbol{\beta}$ levels off at a certain point due to the additive noise that does not satisfy the Picard Condition.

The Picard parameter k is the index for which the β_i values do not decay as fast as the singular values due to the noise. That will mean that the β_i values mostly contain error.

When $s = 1$, the values β_i start behaving like the noise after an index slightly larger than 1000. So we could choose to keep a few more than 1000 singular components for this problem. For $s = 10$ the noise dominates faster and we don't need to keep as many as 1000 singular components. In fact we need approximately 800. This number, the Picard parameter k, is chosen as the index where on average, the singular values keep decreasing whereas the β_i values behave like noise. When there is no such behavior, we need to keep all of the singular values and $k > n$.

2.4 Automatic estimation of the Picard parameter when the noise is Gaussian

The manual estimation of the Picard Parameter in Section 2.3 is determined by the user by eyesight and is not necessary the same for every user. It also requires that the computation be paused so that the user can determine the value. In this section, we seek a method that will work faster and be independent of the user.

We make the assumption that the error that is added to the image is normally distributed. This assumption will help in determining a simple algorithm for the estimation of the Picard parameter. Other types of error would require a different

testing method for determining where the vector $\boldsymbol{\beta}$ behaves like $\boldsymbol{\epsilon} = \mathbf{u}_i^T \mathbf{e}$.

So, again, the Picard parameter k is the index after which the β_i are close to the error vector elements, ϵ_i: $\beta_i \approx \epsilon_i$, for $i \geq k$. We assume that the error vector is sampled from a normal distribution with unknown mean and standard deviation. Our goal is to determine a value k beyond which the β_i are plausible samples from a normal distribution. This procedure is called *normality testing* and can be approached in different ways.

2.4.1 Normality testing by histogram

The easiest way is to construct the histogram of the values β_i, choosing bins so that the distribution doesn't seem too coarse or too fine. If the distribution of the sample seems to have a bell shape, then it is plausible that the sample comes from a normal distribution with mean and standard deviation close to the ones estimated from the sample [6].

In general, the larger the sample is, the better the estimates will be. For example, in Figure 2.4 the histogram of 10000 samples resembles a normal distribution with mean 0 whereas the second histogram that uses only 10 samples from a normal distribution does not resemble a bell-curve. It looks similar to the third histogram in the figure which is of 10 samples from a uniform distribution. Again, increasing the number of samples to 10000, we get a better understanding of the distribution.

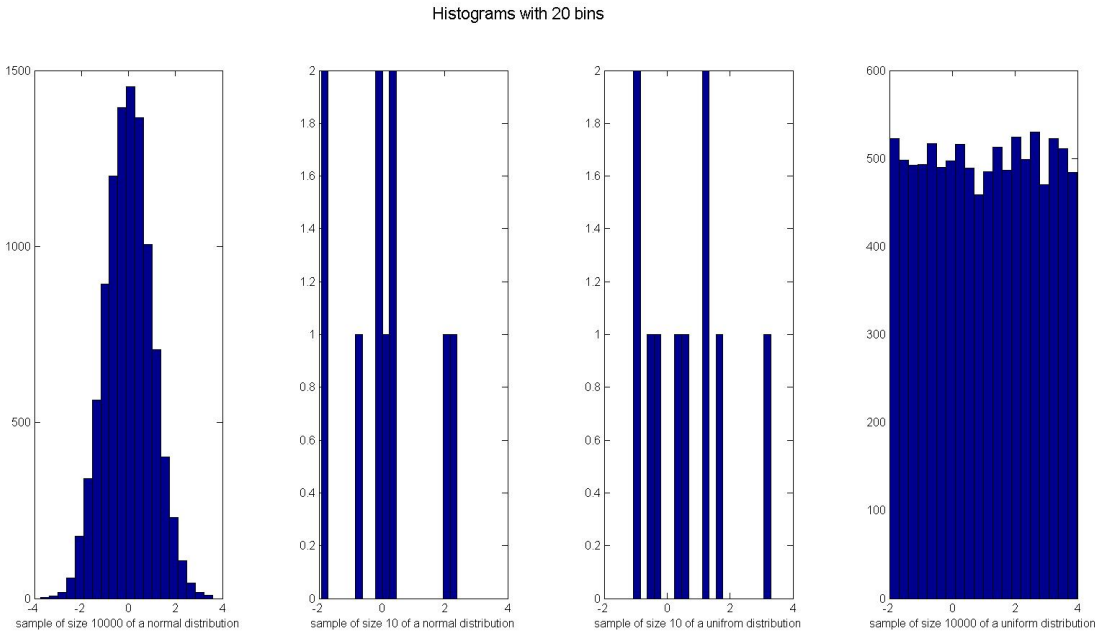

Figure 2.4: Histograms of samples of the normal distribution with size 10000 and 10, and the uniform distribution with size 10 and 10000.

2.4.2 The Lilliefors test

Other methods used for normality testing examine whether a null hypothesis is valid or not. These tests compare a sample with specific mean and standard deviation to a normal distribution with the same parameters. These methods include the D'Agostino's K-squared test [5], the Jarque-Bera test [16], the Lilliefors test [17], the Kolmogorov-Smirnov test [18], and others [6].

For our purposes, we will use the Lilliefors test which is an adaptation of the Kolmogorov-Smirnov test.

The Lilliefors test is performed on a sample of points in three major steps.

1. We estimate the population mean and the population variance using the sample mean and variance.

2. We compute the maximum discrepancy between the empirical distribution function of the sample and the cumulative distribution function of the normal distribution with the estimated mean and variance.

3. We assess whether the maximum discrepancy is large enough to be statistically significant.

More specifically, the *empirical distribution function* F_n for n independent and identically distributed observations X_i is defined as

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{X_i \leq x}, \tag{2.12}$$

where

$$I_{X_i \leq x}(x) = \begin{cases} 1 & \text{if } X_i \leq x, \\ 0 & \text{otherwise.} \end{cases}$$

Since the Lilliefors method is a variation of the Kolmogorov-Smirnov method, we use a statistic to determine whether the sample is taken from a normal distribution with no specified mean and standard deviation. For that, and for a given *cumulative distribution function* (cdf)

$$F(x) = P(X \leq x), \tag{2.13}$$

where $P(E)$ denotes the probability of the event E, we define the *maximum discrepancy* as

$$D_n = \sup_x |F_n(x) - F(x)|. \tag{2.14}$$

In the case of the standard normal distribution, the cdf is the integral

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{t^2}{2}} dt, \tag{2.15}$$

whereas for a general normal distribution with mean μ and standard deviation s, the cdf becomes

$$F(x) = \Phi\left(\frac{x-\mu}{s}\right). \tag{2.16}$$

According to the Glivenko-Cantelli theorem [1, 9], if the sample that we examine comes from a distribution $F(x)$, then $F_n(x)$ will uniformly converge to $F(x)$ almost surely. That is

$$D_n = \|F_n - F\| = \sup_x |F_n(x) - F(x)| \to 0 \tag{2.17}$$

almost surely. In other words, $P(\lim_{n\to\infty} D_n = 0) = 1$.

For the Kolmogorov-Smirnov test and therefore the Lilliefors test, the samples are standardized (i.e., they are manipulated so that they have mean 0 and standard deviation 1) and compared to the standard normal distribution. We define the *null hypothesis* for this to be "the standardized sample is taken from the standard normal distribution". If D_n is less than a desired tolerance, then the null hypothesis that our sample comes from a normal distribution with mean and standard deviation the experimental ones is true with some probability. The smaller the tolerance, the larger the probability is.

2.4.3 Using Lilliefors to estimate the Picard parameter

Based on the above, we want to find the point where the coefficients β_i start behaving like noise by testing whether the sample of β_i comes from a normal distri-

bution.

The Picard parameter is k when the last $n - k + 1$ values of β_i come from a normal distribution but adding more values destroys normality.

We estimate k using the Lilliefors test. Initially, we check a small sample of at least the last four values. While the null hypothesis is satisfied, i.e., the sample comes from a normal distribution with some confidence that we require, then we add the next value and check again. If the sample does not come from a normal distribution with that confidence, then we need to make sure this was not a random failure due to the specific sample we tested. For this reason, we continue checking until the null hypothesis fails for a specific number n_f of consecutive samples. If the last value that we used is β_i, then $k = i + n_f$.

The larger the sample is, the larger the number of elements to be added has to be to actually see some significant difference in the behaviour of a sample and say with confidence that we have to stop. That means that once we see a sample that fails the Lilliefors test, especially for a large sample, we can stop. From our experience and for the sizes of the images we use that go up to 256×256, we set $n_f = 10$. For larger images or for very low rank matrices, it might be necessary to impose a length limit on the sequence used for the Lilliefors test so that the non-normal samples are noticeable, testing elements k through $k + \hat{n}$ when $k + \hat{n} < n$.

Since the definition of the Picard parameter can be interpreted as the index after which the distribution of the right-hand side resembles the distribution of the noise, the statistical properties of the noise can be estimated from these values. So, using the Matlab commands

```
exp_mean=mean(beta(k:n));

exp_stdev=std(beta(k:n));
```

we can estimate the mean and the standard deviation of the noise after we have computed the Picard parameter k using the Lilliefors normality test.

In case there is no noise, we expect that the $|\beta_i|$ will decrease in average and that the DPC will be satisfied. That means that we do not expect the last β values to resemble a normal distribution, so the Lilliefors test will fail. Thus we would need to keep all the singular values and we would set $k = n + 1$.

In Matlab, the Lilliefors method is implemented using the `lillietest` command. It receives as input the sample, and it returns 0 if with 95% certainty the sample comes from a normal distribution and 1 if it does not.

The resulting algorithm is given in Algorithm 1.

2.5 Results

We designed Algorithm 1 to give us a conservative overestimate of k. We are looking for an overestimate because we want to make sure that we will not lose any singular values that would give important information about the image, thus oversmoothing the image. Ignoring fewer singular values solves this problem but also adds noise components. A conservative overestimate helps us not neglect important information but not add too much of the noisy part of the image in the solution.

In case when the DPC is satisfied, the algorithm will return $k = n + 1$ and estimate the mean and standard deviation of the noise to be zero.

Input: The vector $\boldsymbol{\beta}$ of length n with at least 14 trailing entries due to noise.

Output: The estimated Picard parameter k and the estimated standard deviation of the noise (exp_stdev) and the experimental mean of the noise (exp_mean).

$\mathbf{h} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$;

check=sum($\mathbf{h}$(:));

start=n-3;

while check < 10 **do**

  $\mathbf{h} = [\text{lillietest}(\boldsymbol{\beta}(\text{start} : n)), h(1), h(2), \cdots, h(9)]$

  start=start-1;

  check=sum($\mathbf{h}$(:));

end while

k=start+10;

if $k < n - 3$ **then**

  exp_stdev=std($\boldsymbol{\beta}(k : n)$);

  exp_mean=mean($\boldsymbol{\beta}(k : n)$);

else

  exp_stdev= 0;

  exp_mean= 0;

  $k = n + 1$;

end if

Algorithm 1: Picard Parameter Estimation

The Algorithm works well as we illustrate here for the examples discussed earlier in this chapter.

For the problem in Figure 2.2, the result from the code in this case is $k = 2253$. The manual estimation is approximately 1800. For the problem in Figure 2.3, the result from the code in this case is $k = 877$. The manual estimation is approximately 800. In both cases, the parameter estimated by the code is larger than the one estimated manually by the user. That means that there is less chance of disregarding important information given by the image.

Figure 2.1 is the Picard plot for the same image blurred with the same blurring matrix but with no added noise. The result from the code is $k = 4065$ which is very close to the dimension of the problem, 4096. Using the Picard plot in Figure 2.1 we can see that since there is no noise, the elements of $\boldsymbol{\beta}$ keep decreasing on average faster than the corresponding singular values. That means that all of the singular values should be used. In cases like this when the Picard Parameter is computed to be very close to the size of the image, the user might choose to disregard it and use all the singular values.

2.6 Conclusions

In this chapter, we reviewed the discrete Picard condition. We introduced the Picard Parameter and we described manual and automatic ways of computing it. The manual approach is based on the Picard plots and the automatic on histograms and the Lilliefors method for normality testing. We also provided numerical exam-

ples. Estimating the Picard parameter is essential in the next chapters that present the restoration of images using spectral filters. Knowing the Picard parameter will reduce the computational cost of the restoration, and provide approximations of the standard deviation and the mean of the noise that are essential for computing the optimal parameters for the spectral filters we will introduce.

Chapter 3: Near Optimal Filter for TSVD (SOF-TSVD)

3.1 Introduction

Recall from Chapter 1 that our goal is to find a approximate solution to the ill-posed deblurring problem (1.1):

$$\mathbf{b} = \mathbf{A}\mathbf{x}_{true} + \mathbf{e}.$$

Noise makes it difficult to find the solution. One way to find a good estimate of the solution is by designing filters to diminish the effect of the noise. In the work presented in this chapter, we follow an approach similar to the one of O'Leary for the Tikhonov Filter [22] to compute the Optimal TSVD Filter.

In this work, we deviate from the usual definition of the commonly used filters such as the Tikhonov, the TSVD, and the TSCM filters in terms of the index of the ordered singular values. We, instead, convert that notation to one that involves the singular value, rather than the index of the singular value. This gives uniformity for all the filters whether they are continuous or discrete.

The Tikhonov filter is

$$\phi_\lambda(\sigma_i) = \frac{\sigma_i^2}{\sigma_i^2 + \lambda}, \text{ for } i = 1, ..., n, \tag{3.1}$$

with a continuous parameter $\lambda \in \mathbb{R}^+$, and the TSVD filter is

$$\phi_\lambda(\sigma_i) = \begin{cases} 1 & \text{if } 1 \leq i \leq \lambda, \\ 0 & \text{otherwise }, \end{cases} \tag{3.2}$$

parameter $\lambda \in \{1, \ldots, n\}$.

In Section 3.2, we compute the optimal TSVD filter given the Picard parameter computed in Chapter 2. Later, in Section 3.3, we use standard techniques for the special case of blurs that are separable in order to save time in the SVD. The numerical experiments presented in Section 3.4 compare our choice of λ with two standard approaches, the Discrepancy Principle and Generalized Cross Validation. We show that the optimal filter that we compute performs better than that computed using the Discrepancy Principle or Generalized Cross Validation when the Picard parameter is chosen wisely. In addition, the method gives relative errors close to the minimal possible.

3.2 Derivation of the optimal TSVD filter (SOF-TSVD method)

Recall that the solution of the problem (1.3) is

$$\mathbf{x}_{naive} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i = \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i} \mathbf{v}_i. \tag{3.3}$$

This solution is naive since it contains a lot of noise at high frequencies that destroys the computed restored image. The high frequencies appear when the denominator (i.e., the singular value) is small.

A filtered solution takes the form

$$\mathbf{x}_\lambda = \sum_{i=1}^{n} \phi_\lambda(\sigma_i) \frac{\beta_i}{\sigma_i} \mathbf{v}_i. \tag{3.4}$$

The truncated SVD filter leaves out the singular values that are smaller than some value, and the Tikhonov regularization introduces weights to the components of the solution that decrease as the components are more likely to be contaminated by noise.

Using the truncated SVD (TSVD), the regularized solution becomes

$$\mathbf{x}_{TSVD} = \sum_{i=1}^{\lambda} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i = \sum_{i=1}^{\lambda} \frac{\beta_i}{\sigma_i} \mathbf{v}_i. \tag{3.5}$$

For the optimal truncated SVD, we need to compute the $\lambda \in \mathbb{R}^+$ for which the solution is as close to the noise-free problem's solution as possible. This noise-free solution is

$$\mathbf{x}_{true} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T (\mathbf{b} - \mathbf{e})}{\sigma_i} \mathbf{v}_i = \sum_{i=1}^{n} \frac{\beta_i - \epsilon_i}{\sigma_i} \mathbf{v}_i, \tag{3.6}$$

where $\epsilon_i = \mathbf{u}_i^T \mathbf{e}$.

Ideally, we need to minimize the norm of the error:

$$\min_{\lambda} \|\mathbf{x}_{TSVD} - \mathbf{x}_{true}\|^2 \equiv \min_{\lambda} f(\lambda). \tag{3.7}$$

This is not possible, though, since the original image $\mathbf{x}_{true}$ is unknown. If we knew the noise $\mathbf{e}$, we could recover $\mathbf{x}_{true}$, so we want to estimate the noise. If the noise-free system satisfies the discrete Picard condition, we can estimate the properties of the noise, i.e., its mean and its standard deviation, and use this to estimate $f(\lambda)$.

We see from (3.5), (3.6), and (3.7) that the error function is

$$\begin{aligned} f(\lambda) &= \|\mathbf{x}_{TSVD} - \mathbf{x}_{true}\|^2 \\ &= \sum_{i=1}^{\lambda} \left(\frac{\beta_i}{\sigma_i} - \left(\frac{\beta_i - \epsilon_i}{\sigma_i} \right) \right)^2 + \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i - \epsilon_i}{\sigma_i} \right)^2 \\ &= \sum_{i=1}^{\lambda} \left(\frac{\epsilon_i}{\sigma_i} \right)^2 + \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i - \epsilon_i}{\sigma_i} \right)^2 \\ &= \sum_{i=1}^{n} \left(\frac{\epsilon_i}{\sigma_i} \right)^2 + \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i}{\sigma_i} \right)^2 - 2 \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i \epsilon_i}{\sigma_i^2} \right). \end{aligned} \tag{3.8}$$

If the third summation did not exist, the minimum value would be attained for $\lambda = n$ since the first summation is independent of λ and the second one is non-increasing as λ increases. This means that we would use the full SVD and we would not drop any of the singular values.

But for the third summation we have that

- The terms for $i \approx n$ tend to be the largest (in absolute value) since the denominators are the smallest.

- If the system satisfies the discrete Picard condition, then for terms with $i \geq k$, where $k < n$ is the Picard parameter defined in Section 2.2, $\epsilon_i \approx \beta_i$.

Therefore, we can approximate $f(\lambda)$ by

$$\hat{f}(\lambda) = \sum_{i=1}^{n} \left(\frac{\epsilon_i}{\sigma_i} \right)^2 + \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i}{\sigma_i} \right)^2 - 2 \sum_{i=k}^{n} \left(\frac{\beta_i}{\sigma_i} \right)^2 - 2\mathcal{E} \left(\sum_{i=\lambda+1}^{k-1} \frac{\beta_i \epsilon_i}{\sigma_i^2} \right),$$

where $\mathcal{E}$ denotes the expected value and the last summation is understood to be empty if $\lambda > k - 2$.

This helps us because the expected value of the noise is computable and so the last three terms of $\hat{f}(\lambda)$ are computable too.

Because $\mathbf{b} = \mathbf{b}_{true} + \mathbf{e}$, and assuming that the noise has mean 0 and standard deviation s,

$$\begin{aligned}
\mathcal{E}(\beta_i \epsilon_i) &= \mathcal{E}(\mathbf{u}_i^T(\mathbf{b}_{true} + \mathbf{e})\epsilon_i) \\
&= \mathcal{E}(\mathbf{u}_i^T \mathbf{b}_{true}\epsilon_i) + \mathcal{E}(\mathbf{u}_i^T \mathbf{e}\, \epsilon_i) \\
&= \mathbf{u}_i^T \mathbf{b}_{true}\mathcal{E}(\epsilon_i) + \mathcal{E}(\epsilon_i^2) \\
&= \mathcal{E}(\epsilon_i^2) \\
&= s^2.
\end{aligned} \tag{3.9}$$

Therefore,

$$\hat{f}(\lambda) = \sum_{i=1}^{n} \left(\frac{\epsilon_i}{\sigma_i}\right)^2 + \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i}{\sigma_i}\right)^2 - 2\sum_{i=k}^{n} \left(\frac{\beta_i}{\sigma_i}\right)^2 - 2s^2 \sum_{i=\lambda+1}^{k-1} \frac{1}{\sigma_i^2}$$

Since the first and the third terms are independent of λ, we need to find the λ that minimizes the sum of the other two terms

$$g(\lambda) = \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i}{\sigma_i}\right)^2 - 2s^2 \sum_{i=\lambda+1}^{k-1} \frac{1}{\sigma_i^2}. \tag{3.10}$$

We name this approach SOF-TSVD, which stands for Statistical Optimal Filtering with the TSVD, since we use statistical methods to deal with the noise. Our algorithm is summarized as Algorithm 2.

The main cost is that of computing the SVD, $\mathcal{O}(n^3) = \mathcal{O}(n_1^3 n_2^3)$.

The above algorithm is for general blur when we express the image as a vector. For the construction of the blurring matrix, we need to take special care of the boundary conditions, i.e., the boundary pixels of the image blurred. More details are given in Section A.1.

1: Input: Blurring matrix $\mathbf{A}$, given image $\mathbf{b}$, and the Picard parameter k.

2: Output: Optimal parameter λ^*, restored image $\mathbf{x}^*$.

3: Compute SVD of $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

4: Let $\boldsymbol{\beta} = \mathbf{U}^T\mathbf{b}$.

5: Compute experimental standard deviation as:

$s = std(\boldsymbol{\beta}(k : end))$.

6: Find the minimizer λ^* of

$$g(\lambda) = \sum_{i=\lambda+1}^{n} \left(\frac{\beta_i}{\sigma_i}\right)^2 - 2s^2 \sum_{i=\lambda+1}^{k-1} \frac{1}{\sigma_i^2}$$

for $\lambda \in \{1, \ldots, k-1\}$ (for the SOF-TSVD$_k$ variant) or $\lambda \in \{1, \ldots, n\}$ (for the SOF-TSVD variant).

7: Then

$$\mathbf{x}^* = \sum_{i=1}^{\lambda^*} \frac{\beta_i}{\sigma_i} \mathbf{v}_i.$$

Algorithm 2: Computing the SOF-TSVD and SOF-TSVD$_k$ filtered solutions.

3.3 Two special cases

We discuss in this section two special cases of blur that reduce computational cost.

3.3.1 Vertical blur

Vertical blur occurs when the pixel values of the blurred image depend only on the values of the pixels that are in the same column in the original image. In this case, deblurring each column of the image is an independent problem. For an image with n_1 rows of pixels, if the blur for each column of pixels is the same, then we can represent it as an $n_1 \times n_1$ matrix $\mathbf{A}$ and our model becomes

$$\mathbf{AX} + \mathbf{E} = \mathbf{B}.$$

For each column $\mathbf{b}_j$ of the image, we use the method described above to compute the restored column $\mathbf{b}_j$. The final restored image is the set of all the restored columns. If we denote $\boldsymbol{\beta} = \mathbf{U}^T\mathbf{B}$, then the last two steps in Algorithm 2 should be replaced by:

for every column j in the image **do**

Find the minimizer λ_j^* of

$$g(\lambda) = \sum_{i=\lambda+1}^{n_1} \left(\frac{\beta_i}{\sigma_i}\right)^2 - 2s^2 \sum_{i=\lambda+1}^{k-1} \frac{1}{\sigma_i^2}$$

for $\lambda \in \{1, \ldots, k-1\}$ (for the SOF-TSVD$_k$ variant) or $\lambda \in \{1, \ldots, n\}$ (for the SOF-TSVD variant).

Then

$$\mathbf{x}_j^* = \sum_{i=1}^{\lambda^*} \frac{\beta_{ij}}{\sigma_i} \mathbf{v}_i.$$

end for

The cost of this method is $\mathcal{O}(n_1^3)$, since the most costly operation we need to do is to compute the SVD of the $n_1 \times n_1$ matrix. Everything else only requires simple operations of matrices of cost $\mathcal{O}(n_1^2)$. If we do those n_2 times for the columns of the image, then the total cost is $\mathcal{O}(n_1^3 + n_1^2 n_2)$.

Vertical blur is a special case of separable blur that is discussed next.

3.3.2 Kronecker products

In the general blur approach, the sizes of the matrices that are used in the computations are large. This results in large computational cost. In particular cases though, we can increase the efficiency of the algorithms. Some blurring operators are *separable* and can be written as a Kronecker product of two other matrices of sizes that correspond to the number of rows and the number of columns in the image array. An example is the boxcar filter that is developed by the combination of two independent blurring motions, horizontal and vertical.

A separable blurring matrix can be decomposed as $\mathbf{A} = \mathbf{A}_r \otimes \mathbf{A}_c$,where $\mathbf{A}_r$ is the blurring that affects the rows of the image, $\mathbf{A}_c$ is the blurring that affects the columns, and $\otimes$ denotes the Kronecker product of matrices. If $\mathbf{A}_r$ is the identity matrix, then the blur is vertical blur.

For separable blur, the problem becomes: Given the blurred image $\mathbf{B}$ and the matrices $\mathbf{A}_r$ and $\mathbf{A}_c$, find the approximation of the true image $\mathbf{X}$ when $(\mathbf{A}_r \otimes \mathbf{A}_c)\,\mathbf{x} + \mathbf{e} = \mathbf{b}$, or equivalently,

$$\mathbf{A}_c \mathbf{X} \mathbf{A}_r^T + \mathbf{E} = \mathbf{B},$$

where $\mathbf{E}$ is random noise. In Section A.1.2, we discuss one way to compute $\mathbf{A}_r$ and $\mathbf{A}_c$.

Let the SVDs of the matrices $\mathbf{A}_r$ and $\mathbf{A}_c$ be $\mathbf{A}_r = \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^T$ and $\mathbf{A}_c = \mathbf{U}_c \mathbf{\Sigma}_c \mathbf{V}_c^T$. Then, except for ordering, the SVD of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{A}_r \otimes \mathbf{A}_c = (\mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^T) \otimes (\mathbf{U}_c \mathbf{\Sigma}_c \mathbf{V}_c^T) = (\mathbf{U}_r \otimes \mathbf{U}_c)(\mathbf{\Sigma}_r \otimes \mathbf{\Sigma}_c)(\mathbf{V}_r \otimes \mathbf{V}_c)^T,$$

and can be used without being explicitly formed.

The only problem with this is that even if the SVD of the matrix A can be written as a Kronecker product and can be used without being explicitly formed, the diagonal matrix of singular values $\mathbf{\Sigma}_r \otimes \mathbf{\Sigma}_c$ is not ordered properly; i.e., the singular values are not in decreasing order any more. So, in computing our filter, we need to account for this correctly.

Everything that has been sorted will be denoted by its name under a hat.

Now let's call $\mathbf{S}$ the $n_1 \times n_2$ matrix that contains the singular values of the matrix $\mathbf{\Sigma}_c \otimes \mathbf{\Sigma}_r$ as its elements. An ordered vector $\widehat{\boldsymbol{\sigma}}$ gives the singular values of the Kronecker product matrix in nonincreasing order. In Matlab, it is computed as follows.

```
S=diag(Sc)*(diag(Sr))';
sigma=reshape(S, n, 1);
```

`[sigma_hat, ind_sort]=sort(sigma,1,'descend');`

where `ind_sort` is the vector of the indices of the singular values when the singular values are sorted in descending order; i.e., `sigma(ind_sort(1))` is the largest singular value and `sigma(ind_sort(n))` the smallest. The inverse permutation vector `iprm(ind_sort)=(1:n)'` is used to reverse the process.

With the above, the algorithm now changes to Algorithm 3.

For an $n_1 \times n_2$ image, the matrix $\mathbf{A}_r$ is $n_2 \times n_2$ and the matrix $\mathbf{A}_c$ is $n_1 \times n_1$. The cost of the SVDs for these matrices is of order $O(n_2^3)$ and $O(n_1^3)$ respectively. The other costly steps are the formation of $\widetilde{\mathbf{B}}$ and $\mathbf{X}^*$, which can be done in $O(\min(n_1, n_2) \max(n_1, n_2)^2)$ time. So the total computational cost of the algorithm is $O(\max(n_1, n_2))^3$ which is much less than $O(n_1^3 n_2^3)$.

3.4 Numerical experiments

Numerical experiments were performed using the Barbara image with separable blur with varied resolution. The results for the SOF choice of parameter were compared to the corresponding ones coming from choosing λ using the Generalized Cross Validation (GCV) method and the Discrepancy Principle (DP) [14] and by minimizing the real error of the filtered solution.

3.4.1 Set-up

We apply the methods to the problem using the same noise sample. We manage that by using the same seed in the random number generation. The mean

1: Input: $\mathbf{A}_r$, $\mathbf{A}_c$, $\mathbf{B}$, and the Picard parameter k.

2: Output: λ^*, $\mathbf{X}^*$.

3: Compute the SVD of $\mathbf{A}_r = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^T$ and $\mathbf{A}_c = \mathbf{U}_c\mathbf{\Sigma}_c\mathbf{V}_c^T$.

4: Let $\widetilde{\mathbf{B}} = \mathbf{U}_c^T\mathbf{B}\mathbf{U}_r$.

5: Compute the $n_1 \times n_2$ matrix $\mathbf{S} = \text{diag}(\mathbf{\Sigma}_c)\text{diag}(\mathbf{\Sigma}_r)^T$ of singular values.

6: Let $\widehat{\boldsymbol{\sigma}}$ be the vector of singular values ordered in descending order, and let $\widehat{\boldsymbol{\beta}}$ contain the elements of $\widetilde{\mathbf{B}}$ permuted to correspond to the ordering of $\widehat{\boldsymbol{\sigma}}$.

7: Compute the experimental standard deviation as $s = std(\widehat{\boldsymbol{\beta}}(k : end))$.

8: Find the minimizer λ^* of

$$g(\lambda) = \sum_{i=\lambda+1}^{n} \left(\frac{\hat{\beta}_i}{\hat{\sigma}_i} \right)^2 - 2s^2 \sum_{i=\lambda+1}^{k-1} \frac{1}{\hat{\sigma}_i^2}$$

for $\lambda \in \{1, \ldots, k-1\}$ (for the SOF-TSVD$_k$ variant) or $\lambda \in \{1, \ldots, n\}$ (for the SOF-TSVD variant).

9: Then $\mathbf{X}^* = \mathbf{V}_c[(\widetilde{\mathbf{B}}./\mathbf{S}) .* \mathbf{\Phi}]\mathbf{V}_r^T$, where $\mathbf{\Phi}_{ij} = \begin{cases} 1 & \text{if } \mathbf{S}_{ij} \geq \hat{\sigma}(\lambda), \\ 0 & \text{otherwise.} \end{cases}$

Algorithm 3: Computing the SOF-TSVD and SOF-TSVD$_k$ filtered solutions for a separable blur.

of the noise is taken to be 0 and the standard deviation will vary, to explore how it affects the results.

The following results come from the 256×256 Barbara image. The tables show the results of five different methods of computing the solution of the problem together with some parameters of the noise. The first set of three tables is for a low resolution 64×64 image, and the last set of three tables is for the 128×128 image. For ease in reading, all the tables show the real standard deviation of the noise in the first row and the first two tables in each set show the experimental standard deviation computed by Algorithm 1 in the second row. The third row of the first two tables in each set tables shows the *signal-to-noise-ratio, (SNR)* of the blurred images which is $\text{SNR} = \frac{\mu}{s}$, where μ is the mean of the image $\mathbf{B}$ and s the standard deviation of the noise. The rest of the rows of the tables compare the results of five different methods using the TSVD filter. These methods are:

- R-TSVD: The real estimate computed by minimizing $\|\mathbf{x}_{filt} - \mathbf{x}_{true}\|$. Note that this is done with knowledge of the true solution.

- SOF-TSVD: The minimization of $g(\lambda)$ is performed over possible values $\lambda = \lambda_1, \ldots, \lambda_n$, using the Picard parameter to estimate the standard deviation.

- SOF-TSVDk: The minimization of $g(\lambda)$ is performed over possible values $\lambda = \lambda_1, \ldots, \lambda_{k-1}$, using the Picard parameter and the estimated standard deviation.

- GCV-TSVDk: TSVDk filtered solution with parameter chosen using Generalized Cross Validation.

- DP-TSVDk: TSVDk filtered solution with parameter chosen using the Discrepancy Principle.

In each set of tables, the first table compares the relative error of the methods, whereas the second table compares the parameter chosen, the index of the singular value at which we truncate, and the third the running time for each of the methods.

Looking at the relative errors, it is seen that the SOF-TSVDk method has in general a smaller relative error than the Discrepancy Principle method for the TSVDk filter and thus it better estimates the image. The relative error is also close to the real relative error when we know the true solution

Example 1

For the first example, the dimension of the Barbara image is 64×64.

Table 3.1: Relative errors $\frac{\|x_{filt}-x_{true}\|}{\|x_{true}\|}$ for a 64×64 image.

std	0.1	1	10	25
s	0.127	1.149	10.633	25.598
SNR	1117.136	111.729	11.188	4.486
R-TSVD	0.0296	0.0636	0.126	0.179
SOF-TSVD	0.0296	0.0639	0.126	0.180
SOF-TSVDk	0.0296	0.0639	0.126	0.180
GCV-TSVDk	0.0296	0.0639	0.126	0.180
DP-TSVDk	0.0500	0.101	0.244	0.428

Table 3.2: Estimation of parameter λ for a 64×64 image.

std	0.1	1	10	25
s	0.127	1.149	10.633	25.598
SNR	1117.136	111.729	11.188	4.486
R-TSVD	3596	2394	768	491
SOF-TSVD	3596	2377	768	567
SOF-TSVDk	3596	2377	768	567
GCV-TSVDk	3596	2394	768	491
DP-TSVDk	2640	944	178	18

We can see that as the standard deviation of the noise increases, the SNR decreases which means that the noise affects the image more. That has an effect in the results as well. The larger the standard deviation, the larger the relative errors are. But our SOF-TSVDk method works comparably to the R-TSVD method if the Picard parameter is estimated properly, i.e., if we use all the important information. In addition, SOF-TSVDk performs better than the TSVDk with parameter computed using the Discrepancy Principle, in general.

Figure 3.1: Solutions for low resolution 64×64 Barbara image and various noise levels.

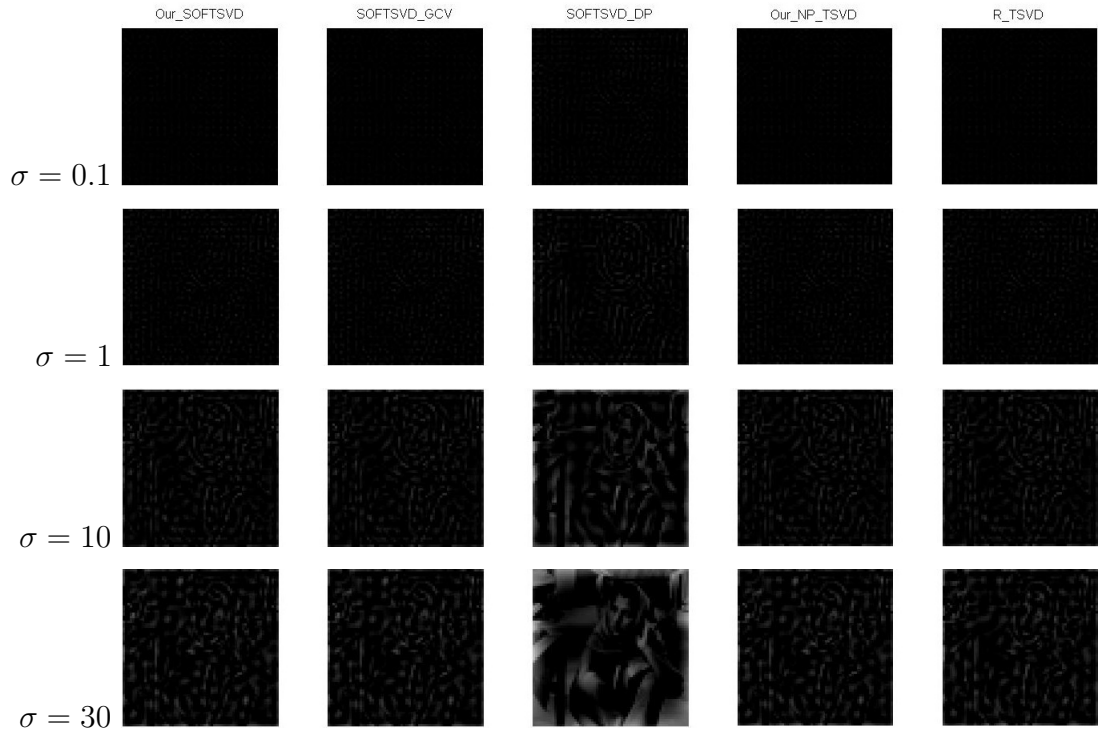

Figure 3.2: Errors for low resolution 64×64 Barbara image and various noise levels.

Example 2

For the second example, the dimension of the Barbara image is 128×128.

Table 3.3: Relative errors $\frac{\|x_{filt}-x_{true}\|}{\|x_{true}\|}$ for a 128×128 image.

std	0.1	1	10	25
s	0.113	1.057	10.424	25.618
SNR	1128.206	112.820	11.282	4.512
R-TSVD	0.0213	0.0512	0.105	0.149
SOF-TSVD	0.0214	0.0526	0.105	0.150
SOF-TSVDk	0.0214	0.0526	0.105	0.150
GCV-TSVDk	0.0214	0.0512	0.105	0.149
DP-TSVDk	0.383	0.0772	0.209	0.400

Table 3.4: Estimation of parameter λ for a 128×128 image.

std	0.1	1	10	25
s	0.113	1.057	10.424	25.618
SNR	1128.206	112.820	11.282	4.512
R-TSVD	14370	9084	2721	1522
SOF-TSVD	14172	8923	2721	1502
SOF-TSVDk	14172	8923	2721	1502
GCV-TSVDk	14370	9084	2721	1522
DP-TSVDk	9968	3477	395	34

We can see that as the standard deviation of the noise increases, the SNR decreases which means that the noise affects the image more. That has an effect in the results as well. The larger the standard deviation, the larger the relative errors

are. But our SOF-TSVDk method works comparably to the R-TSVD method if the Picard parameter is estimated properly, i.e., if we use all the important information. In addition, SOF-TSVDk performs better than the TSVDk with parameter computed using the Discrepancy Principle, in general.

Figure 3.3: Solutions for low resolution 128×128 Barbara image and various noise levels.

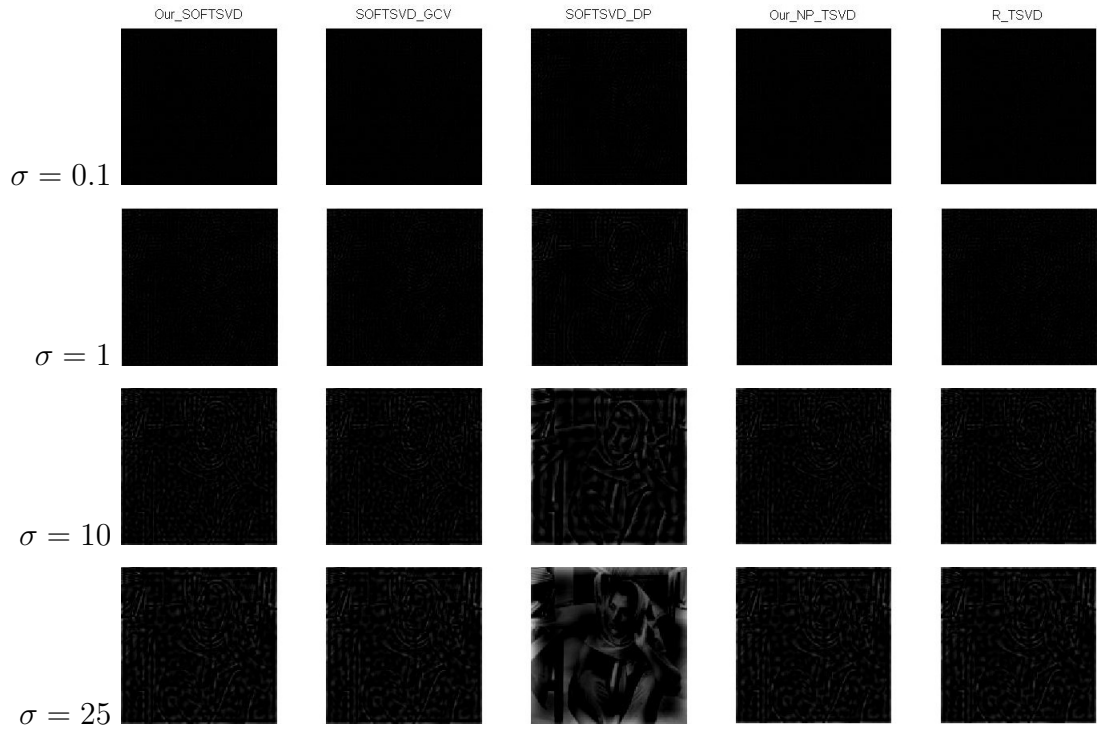

Figure 3.4: Errors for low resolution 128×128 Barbara image and various noise levels.

3.5 Conclusions

In this chapter, we presented a new algorithm for the computation of the near optimal parameter of the TSVD filter. The analysis uses the statistical properties of the noise as computed using the Picard parameter discussed in Chapter 2. Ideally, we want to minimize the norm of the error. That is not possible without knowing the noise. We overcome this problem by using expected values. That helps in the sense that we obtain a function that depends only on the standard deviation of the noise. This can be estimated by using the Picard parameter if the noise-free problem satisfies the Discrete Picard Condition.

We compared our new SOF-TSVD and SOF-TSVDk methods with methods that choose the TSVD parameter using GCV or the Discrepancy Principle. Experimental results showed that our method performs well compared to these two other methods, computing solutions with smaller relative error than the discrepancy principle. When the Picard parameter is computed properly, the solutions from the SOF-TSVD and SOF-TSVDk methods are comparable to the true optimal (uncomputable) solution.

Chapter 4: Near Optimal Parameter Choice and Uncertainty Quantification for General Filters

4.1 Introduction

We start again with the simple linear model for the image restoration of

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}, \tag{4.1}$$

where $\mathbf{b}$ is the observed image in vector form, $\mathbf{A}$ the blurring matrix that is known or can be estimated using the same device that captured the unknown image $\mathbf{x}$ to create $\mathbf{b}$, and $\mathbf{e}$ is the unknown random noise.

In Chapter 3 we used the TSVD filter to find the solution $\mathbf{x}$. The goal was to use statistical analysis to estimate the optimal parameter for the filter so that the computational cost becomes small and the solution more accurate.

In this chapter, we will follow a similar approach for general filters. Again the goal is to approximate the real solution $\mathbf{x}_{true}$. The existence of noise creates artifacts that destroy the naive solution, as we saw in Chapter 3. Like TSVD, the filters considered in this chapter weight the terms in the solution to reduce these artifacts.

Once again, if the SVD of $\mathbf{A}$ is $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$,

$\beta_i \equiv \mathbf{u}_i^T\mathbf{b}$, and $\epsilon_i \equiv \mathbf{u}^T\mathbf{e}$, then the true solution of problem (4.1) is

$$\mathbf{x}_{true} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T(\mathbf{b}-\mathbf{e})}{\sigma_i}\mathbf{v}_i = \sum_{i=1}^{n} \frac{\beta_i - \epsilon_i}{\sigma_i}\mathbf{v}_i. \tag{4.2}$$

As discussed in Chapter 2, the Discrete Picard condition is satisfied when, on average, the values β_i decay faster than the singular values σ_i. When noise is present, this might not be true. The filters weight the terms in (4.2) to decrease the effect of the noise on the image.

The filtered solution of problem (4.1) is

$$\mathbf{x}_{filt} = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i = \sum_{i=1}^{n} \phi_i \frac{\beta_i}{\sigma_i}\mathbf{v}_i, \tag{4.3}$$

where $\phi_i = \phi_{\boldsymbol{\lambda}}(\sigma_i)$ is the filter component.

So in general, we want to compute the parameters $\boldsymbol{\lambda}$ of the filter such that the quantity

$$\|\mathbf{x}_{true} - \mathbf{x}_{filt}\| \tag{4.4}$$

is minimized.

The Truncated SVD filter has already been discussed in Chapter 3. The Tikhonov filter has been analyzed in [22]. In Section 4.2, we show how to approximately minimize (4.4) for a general filter. Then in Section 4.3 we discuss how the uncertainty in our solution can be quantified by estimating $\|\mathbf{x}_{true} - \mathbf{x}_{filt}\|$. In the next chapter, we will present some specific cases of useful filters.

4.2 Our new near-optimal filter method

We repeat our computation of the error as in Chapter 3 but for the general filter rather than the TSVD filter.

Our assumptions are that the noise has mean 0 and standard deviation s, and that k is the Discrete Picard parameter.

We have

$$\begin{aligned}
\|\mathbf{x}_{true} - \mathbf{x}_{filt}\|^2 &= \sum_{i=1}^{n} \left[\left(\frac{\beta_i - \epsilon_i}{\sigma_i} - \phi_i \frac{\beta_i}{\sigma_i} \right) \right]^2 \\
&= \sum_{i=1}^{n} \left(\frac{(1-\phi_i)\beta_i - \epsilon_i}{\sigma_i} \right)^2 \\
&= \sum_{i=1}^{n} \frac{(1-\phi_i)^2\beta_i^2 + \epsilon_i^2 - 2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2} \\
&= \sum_{i=1}^{n} \frac{\epsilon_i^2}{\sigma_i^2} + \sum_{i=1}^{n} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2} \\
&= \sum_{i=1}^{n} \frac{\epsilon_i^2}{\sigma_i^2} + \sum_{i=1}^{k-1} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2} \\
&\quad + \sum_{i=k}^{n} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2}. \qquad (4.5)
\end{aligned}$$

The problem with this is that we do not know what the noise is and so we cannot compute ϵ_i or $\beta_i\epsilon_i$. What we can do is notice that the first term of the last equation does not depend on the filter and so when we try to minimize the error, we can ignore that term. So now we need to minimize the function

$$\hat{g}(\boldsymbol{\phi}) = \sum_{i=1}^{k-1} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2} + \sum_{i=k}^{n} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2}. \qquad (4.6)$$

We can do that by approximating this function with another one while esti-

mating $\beta_i\epsilon_i$ by $\mathcal{E}(\beta_i\epsilon_i)$. The values β_i are known since they are the blurred image values in the orthogonal space.

So now, using the fact that $\beta_i \approx \epsilon_i$ for $i \geq k$, we have

$$\begin{aligned}\hat{g}(\boldsymbol{\phi}) &\approx \sum_{i=1}^{k-1} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\mathcal{E}(\beta_i\epsilon_i)}{\sigma_i^2} + \sum_{i=k}^{n} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i^2}{\sigma_i^2} \\ &= \sum_{i=1}^{k-1} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)s^2}{\sigma_i^2} + \sum_{i=k}^{n} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i^2}{\sigma_i^2},\end{aligned}$$

since

$$\begin{aligned}\mathcal{E}(\beta_i\epsilon_i) &= \mathcal{E}(\mathbf{u}_i^T(\mathbf{b}_{true} + \mathbf{e})\epsilon_i) \\ &= \mathcal{E}(\mathbf{u}_i^T\mathbf{b}_{true}\epsilon_i) + \mathcal{E}(\mathbf{u}_i^T\mathbf{e}\,\epsilon_i) \\ &= \mathbf{u}_i^T\mathbf{b}_{true}\mathcal{E}(\epsilon_i) + \mathcal{E}(\epsilon_i^2) \\ &= \mathcal{E}(\epsilon_i^2) \\ &= s^2.\end{aligned} \tag{4.7}$$

assuming that $\mathbf{b} = \mathbf{b}_{true} + \mathbf{e}$, and that the noise has mean 0 and standard deviation s.

So our approach is to minimize the function

$$g(\boldsymbol{\lambda}) = \sum_{i=1}^{k-1} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)s^2}{\sigma_i^2} + \sum_{i=k}^{n} \frac{(1-\phi_i)^2\beta_i^2 - 2(1-\phi_i)\beta_i^2}{\sigma_i^2}, \tag{4.8}$$

with respect to the parameters $\boldsymbol{\lambda}$ that define the filter $\phi_i = \phi_{\boldsymbol{\lambda}}(\sigma_i)$, and thus find the near-optimal filter, which will give us the near-optimal solution. If the mean of the error distribution is nonzero, then the correct expression is more complicated, and this expression should not be used.

In this general case, the algorithm is given in Algorithm 4.

1: **Input**: Blurring matrix $\mathbf{A}$, given image $\mathbf{b}$, the Picard parameter k and the formula for the filter $\boldsymbol{\phi}_{\boldsymbol{\lambda}}$.

2: **Output**: Optimal parameter $\boldsymbol{\lambda}^*$ of the filter, restored image $\mathbf{x}^*$.

3: Compute the SVD of $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$.

4: Let $\boldsymbol{\beta} = \mathbf{U}^T\mathbf{b}$.

5: Estimate the experimental standard deviation as:

exp-stdev = $std(\boldsymbol{\beta}(k : end)$.

6: Find the minimizer $\boldsymbol{\lambda}^*$ of

$$g(\boldsymbol{\lambda}) = \sum_{i=1}^{k-1} \frac{(1-\phi_{\boldsymbol{\lambda}}(\sigma_i))^2\beta_i^2 - 2(1-\phi_{\boldsymbol{\lambda}}(\sigma_i))s^2}{\sigma_i^2} + \sum_{i=k}^{n} \frac{(1-\phi_{\boldsymbol{\lambda}}(\sigma_i))^2\beta_i^2 - 2(1-\phi_{\boldsymbol{\lambda}}(\sigma_i))\beta_i^2}{\sigma_i^2},$$

with respect to $\boldsymbol{\lambda}$.

7: Then, $x^* = \sum_{i=1}^{n} \phi_{\boldsymbol{\lambda}^*}(\sigma_i)\frac{\beta_i}{\sigma_i}\mathbf{v}_i$.

Algorithm 4: Computing the near optimal filter.

Remark 1. *It is easy to see that equation* (4.8) *agrees with the result obtained before for the TSVD. By substituting*

$$\phi_i = \begin{cases} 1 & \text{if } 1 \leq i \leq \lambda \\ 0 & \text{otherwise} \end{cases}$$

in (4.8)*, we have that*

$$\begin{aligned} g_1(\lambda) &= \sum_{i=\lambda+1}^{k-1} \frac{\beta_i^2 - 2s^2}{\sigma_i^2} + \sum_{i=k}^{n} \frac{\beta_i^2 - 2\beta_i^2}{\sigma_i^2} \\ &= \sum_{i=\lambda+1}^{n} \frac{\beta_i^2}{\sigma_i^2} - 2s^2 \sum_{\lambda+1}^{k-1} \frac{1}{\sigma_i^2} - 2\sum_{k}^{n} \frac{\beta_i^2}{\sigma_i^2}. \end{aligned} \tag{4.9}$$

But here again the last term is independent of λ*, so what we really need to minimize is*

$$g(\boldsymbol{\lambda}) = \sum_{i=\lambda+1}^{n} \frac{\beta_i^2}{\sigma_i^2} - 2s^2 \sum_{\lambda+1}^{k-1} \frac{1}{\sigma_i^2}, \tag{4.10}$$

which is the same quantity that we computed in Chapter 3, Equation (3.10). □

Remark 2. *Minimizing equation* (4.8) *for the Tikhonov filter will give the same solution as the one produced by O'Leary in [22].*

Substituting the Tikhonov filter

$$\phi_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda}, \text{ for } i = 1, ..., n$$

in (4.8)*, we have*

$$\begin{aligned} g_2(\lambda) &= \sum_{i=1}^{k-1} \frac{\left(\frac{\lambda}{\sigma_i^2+\lambda}\right)^2 \beta_i^2 - 2\frac{\lambda}{\sigma_i^2+\lambda} s^2}{\sigma_i^2} + \sum_{i=k}^{n} \frac{\left(\frac{\lambda}{\sigma_i^2+\lambda}\right)^2 \beta_i^2 - 2\frac{\lambda}{\sigma_i^2+\lambda} \beta_i^2}{\sigma_i^2} \\ &= \sum_{i=1}^{n} \frac{\left(\frac{\lambda}{\sigma_i^2+\lambda}\right)^2 \beta_i^2}{\sigma_i^2} - 2\sum_{i=1}^{k-1} \frac{\left(\frac{\lambda}{\sigma_i^2+\lambda}\right) s^2}{\sigma_i^2} - 2\sum_{i=k}^{n} \frac{\left(\frac{\lambda}{\sigma_i^2+\lambda}\right) \beta_i^2}{\sigma_i^2}. \end{aligned} \tag{4.11}$$

A way to minimize this is to find the zeros of the derivative

$$g_2'(\lambda) = \sum_{i=1}^{n} 2\frac{\lambda\beta_i^2}{(\sigma_i^2+\lambda)^3} - 2\sum_{i=1}^{k-1}\frac{s^2}{(\sigma_i^2+\lambda)^2} - 2\sum_{i=k}^{n}\frac{\beta_i^2}{(\sigma_i^2+\lambda)^2} \tag{4.12}$$

which is the same as finding the zeros of the function

$$g(\boldsymbol{\lambda}) = \sum_{i=1}^{n}\frac{\lambda\beta_i^2}{(\sigma_i^2+\lambda)^3} - \sum_{i=1}^{k-1}\frac{s^2}{(\sigma_i^2+\lambda)^2} - \sum_{i=k}^{n}\frac{\beta_i^2}{(\sigma_i^2+\lambda)^2} \tag{4.13}$$

which is Equation (2.2) *in [22].* □

4.3 Uncertainty quantification

The statistical analysis we used to develop a computable function from the norm of the error to be minimized can also be used to develop a criterion of uncertainty quantification.

Recall from (4.5) that

$$\begin{aligned} &\|\mathbf{x}_{true} - \mathbf{x}_{filt}\|^2 \\ = \;& \sum_{i=1}^{n}\frac{\epsilon_i^2}{\sigma_i^2} + \sum_{i=1}^{n}\frac{(1-\phi_i)^2\beta_i^2}{\sigma_i^2} - \sum_{i=1}^{k-1}\frac{2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2} - \sum_{i=k}^{n}\frac{2(1-\phi_i)\beta_i\epsilon_i}{\sigma_i^2}. \end{aligned}$$

In order to define the function (4.8), we ignored the first term of the above equation since it did not depend on the filter ϕ or the parameter λ. In order to estimate the error though, we need to keep the first term. Since that term is noise and is unknown, we cannot compute it and so we will use the expected value $\mathcal{E}(\epsilon_i^2)$ to estimate it.

We also take advantage of the fact that, by hypothesis, the true data components are approximately zero for $i = k, \ldots, n$. So we estimate the error only in the

directions corresponding to $i = 1, \ldots, k-1$, and we call this $\|\mathbf{x}_{true} - \mathbf{x}_{filt}\|_k$ So now,

$$\begin{aligned} & \|\mathbf{x}_{true} - \mathbf{x}_{filt}\|_k^2 \\ \approx & \sum_{i=1}^{k-1} \frac{s^2}{\sigma_i^2} + \sum_{i=1}^{k-1} \frac{(1-\phi_i)^2 \beta_i^2}{\sigma_i^2} - \sum_{i=1}^{k-1} \frac{2(1-\phi_i)s^2}{\sigma_i^2} \end{aligned}$$

This quantity is computable. The smaller this quantity is, the more we can believe in our filtered solution.

4.4 Conclusions

The statistical approach for the computation of the near-optimal parameters presented for the TSVD in Chapter 3 was extended in this chapter in a more general form. We call this general approach the SOF (Statistically Optimal Filtering method). With this SOF method, we can estimate near-optimal parameters for any filters whether the parameters to be determined are continuous or discrete, or both, and regardless of their number.

This method also provides a nice criterion for uncertainty quantification when the Discrete Picard condition is satisfied by our problem, since the error of the solution that we get from a specific filter can be estimated using the known data.

In order to show how our SOF method works compared to the established methods, Generalized Cross Validation (GCV) method and the Discrepancy Principle (DP), in Chapter 5, we develop new filters with different kind and number of parameters. In Chapter 6, we will see that our SOF method behaves comparably to the GCV and the DP methods.

Chapter 5: New Spectral Filters

5.1 Introduction

In this chapter, we introduce some new and useful special cases of the general filter $\phi_{\boldsymbol{\lambda}}(\sigma_i)$, $i = 1, \ldots, n$, for which we performed statistical analysis in Section 4.2.

We note that in order to use our SOF method, Algorithm 4, we only need to have a way to evaluate $\phi_{\boldsymbol{\lambda}}$ at the singular values, so we focus on the definition of $\phi_{\boldsymbol{\lambda}}$ and its geometric shape.

In Section 5.2, we discuss a new Truncated Tikhonov filter and in Section 5.3, the Truncated Singular Component Method filter. Later, in Section 5.4, we introduce a new hybrid method that combines the TSVD with the Tikhonov filter. The next section, Section 5.5, discusses a new Continuous TSVD filter as an example of how discrete filters can be modified to be viewed as continuous, which can create easier minimization problems. Heaviside and Tangent filters are briefly introduced in Section 5.6. Finally, another new method, the cubic spline with a general number of knots, is presented in Section 5.7. Numerical examples in the next chapter provide some insight about how these filters perform. Our SOF method for finding a near-optimal $\boldsymbol{\lambda}$ will be compared to Generalized Cross Validation (GCV) and the Discrepancy Principle (DP) for the same methods.

In this chapter we plot typical shapes of the filters using a blurring matrix whose singular values are plotted in Figure 5.1. When convenient we use the notation $\sigma(i) = \sigma_i$.

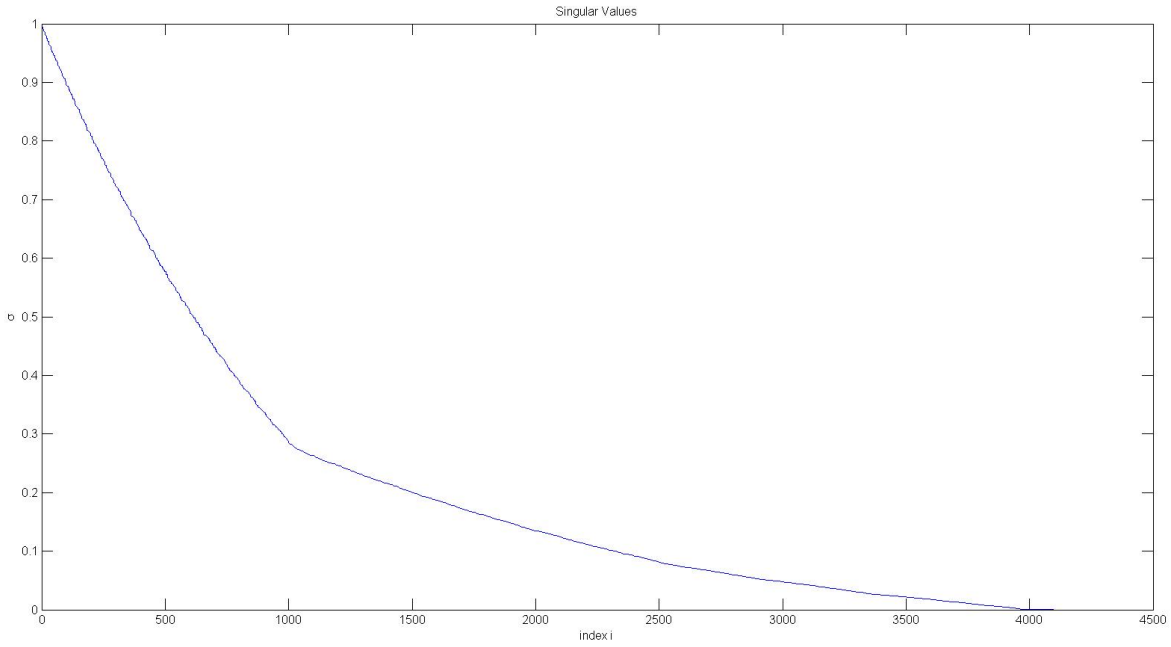

Figure 5.1: Singular values in descending order.

5.2 Truncated Tikhonov Filter (TTik)

The Truncated Tikhonov filter is a truncated form of the Tikhonov filter:

$$\phi_i = \begin{cases} \frac{\sigma_i^2}{\sigma_i^2+\lambda_2}, & \text{if } 1 \leq i \leq \lambda_1, \\ 0, & \text{otherwise,} \end{cases} \tag{5.1}$$

where λ_1 and λ_2 are the parameters of the filter, with $\lambda_1 \in \mathbb{Z}^+$ and $\lambda_1 < k$, k being the Picard Parameter, and $\lambda_2 \in \mathbb{R}^+$.

The function $\phi_{\boldsymbol{\lambda}}$, shown in Figure 5.2, is

$$\phi_{\boldsymbol{\lambda}}(\sigma) = \begin{cases} \frac{\sigma^2}{\sigma^2+\lambda_2}, & \text{if } \sigma \leq \sigma(\lambda_1), \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

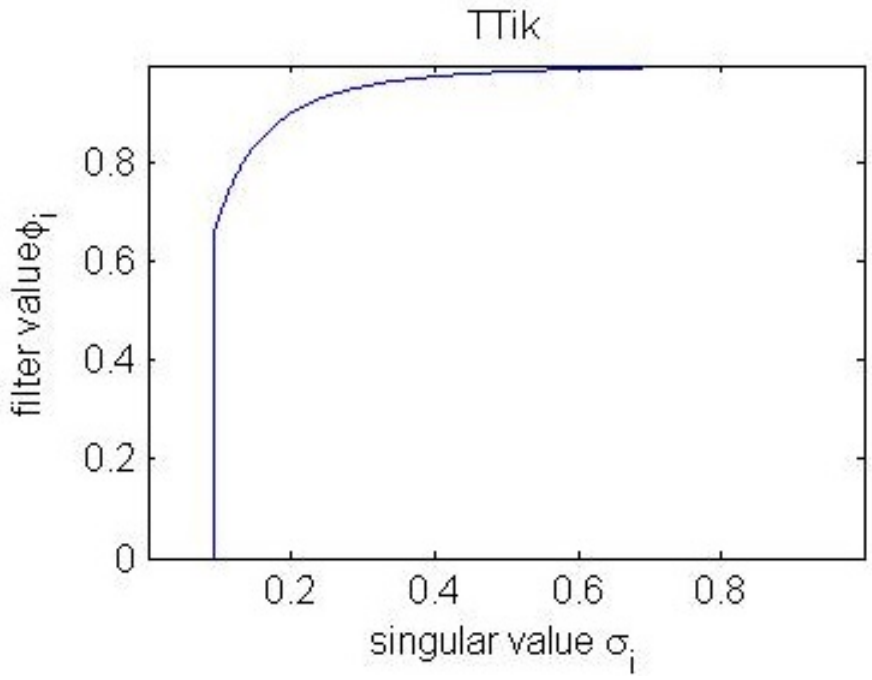

Figure 5.2: TTik filter.

Using this filter, we retain the good properties of the Tikhonov filter, i.e., we decrease the weight on the terms that contain the small singular values, and also we decrease the computational cost since we do not add the terms that do not contribute any significant information but are dominated by noise.

We can find the optimal parameter set for this filter by finding the optimal λ_2 for every possible value of λ_1 and then choosing the best overall. Alternatively, we can set $\lambda_1 = k - 1$ and solve for λ_2, as we did in our numerical experiments.

5.3 Truncated Singular Component Filter (TSCMk)

The Truncated Singular Component Method (TSCM) has been proposed by Rust [24]. The filter is very close to the Truncated Singular Value Decomposition filter but instead of only truncating with respect to the singular value's magnitude, it truncates with respect to $\mathbf{u}_i^T\mathbf{b}$ as well. The resulting TSCMk filter is

$$\phi_i = \begin{cases} 1, & \text{if } 1 \leq i \leq \lambda \text{ and } |\mathbf{u}_i^T\mathbf{b}| > \tau s^2, \\ 0, & \text{otherwise }, \end{cases} \tag{5.3}$$

where τ is truncation level, s the standard deviation of the noise elements, and $\lambda \in \mathbb{Z}^+$, $\lambda \leq k$, with k being the Picard Parameter.

This filter, shown in Figure 5.3, is unusual in that it is undefined except at the singular values $\sigma_1, \ldots, \sigma_n$ and depends on the right-hand-side data of the problem.

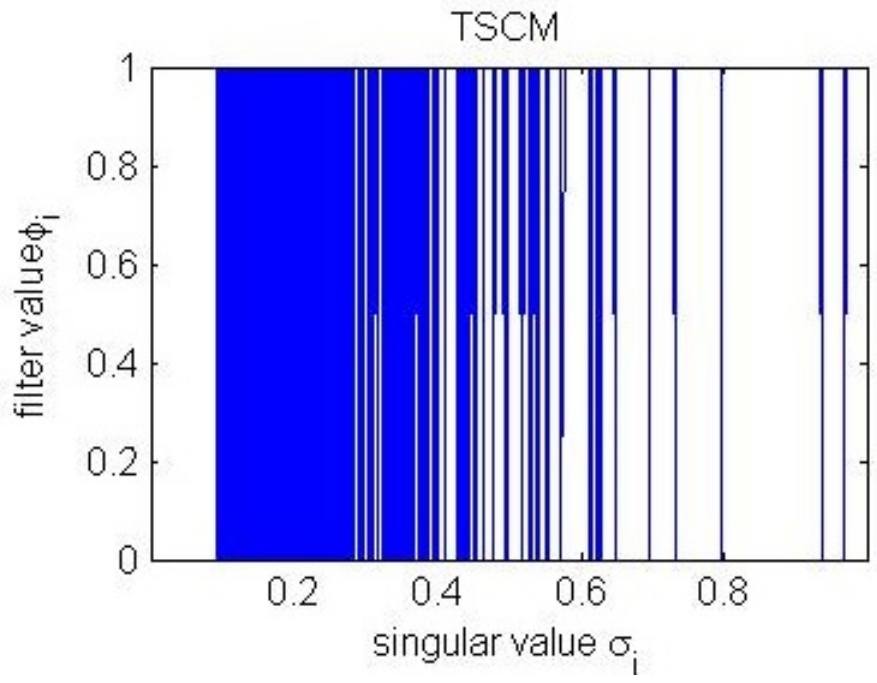

Figure 5.3: TSCM filter.

Given the values of the Picard parameter k and the truncation level τ, the filter depends only on the one parameter λ. So given a value for τ, the minimization in Algorithm 4 is easily done for the single parameter λ.

5.4 The Hybrid Tikhonov-TSVD (HYBR) Filter

In this section, we introduce a new, hybrid filter that combines the Tikhonov regularization filter with the TSVD filter.

Define the hybrid Tikhonov-TSVD filter as

$$\phi_i = \begin{cases} 1, & \text{if } i \leq \lambda_1, \\ \frac{\sigma_i^2}{\sigma_i^2+\lambda_3}, & \text{if } \lambda_1 < i \leq \lambda_2, \\ 0, & \text{if } \lambda_2 < i \leq n. \end{cases} \tag{5.4}$$

This filter depends on three parameters, $\lambda_1, \lambda_2 \in \mathbb{Z}^+$, and $\lambda_3 \in \mathbb{R}^+$.

That means that we assume that all of the singular values with an index less than λ_1 correspond to important information whereas those which have an index greater than λ_2 correspond to components dominated by noise and as such they are discarded. In addition, we want $\lambda_2 \leq k$, where k is the Picard parameter. For the intermediate singular values, we apply the Tikhonov filter.

The function $\phi_{\boldsymbol{\lambda}}$, shown in Figure 5.4, is defined by

$$\phi_{\boldsymbol{\lambda}}(\sigma) = \begin{cases} 1, & \text{if } \sigma \geq \sigma(\lambda_1), \\ \frac{\sigma^2}{\sigma^2+\lambda_3}, & \text{if } \sigma(\lambda_1) > \sigma \geq \sigma(\lambda_2), \\ 0, & \text{if } \sigma(\lambda_2) > \sigma \geq \sigma_n. \end{cases}$$

It is useful to note here that with the assumption that the problem satisfies

the discrete Picard condition with parameter k, the values that the parameters can take are limited: $0 \leq \lambda_1 \leq \lambda_2 < k$, and $\lambda_3 \in \mathbb{R}^+$.

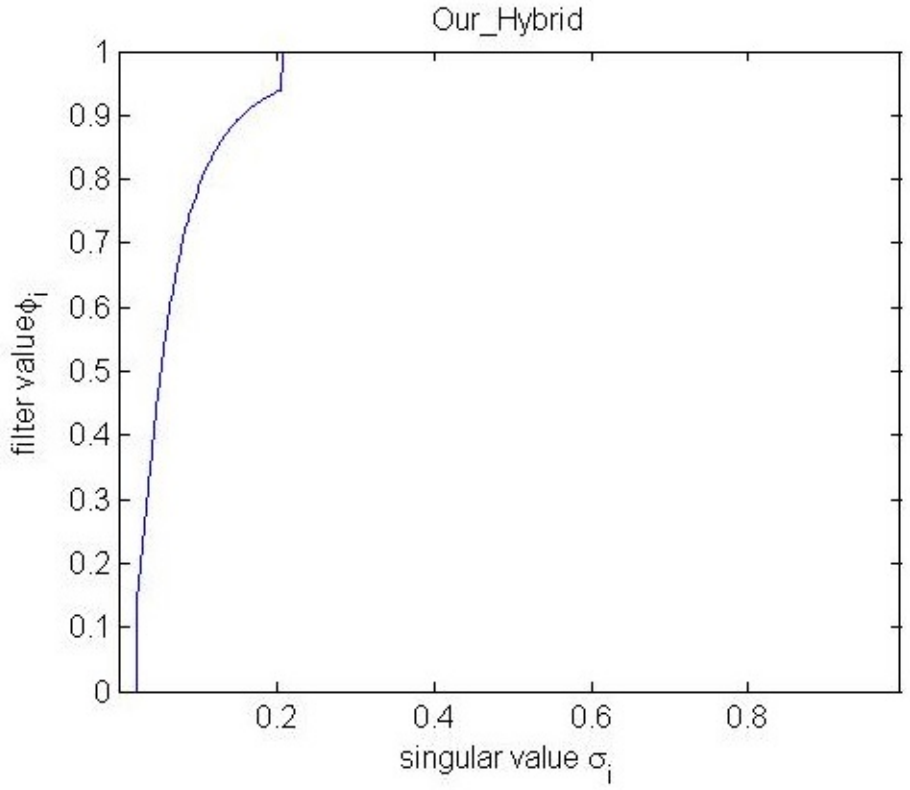

Figure 5.4: Hybrid filter.

In Algorithm 4, we minimize $g(\boldsymbol{\lambda})$ for each possible value of (λ_1, λ_2) and then choose the best overall, where $0 < \lambda_1 \leq \lambda_2 < k$.

If instead we constrain $\lambda_2 = k - 1$, then the hybrid filter is simplified:

$$\phi_i = \begin{cases} 1, & \text{if } i \leq \lambda_1, \\ \frac{\sigma_i^2}{\sigma_i^2 + \lambda_3}, & \text{if } \lambda_1 < i \leq k-1, \\ 0, & \text{if } i \geq k. \end{cases} \tag{5.5}$$

For simplicity in notation and comparison to the filter described earlier, we do not

rename the continuous parameter λ_3.

Fuhry and Reichel [8] have created a similar filter called the regularized Tikhonov Filter. Their filter is the following:

$$\phi_i = \begin{cases} 1, & \text{if } \sigma_i > \lambda, \\ \frac{\sigma_i^2}{\lambda^2}, & \text{if } \lambda > \sigma_i. \end{cases} \tag{5.6}$$

5.5 A continuous TSVD (ContTSVD) Filter

Some filters have continuous parameters (e.g., Tikhonov), some have discrete parameters (e.g., TSVD) and some have both (e.g., the new HYBR filter of Section 5.4).

Continuous functions of continuous parameters are somewhat easier to handle, so we show in this section, using TSVD as an example, how discrete parameters might sometimes be replaced by continuous ones, making a closely related filter.

It is possible to formulate TSVD with a continuous parameter, and we develop this new filter here. Recall that the TSVD filter is

$$\phi_i = \begin{cases} 1, & \text{if } i \leq \lambda, \\ 0, & \text{if } i > \lambda. \end{cases}$$

Here λ is an integer parameter. For some particular integer value λ the singular values with index $i \leq \lambda$ are kept but for $i \geq \lambda$, we assume that they are the cause of the noise features in the image and we discard them.

By changing this to

$$\phi_i = \begin{cases} 1, & \text{if } \sigma_i \geq \lambda, \\ 0, & \text{if } \sigma_i < \lambda, \end{cases} \tag{5.7}$$

λ becomes a continuous parameter. The cut-off point would depend on the value of the singular values and not the indices but still the function (4.8) would be discontinuous with respect to the parameter λ.

One idea to create a function of a continuous parameter λ would be the following:

Suppose λ is such that $\sigma_{i+1} < \sigma_i \leq \lambda \leq \sigma_{i-1}$. Then define

$$\phi_\lambda(\sigma) = \begin{cases} 1, & \text{if } \lambda \leq \sigma, \\ \frac{\sigma - \sigma_{i+1}}{\lambda - \sigma_{i+1}}, & \text{if } \sigma_{i+1} \leq \sigma < \lambda, \\ 0, & \text{if } \sigma_{i+1} > \sigma. \end{cases} \tag{5.8}$$

Here, σ_i is defined in a way that enforces a gap between singular value σ_i and singular value σ_{i+1} and thus avoids the singularities in the denominator of the filter function value.

Thus,

$$\phi_\lambda(\sigma_j) = \begin{cases} 1, & \text{if } j < i, \\ \frac{\sigma_i - \sigma_{i+1}}{\lambda - \sigma_{i+1}}, & \text{if } j = i, \\ 0, & \text{if } j > i. \end{cases}$$

In the case where $i = n$, define $\phi_\lambda(\sigma_n) = 1$. Notice that the filtered solution would then be the naive solution since $\phi_\lambda(\sigma_j) = 1$ for $j = 1, \ldots, n$.

Unlike the standard TSVD filter, this filter is a continuous function of σ. But it is not a continuous function of λ: at the point where λ crosses a singular value, the filter loses its continuity.

The following is a successful approach to the problem of transforming the discrete TSVD to one that is continous with respect to λ.

First, let's assume that the continuous parameter of the filter is λ and that $\sigma_{i+1} \leq \lambda < \sigma_i$. Let's also assume that the next distinct singular value less than σ_{i+1}

is $\sigma_{i+\ell}$. Distinct means that the singular values are some chosen tolerance apart. We could choose the tolerance to be 10^{-6}. Then we define a parameter μ, the point at which the filter becomes zero, to be

$$\mu = \sigma_{i+\ell} + \frac{\sigma_{i+1} - \sigma_{i+\ell}}{\sigma_i - \sigma_{i+1}}(\lambda - \sigma_{i+1}). \tag{5.9}$$

Thus the ratio of the distance $\lambda - \sigma_{i+1}$ to the distance $\sigma_i - \sigma_{i+1}$ is equal to the ratio of the distance $\mu - \sigma_{i+\ell}$ to the distance $\sigma_{i+1} - \sigma_{i+\ell}$; see Figure 5.5. So as λ slides toward σ_{i+1}, μ slides toward $\sigma_{i+\ell}$.

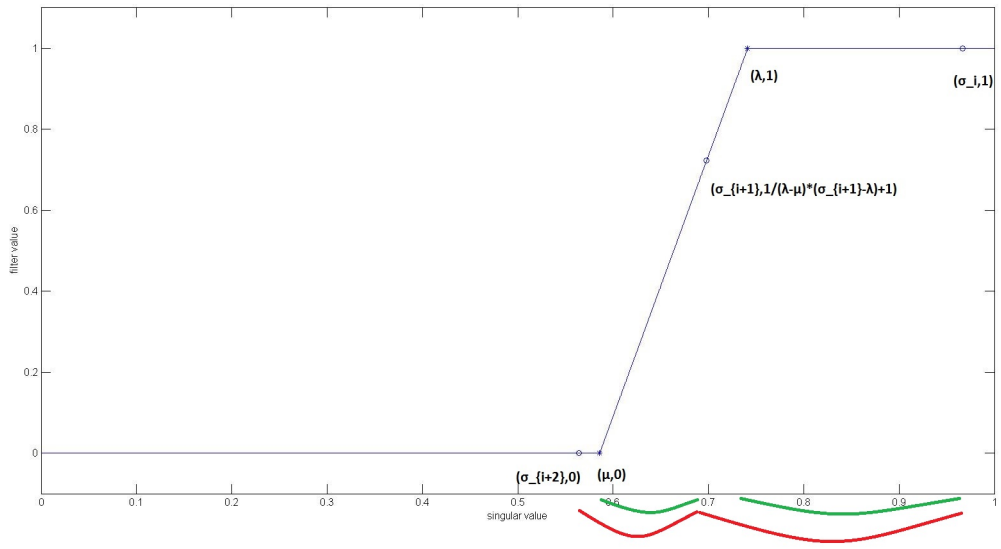

Figure 5.5: ContTSVD μ parameter definition. The ratio of the distance denoted by the right green curve over the right red curve is the same as the ratio of the distance denoted by the left green curve over the distance denoted by the left right curve.

With that, we define the continuous TSVD filter as

$$\phi_\lambda(\sigma) = \begin{cases} 1, & \text{if } \sigma > \lambda, \\ \frac{\sigma-\lambda}{\lambda-\mu} + 1, & \text{if } \mu < \sigma \leq \lambda, \\ 0, & \text{if } \sigma \leq \mu. \end{cases} \tag{5.10}$$

This is a continuous version of the TSVD filter. In general, we don't need to evaluate the filter at any points other than the singular values. So, with the above notation and definition of λ and μ, we have that

$$\phi_\lambda(\sigma_j) = \begin{cases} 1, & \text{if } j < i, \\ \frac{\sigma_j-\lambda}{\lambda-\mu} + 1, & \text{if } j = i, \\ 0, & \text{if } j > i. \end{cases} \tag{5.11}$$

.

The filter can be seen in the left plot of Figure 5.6. The right plot shows the area of interest of the filter. It is obvious that the filter is different than the TSVD filter but based on the definition, only at one singular value. The filter transitions from 1 at λ to 0 at μ.

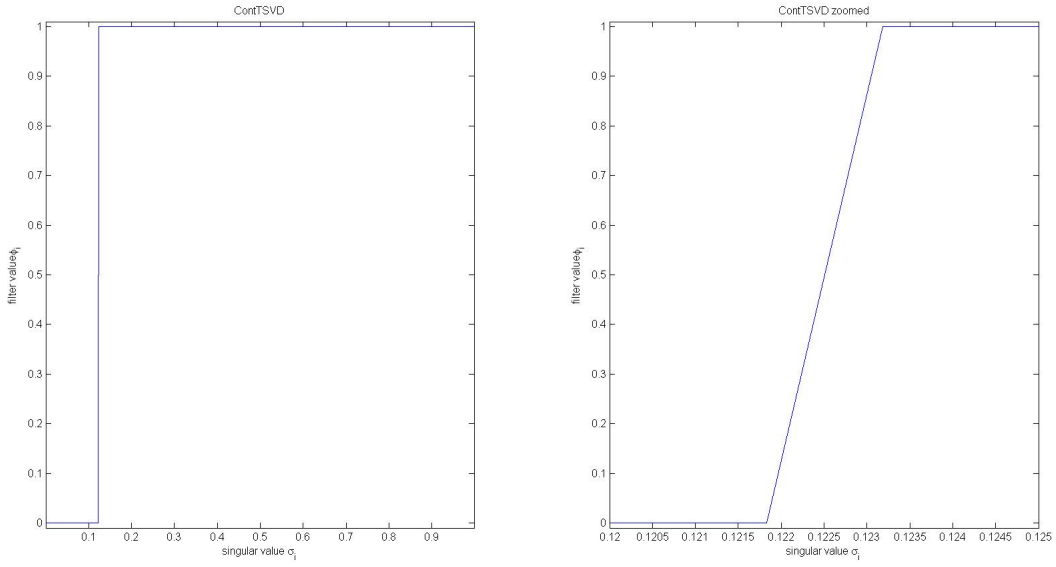

Figure 5.6: ContTSVD filter on the left and zoomed area of interest of the ContTSVD filter on the right.

In this section we transformed a discrete filter to a continuous one that can also be used in Algorithm 4.

5.6 The Heaviside (HS) and the Tangent (TAN) Filters

In general, the filters give more weight to the components of the solution that correspond to the large singular values and less to those with smaller singular values. In other words, they are low-pass filters.

The TSVD can be considered as a Heaviside function.

$$H(y) = \begin{cases} 1, & \text{if } y > 0, \\ \frac{1}{2}, & \text{if } y = 0, \\ 0, & \text{if } y < 0. \end{cases} \tag{5.12}$$

We can use a function that is close to $H(y)$, modified so that some value $\tilde{\sigma}$

maps to 0. Some examples are shown in Figure 5.7.

$$\phi_\lambda(\sigma_i) = e^{-e^{-(\sigma_i - \tilde{\sigma})/\lambda}} \tag{5.13}$$

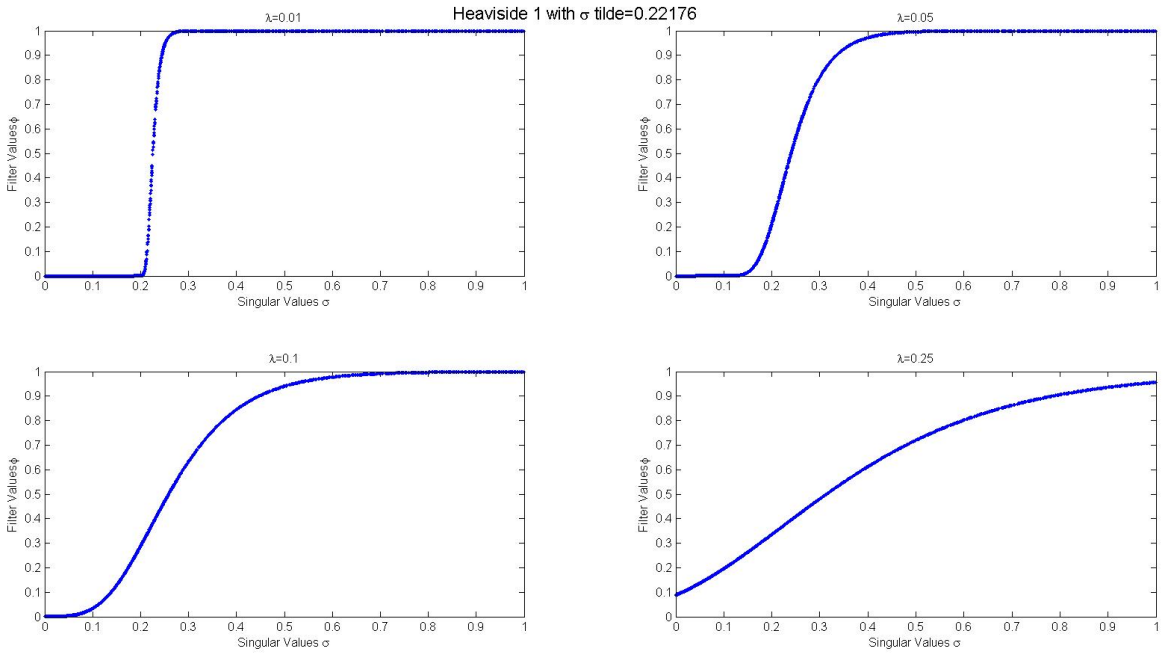

Figure 5.7: Heaviside 1 Filter.

Alternatively, we can use the filters

$$\phi_\lambda(\sigma) = \frac{1}{1 + e^{-(\sigma - \tilde{\sigma})/\lambda}} \tag{5.14}$$

shown in Figure 5.8.

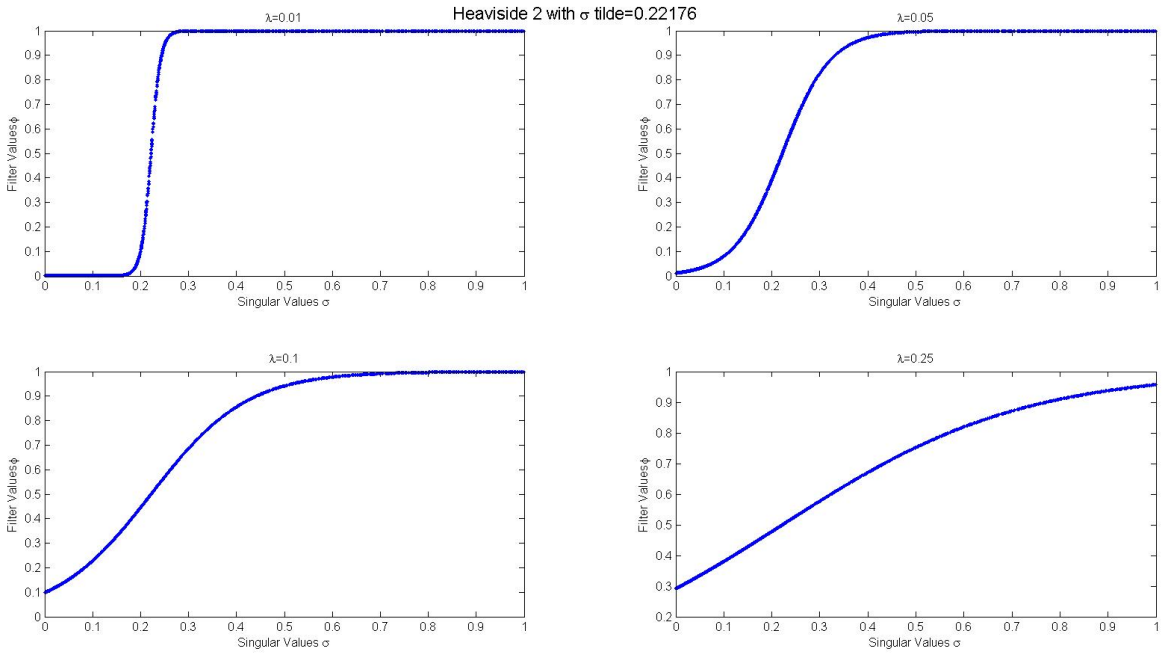

Figure 5.8: Heaviside 2 Filter.

In both cases, the filters depend on the continuous parameter λ and the choice of the value $\sigma_1 \leq \tilde{\sigma} \leq \sigma_n$, which can either be specified or used as a second parameter in the filter.

To avoid the choice of the centering parameter, we can use the Tangent filter

$$\phi_{\boldsymbol{\lambda}}(\sigma) = \lambda_1 \tan\left(\lambda_2 \frac{\pi}{4} \frac{\sigma}{\sigma_1}\right), \tag{5.15}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2]$.

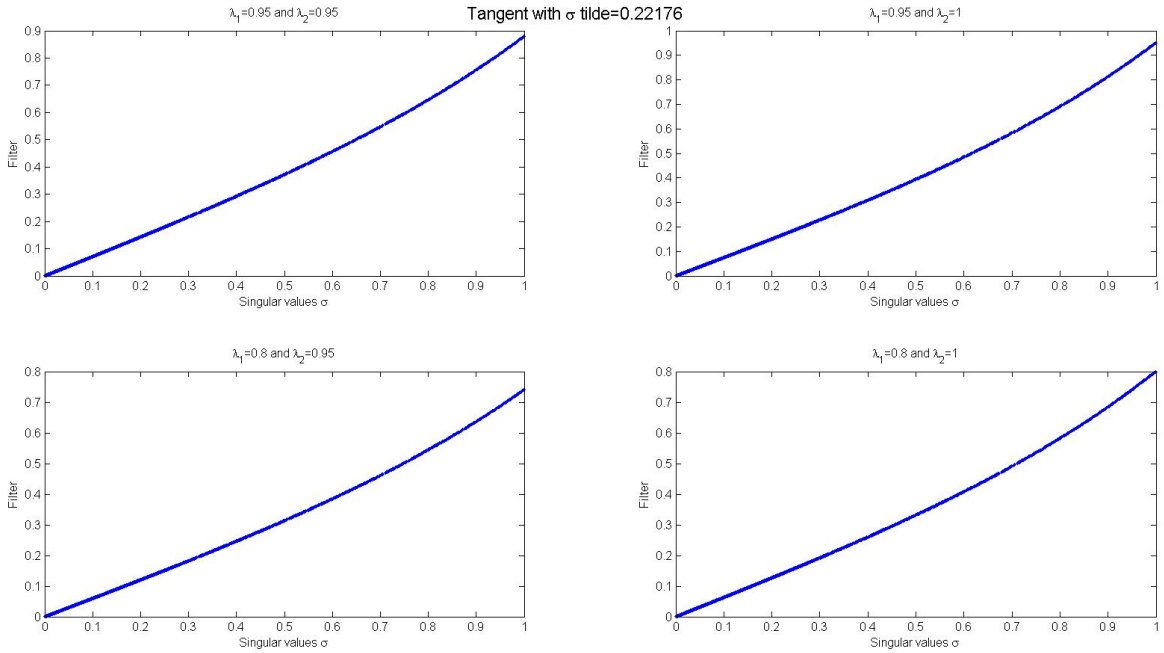

Figure 5.9: Tangent Filter.

This filter, shown in Figure 5.9, depends on two continuous parameters λ_1 and λ_2, $0 \leq \lambda_1 \leq \lambda_2 \leq 1$.

5.7 The cubic spline filter

Another idea is to create a filter using a cubic spline. A *cubic spline* $s(y)$ *with knots* $y_1 < y_2 < \cdots < y_p$ defined on $[y_1, y_p]$ has the properties:

- It is a cubic polynomial in each interval $[y_i, y_{i+1}]$ for $i = 1, \cdots, p-1$.
- The function, its 1st derivative, and its 2nd derivative are continuous.
- The 3rd derivative can be discontinous only at the knots.

Define $h_{i+1} = y_{i+1} - y_i$ for $i = 1, \cdots, p-1$. In the interval $[y_i, y_{i+1}]$, we define the spline as

$$s_{i+1}(y) = m_i \frac{(y_{i+1} - y)^3}{6h_{i+1}} + m_{i+1} \frac{(y - y_i)^3}{6h_{i+1}} + a_i(y - y_i) + b_i \tag{5.16}$$

for some constants m_i, m_{i+1}, a_i, b_i.

So in total, in order to determine the spline, we need to find $a_1, \ldots, a_{p-1}$, $b_1, \ldots, b_{p-1}$, and $m_1, \ldots, m_p$. That means our total number of parameters to be determined is $3p - 2$.

Our constraints are continuity of the function and the derivatives at the knots.

Continuity of $s(y)$

$$s_{i+1}(y_i) = m_i \frac{h_{i+1}^2}{6} + m_{i+1}0 + a_i 0 + b_i$$

and

$$s_i(y_i) = m_{i-1}0 + m_i \frac{h_i^2}{6} + a_{i-1}h_i + b_{i-1}.$$

By setting these two expressions equal to each other, we have that:

$$s_{i+1}(y_i) = m_i \frac{h_{i+1}^2}{6} + b_i = m_i \frac{h_i^2}{6} + a_{i-1}(h_i) + b_{i-1} = s_i(y_i).$$

Doing this for every intermediate knot, this gives $p - 2$ constraints.

Continuity of $s'(y)$

We follow the same procedure for the continuity of the first derivative. We have:

$$s'_{i+1}(y) = -m_i \frac{(y_{i+1} - y)^2}{2h_{i+1}} + m_{i+1} \frac{(y - y_i)^2}{2h_{i+1}} + a_i \tag{5.17}$$

Now,

$$s'_{i+1}(y_i) = -m_i \frac{h_{i+1}}{2} + m_{i+1}0 + a_i$$

and

$$s'_i(y_i) = -m_{i-1}0 + m_i \frac{h_i}{2} + a_{i-1}.$$

By setting these two expressions equal to each other, we have that:

$$s'_{i+1}(y_i) = -m_i \frac{h_{i+1}}{2} + a_i = +m_i \frac{h_i}{2} + a_{i-1} = s'_i(y_i).$$

and again, we have $p-2$ constraints.

Continuity of $s''(y)$

For the second derivative, we have:

$$s''_{i+1}(y) = m_i \frac{(y_{i+1} - y)}{h_{i+1}} + m_{i+1} \frac{(y - y_i)}{h_{i+1}}. \tag{5.18}$$

Now,

$$s''_{i+1}(y_i) = m_i + m_{i+1}0$$

and

$$s''_i(y_i) = m_{i-1}0 + m_i.$$

By setting these two expressions equal to each other, we have that:

$$s''_{i+1}(y_i) = m_i = s''_i(y_i).$$

Thus our definition of the spline ensures this continuity.

So in total, we have $2p-4$ conditions.

If we set $m_1 = m_p = 0$, then we have a total of $2p-2$ conditions and the p remaining free parameters are the parameters $\boldsymbol{\lambda}$ of the spline filter.

We choose $\boldsymbol{\lambda} = [m_2, \cdots, m_{p-1}, a_1, b_1]$. Then we can see that, given values of these parameters, the other parameters can be determined from

$$a_i = a_{i-1} + m_i \frac{h_i + h_{i+1}}{2}$$

and

$$b_i = b_{i-1} + a_{i-1} h_i + m_i \frac{h_i^2 - h_{i+1}^2}{6}.$$

Instead of setting $m_1 = m_p = 0$, we can set $s(y_1) = 0$ and/or $s(y_p) = 1$.

When $s(y_1) = 0$, we get that

$$m_1 = -b_1 \frac{6}{h_2^2}.$$

Since b_1 is a free parameter and y_1 and y_2 are known, we can find m_1 and still use forward substitution as described above.

In the case when $s(y_p) = 1$, we get that

$$m_p = \frac{6(1 - b_{p-1} - a_{p-1} h_p)}{(h_p)^2}.$$

Since m_p is not needed to compute any of the a_i or b_i, we can still use forward substitution.

Figures 5.10 – 5.13 show the spline filter with $p = 3$, 5, 10, and 30 knots. The plots on the left show the spline filter when the knots are equally spaced on a linear scale, whereas the plots on the right have the knots equally spaced on a logarithmic scale.

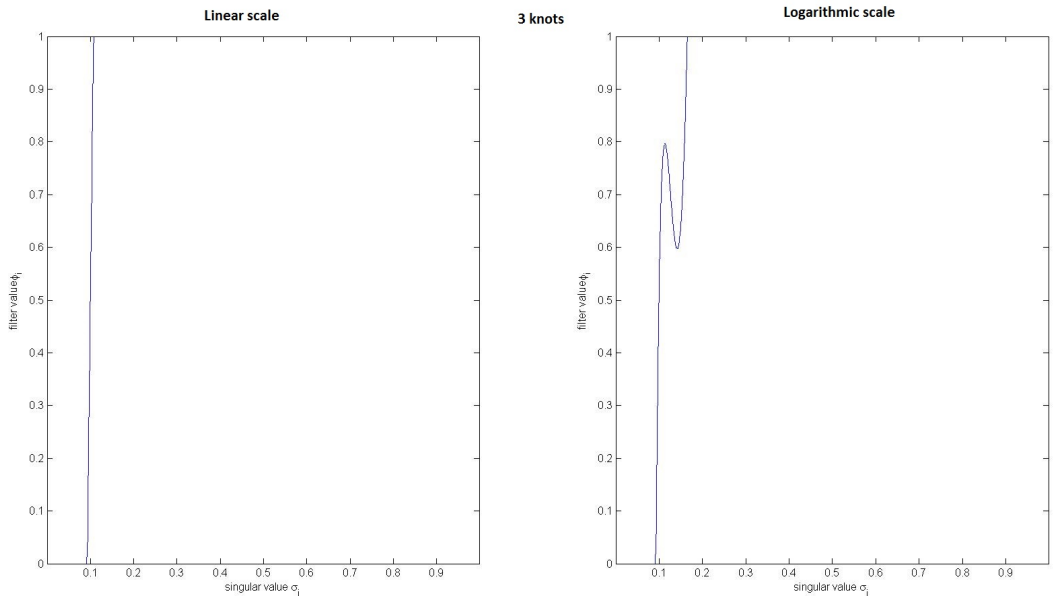

Figure 5.10: Spline filter with 3 knots.

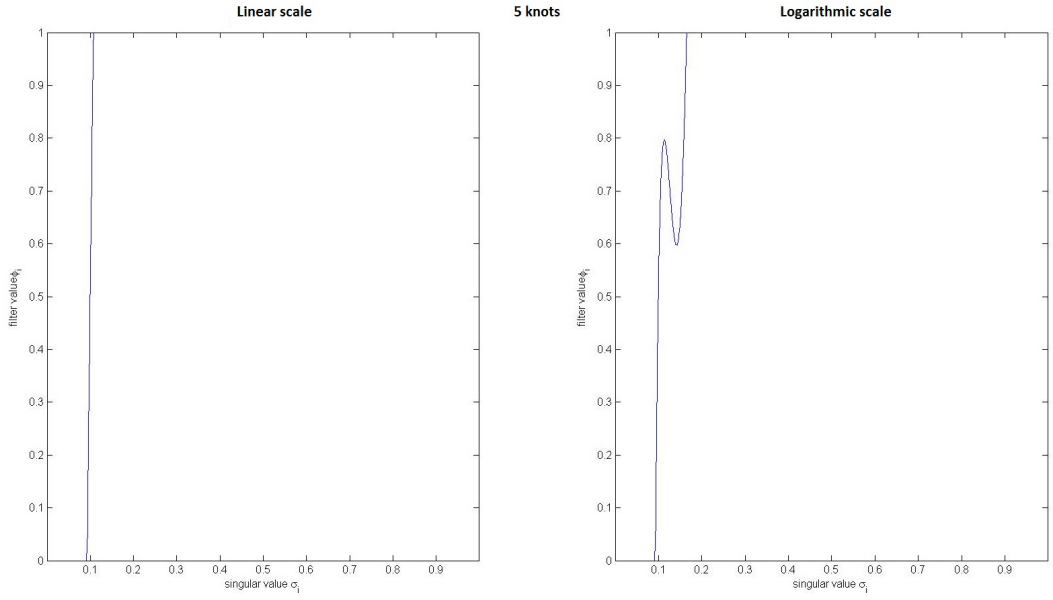

Figure 5.11: Spline filter with 5 knots.

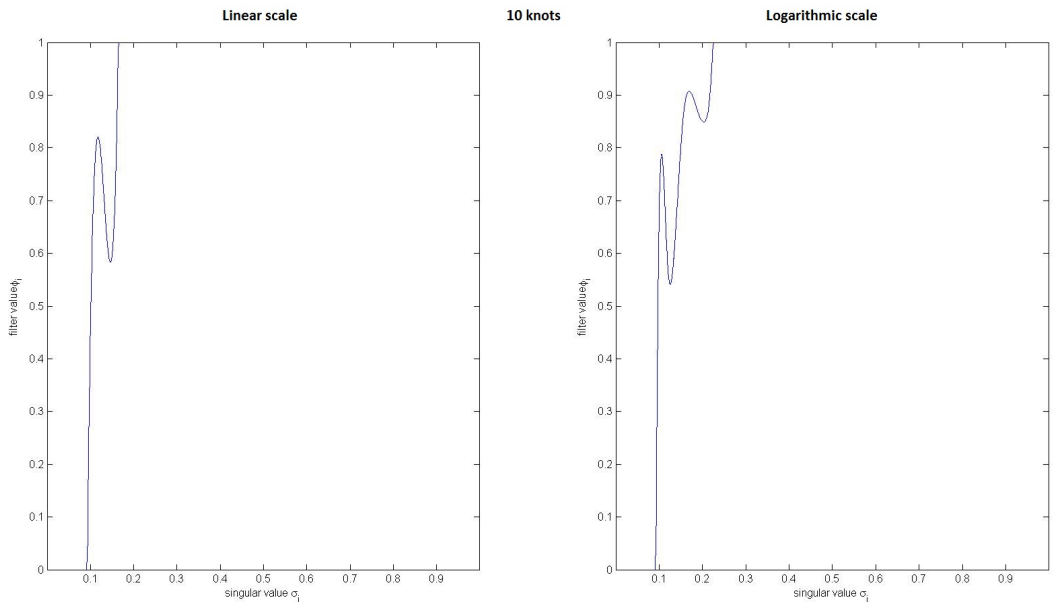

Figure 5.12: Spline filter with 10 knots.

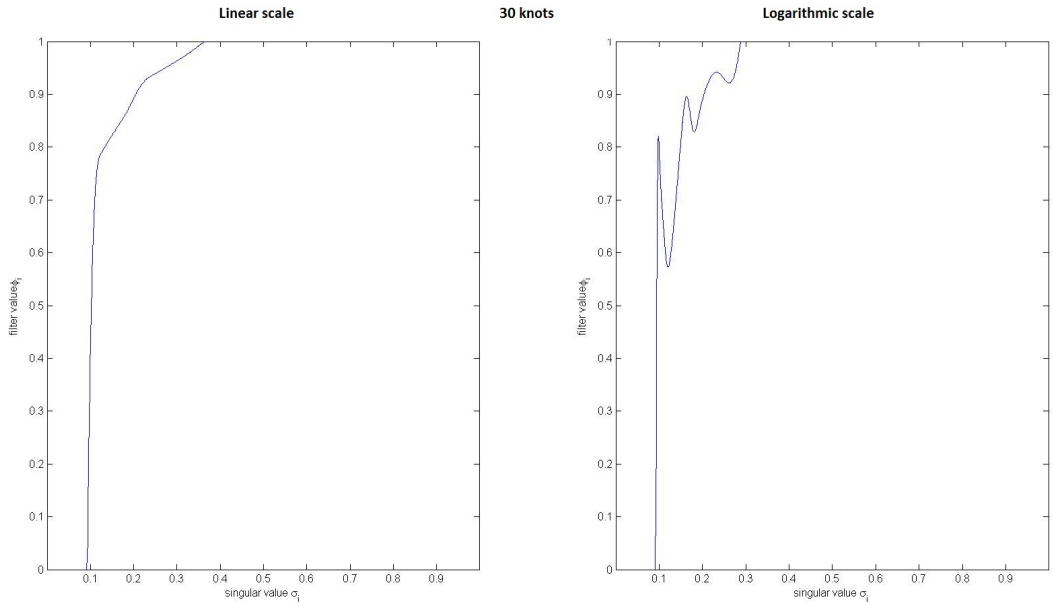

Figure 5.13: Spline filter with 30 knots.

So, using Algorithm 4, we can determine the optimal $\boldsymbol{\lambda}$ for the filter $\phi_{\boldsymbol{\lambda}}(\sigma) = s(\sigma)$. We can add the condition that $\phi_{\boldsymbol{\lambda}}(\sigma) \in [0, 1]$, for $\sigma \in [\sigma_n, \sigma_1]$, which holds for the other filters discussed in this chapter. This could be done as a constraint on the spline, but we chose to impose it afterward, setting $\phi_{\boldsymbol{\lambda}}$ to 0 or 1 if $s(\sigma_i)$ falls outside $[0, 1]$.

Based on our experience, a small number of knots ($p = 3$, 4, or 5), with $y_1 = \sigma_n$ and $y_p = \sigma_1$ (or alternatively, $y_1 = \log(\sigma_n)$ and $y_p = \log(\sigma_1)$), is sufficient for good performance and keeps the computational cost low.

5.8 Conclusions

In this chapter, we introduced several new spectral filters. Some of them are truncated versions of known filters, using the Picard Parameter as estimated in Chapter 2, and some others are completely new filters that again include the Picard Parameter but could be used even if someone does not want to crop them using it.

In Chapter 3, we introduced the TSVDk filter that is a variation of the TSVD filter when the Discrete Picard condition is satisfied. Assuming that the Discrete Picard condition is satisfied in our problem and that the Picard parameter has been computed, we introduced the Truncated Tikhonov filter, a variation of the Tikhonov filter. In addition, we modified the TSCM method into one that takes into account the Picard parameter. A Hybrid filter that combines the TSVDk and the Truncated Tikhonov filter was introduced next to show that we can combine filters to obtain some of the good properties of both filters. In addition, we created a continuous

alternative to the discrete TSVDk filter which gives insight on how methods can be treated as continuous. Two continuous Heaviside filters and a tangent filter were also introduced. Finally, we defined and presented the cubic spline filter with a general number of knots. These knots could be spaced either linearly or logarithmically.

In the next Chapter we implement those filters with three different methods for determining the parameters $\boldsymbol{\lambda}$, our new SOF method as well as GCV and the DP. The results show that these filters are good filters and that in some cases they outperform some of the already established filters.

Chapter 6: Experimental Evaluation of the Methods

6.1 Introduction

We discussed in Chapter 5 several filters that could be used to compute a solution $\mathbf{x}_{filt}$ to the inverse problem $\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$ when we know the vector $\mathbf{b}$ and the matrix $\mathbf{A}$. In this chapter, we consider $\mathbf{b}$ to be the vector of an image that is contaminated by the known blur $\mathbf{A}$ and unknown noise $\mathbf{e}$. We will compare the performance of the usual TSVD and Tikhonov (TIK) filters to some of the ones that we developed. The filters will be denoted as follows:

- TIKk: The truncated Tikhonov filter developed in Section 5.2 using the Picard parameter estimation developed in Chapter 2,
- TSVDk: The TSVD filter developed in Chapter 3 using the Picard parameter estimation developed in Chapter 2,
- TSCMk: The TSCM filter presented in Section 5.3, truncated using the Picard parameter estimation developed in Chapter 2,
- HYBR: The Hybrid filter developed in Section 5.4 that combines TSVDk and TIKk,

- SPL_lin: The spline filter developed in Section 5.7, with linear spacing of the knots,

- SPL_log: The spline filter developed in Section 5.7, with logarithmic spacing of the knots.

We use three methods to find the optimal parameter $\boldsymbol{\lambda}$ for each of the filters. The name of the method will be denoted as a prefix.

- SOF: Our method for Statistically Optimizing the Filter (SOF) developed in Chapters 3 and 4

- GCV: Generalized Cross Validation, described in Section 1.5

- DP: Discrepancy Principle, described in Section 1.5

For example, using the above notation, SOF_HYBR will denote the Hybrid filter when computed using our SOF method.

The results are presented in different ways. Tables report the relative errors, the average error per pixel, and the parameter $\boldsymbol{\lambda}$ for each filter. Figures show the images of the solutions and the errors, as well as the optimal filters. The error images are scaled with white denoting values of at least 0.2 standard deviations. The better the solution is, the darker the error image should be.

6.2 Numerical Results

In this section, we present a few different examples, using the Barbara, the Pirate, and the UMD images. The blurring matrix that is used is a separable

Gaussian blur as described in Appendix A and is the same for all images of the same size. The additive noise is always of mean zero and with standard deviation that varies and is mentioned in each example since it is important.

6.2.1 Example 1

This experiment is performed using a 64×64 resolution Pirate image. The noise that is added is of standard deviation $s = 10$. The signal to noise ratio that comes from that is SNR=10.910. The Picard parameter that is computed and used for the methods that require it is $k = 767$ that results into an estimated standard deviation of exp_stdev = 10.723.

Tables 6.1 and 6.2 present the results. Our methods perform comparably to the Full-TSVD method that is computed using all available singular values. That means that the Picard parameter has been computed correctly and we didn't lose much information. Of the new methods, the Hybrid method using GCV to find the solution is the best. The relative error of the HYBR filter is very close to that of the TSVD which seems to be the best filter according to our SOF method. It is interesting to see that whereas the average error of the TSVD is larger that the one of the HYBR when computed with the GCV method, the relative error behaves the opposite way. The TSVD and TSVDk with the Discrepancy Principle behave the worst. It is also interesting to see that the two Spline methods behave very similarly, and that the Tikhonov filter is a lot worse than the rest of the filters when computed with the GCV method but better that the rest when we use the

Discrepancy Principle.

From Figures 6.1 and 6.2, we can see that the Discrepancy Principle method oversmooths the image and we can verify the results we mentioned above.

From the error Figures 6.3 and 6.4 we can see that the Discrepancy Principle method fails to restore the edges of the image. We can also see that the TSVD filter with the SOF and GCV methods gives the darkest error images, i.e., error images close to zero. In general, the filters have more error at the edges of the pirate.

Comparing the filter figures (Figures 6.5 and 6.6) we can draw conclusions about the weight we should give to each singular value. For example, looking at the TSVD filter and the others as well and knowing that the errors are smaller for the SOF and GCV methods, we can deduce that the large singular values should be given a larger weight than what the Discrepancy Principle method assigns but we shouldn't give much weight on smaller singular values like the GCV-TIK suggests.

Table 6.1: Example 1 results.

Method	Relative error	Average error	Lambda
SOF-TSVD	1.37e-001	1.29e+001	1.00e+003
GCV-TSVD	1.37e-001	1.29e+001	1.00e+003
DP-TSVD	2.57e-001	2.26e+001	1.52e+002
SOF-TIK	1.66e-001	1.57e+001	1.06e-001
GCV-TIK	2.29e-001	2.18e+001	1.72e-002
DP-TIK	2.12e-001	2.08e+001	1.94e-001
SOF-TIKk	1.42e-001	1.30e+001	9.82e-003
GCV-TIKk	1.42e-001	1.31e+001	3.67e-003
DP-TIKk	2.16e-001	2.09e+001	1.87e-001
SOF-TSVDk	1.43e-001	1.31e+001	7.66e+002
GCV-TSVDk	1.43e-001	1.31e+001	7.66e+002
DP-TSVDk	2.57e-001	2.26e+001	1.52e+002
SOF-TSCM	1.45e-001	1.33e+001	1.52e+001
GCV-TSCM	1.45e-001	1.33e+001	1.50e+001
DP-TSCM	2.47e-001	2.26e+001	1.64e+002
SOF-HYBR	1.41e-001	1.29e+001	[4.82e+002 , 6.57e-002]
GCV-HYBR	1.41e-001	1.28e+001	[4.16e+002 , 5.48e-002]
DP-HYBR	2.30e-001	2.07e+001	[1.00e+001 , 5.35e-001]

Table 6.2: Example 1 results, continued.

Method	Relative error	Average error	Lambda
SOF-SPL_lin	1.43e-001	1.30e+001	$\begin{bmatrix} -8.67e-001; 1.58e+001 \\ -3.17e+000; -1.78e-001 \\ 9.614154e-001 \end{bmatrix}$
GCV-SPL_lin	1.43e-001	1.30e+001	$\begin{bmatrix} 2.70e+001; -1.12e+001 \\ 3.74e+000; -2.55e+000 \\ 1.167968e+000 \end{bmatrix}$
DP-SPL_lin	2.42e-001	2.10e+001	$\begin{bmatrix} 1.70e-001; 8.22e-001 \\ 2.34e-001; 8.59e-001 \\ 4.238888e-002 \end{bmatrix}$
SOF-SPL_log	1.43e-001	1.29e+001	$\begin{bmatrix} 3.25e-001; 2.78e-004 \\ 1.52e-001; 1.72e-001 \\ 8.324527e-001 \end{bmatrix}$
GCV-SPL_log	1.43e-001	1.29e+001	$\begin{bmatrix} 2.78e-001; -5.40e-001 \\ -5.97e-001; 2.79e-001 \\ 7.974489e-001 \end{bmatrix}$
DP-SPL_log	2.44e-001	2.10e+001	$\begin{bmatrix} 9.57e-001; 5.73e-001 \\ 7.16e-001; 2.87e-001 \\ 2.724047e-002 \end{bmatrix}$

Figure 6.1: Example 1: Computed solutions.

Figure 6.2: Example 1: Computed solutions, continued.

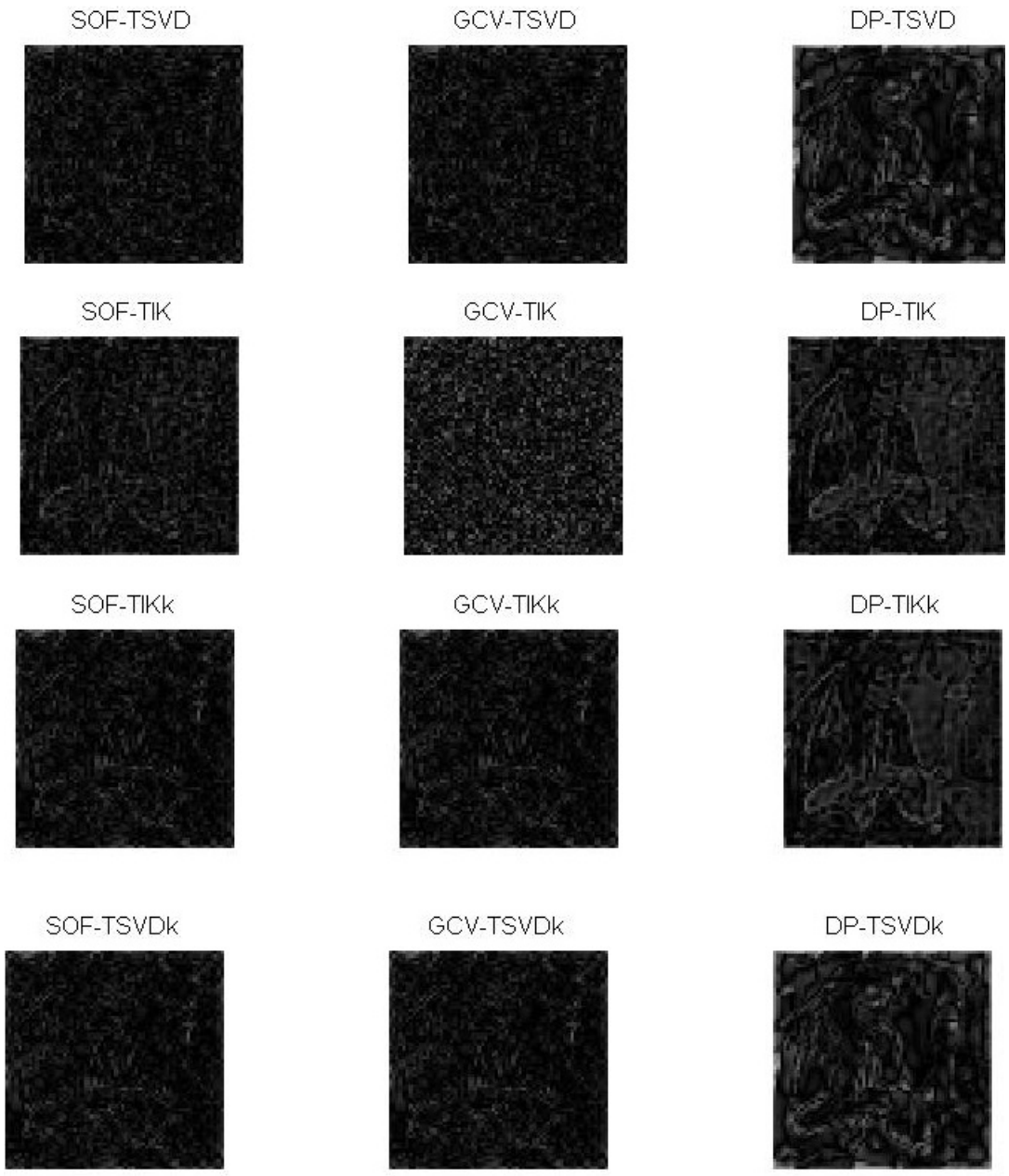

Figure 6.3: Example 1: Errors.

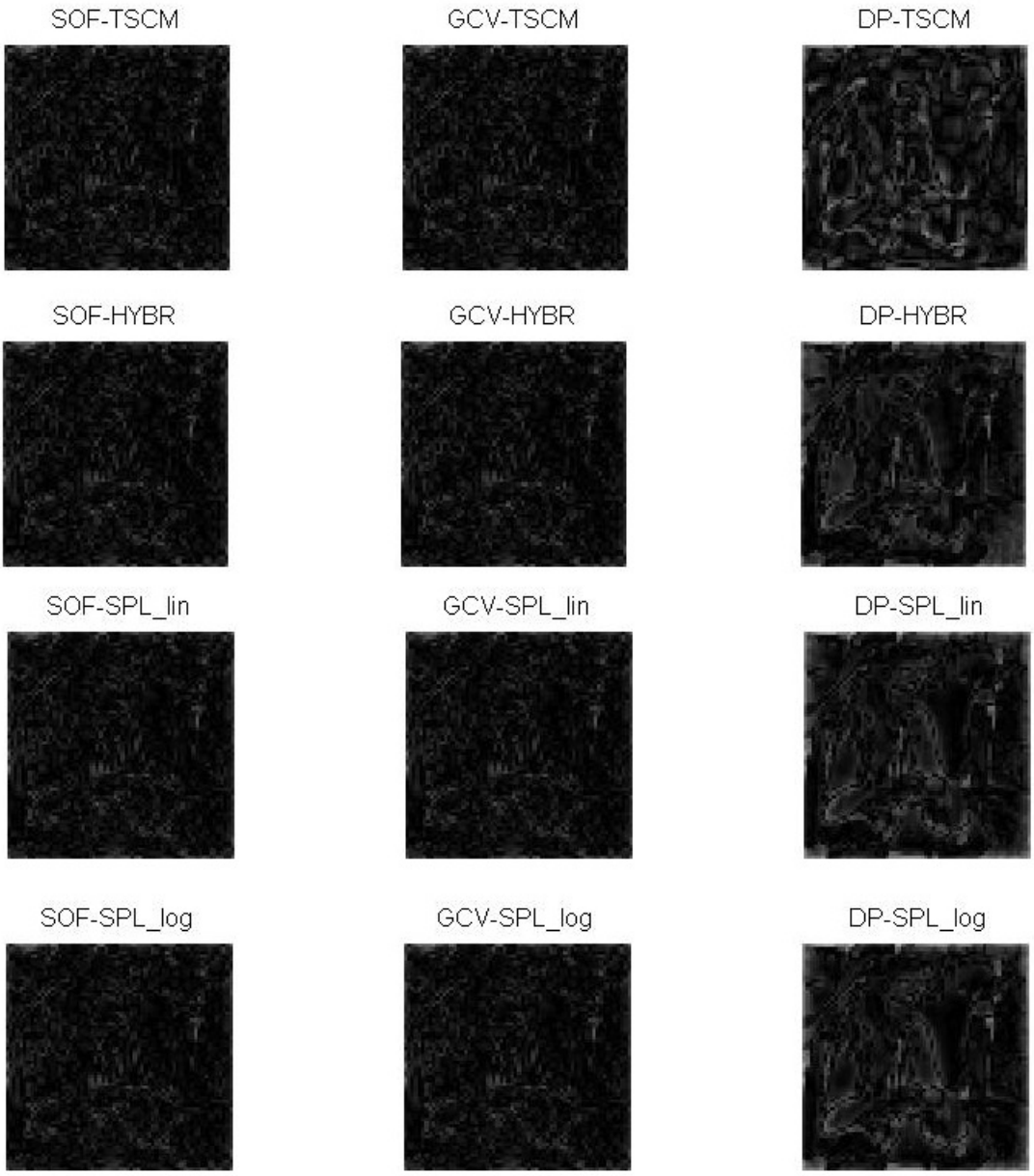

Figure 6.4: Example 1: Errors, continued.

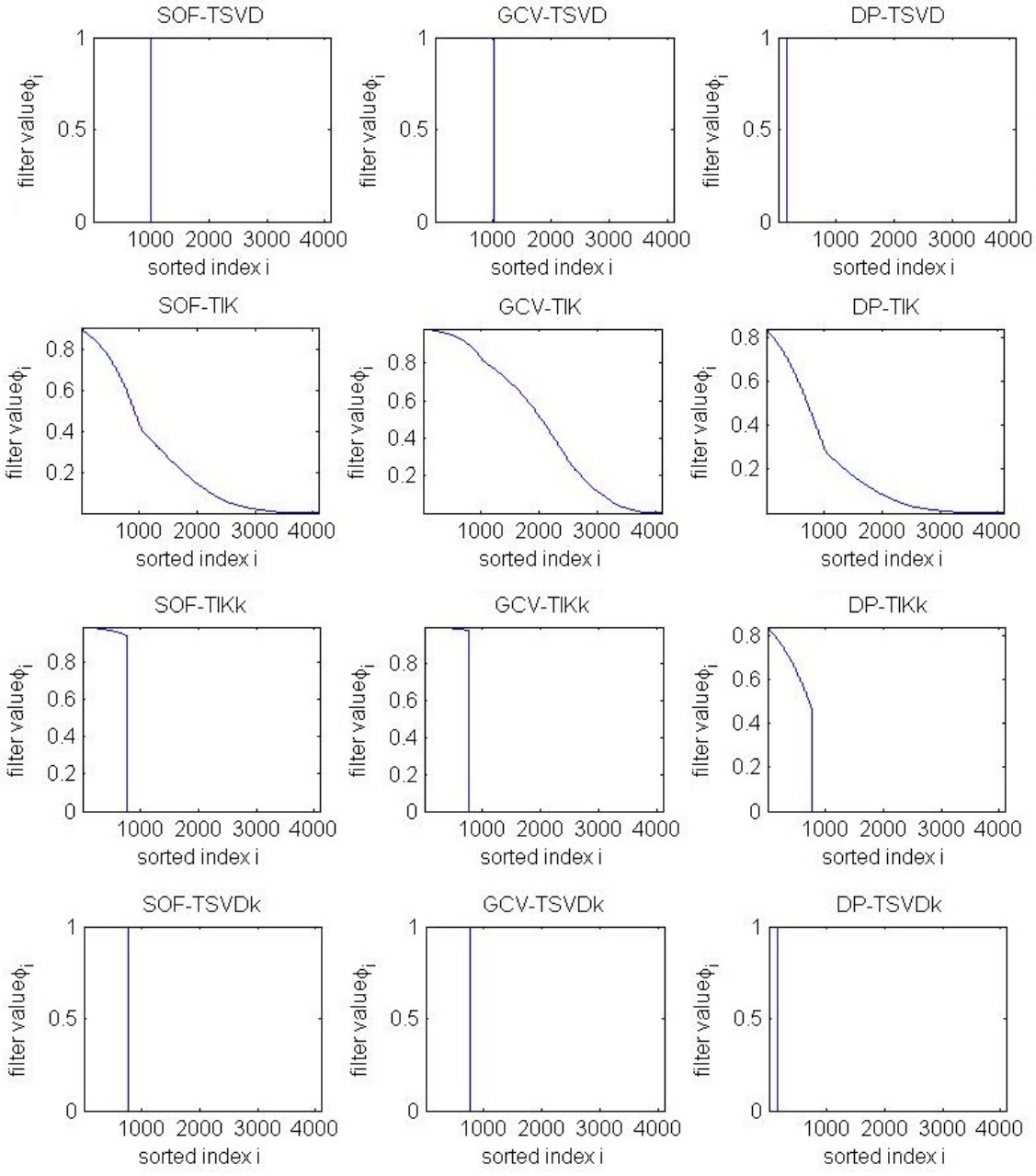

Figure 6.5: Example 1: Filters.

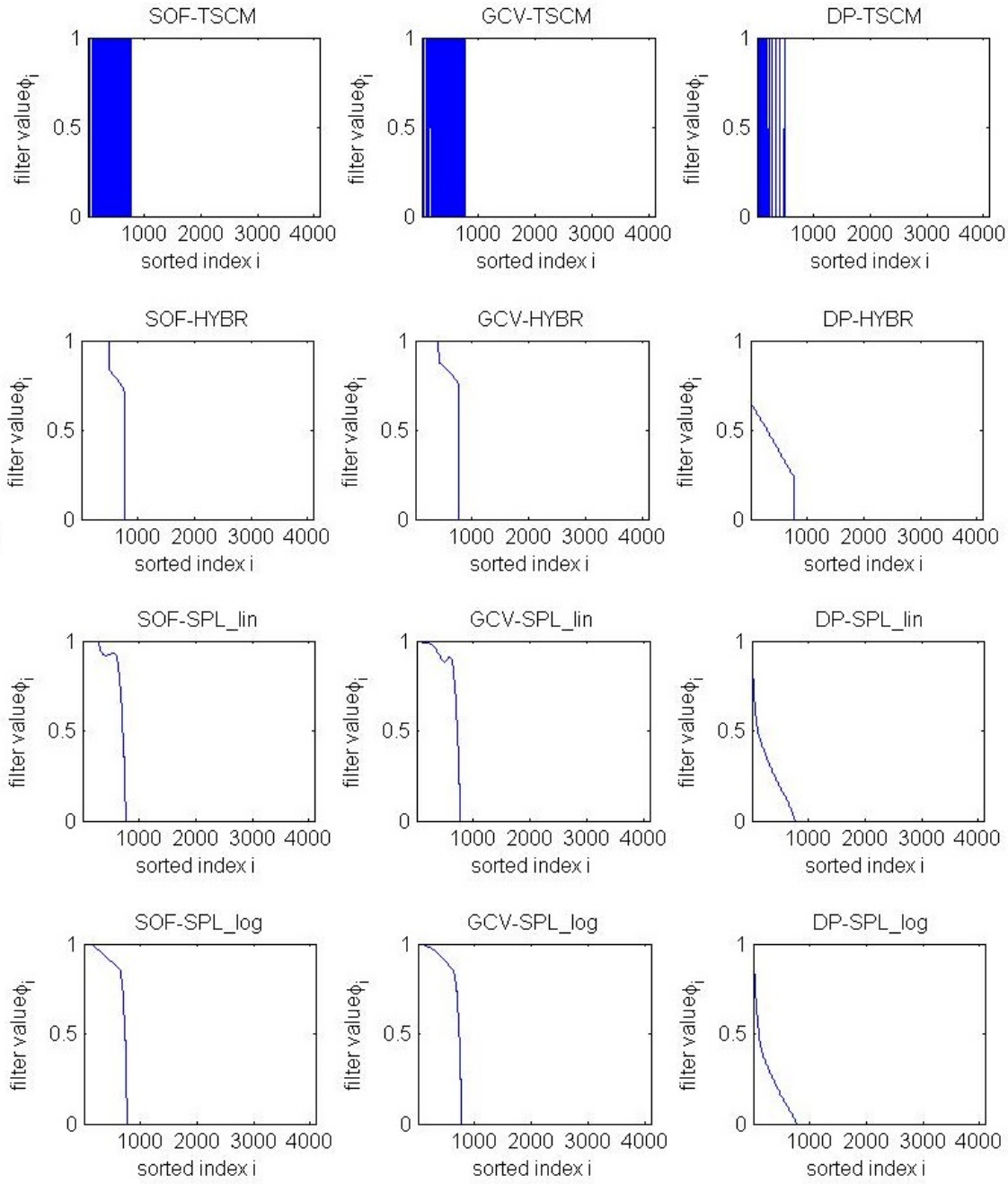

Figure 6.6: Example 1: Filters, continued.

6.2.2 Example 2

This experiment is performed using a 64×64 resolution Pirate image. The noise that is added is of standard deviation $s = 1$. The signal to noise ratio that comes from that is SNR=108.943. The Picard Parameter that is computed and used for the methods that require it is $k = 2770$ that results into an estimated standard deviation of exp_stdev = 1.073.

Tables 6.3 and 6.4 present the results. In general, with the smaller standard deviation of the noise, the results are better as expected by an order of magnitude.

Our methods perform comparably to the TSVD and the TIK filter that are computed using all available singular values. That means that the Picard parameter has been computed appropriately and we didn't lose much information. The best results seem to come from the SOF-TIK method, whereas the DP-TSVD and DP-TSVDk give the worst results. Once again, for each method, our method gives similar results with those using the GCV method and better than those using the Discrepancy Principle method. Looking at the two spline filters with the different spacing, we can see that one filter does not perform better than the other with all three methods.

Figures 6.7 and 6.8 show the restored images using the above methods. Figures 6.9 and 6.10 show the error images in an appropriate scale and Figures 6.11 and 6.12 show the optimal filters.

The error images now are scaled based on the smaller standard deviation, so we cannot compare them to the ones from the previous example based on color. We

can however see that here, the edges of the image are not as clear as they were with the higher standard deviation of the noise and again we can mostly see something at the DP method solutions and not so much in the SOF or GCV results.

From Figures 6.11 and 6.12, we see that fewer singular values are associated with noise and so we need to give higher weights to more than we did in the example in Section 6.2.1. Comparing the filters of the SOF-TSVD and the SOF-TSVDk, we can deduct that the Picard Parameter has been computed properly.

Table 6.3: Example 2 results.

Method	Relative error	Average error	Lambda
SOF-TSVD	6.99e-002	6.51e+000	2.41e+003
GCV-TSVD	7.02e-002	6.53e+000	2.38e+003
DP-TSVD	1.08e-001	9.84e+000	1.01e+003
SOF-TIK	6.53e-002	6.02e+000	8.67e-003
GCV-TIK	7.55e-002	7.17e+000	1.30e-003
DP-TIK	7.17e-002	6.53e+000	1.57e-002
SOF-TIKk	6.69e-002	6.17e+000	4.98e-003
GCV-TIKk	6.88e-002	6.42e+000	1.22e-003
DP-TIKk	7.40e-002	6.73e+000	1.56e-002
SOF-TSVDk	6.99e-002	6.51e+000	2.41e+003
GCV-TSVDk	7.02e-002	6.53e+000	2.38e+003
DP-TSVDk	1.08e-001	9.84e+000	1.01e+003
SOF-TSCM	7.24e-002	6.79e+000	1.52e+000
GCV-TSCM	7.20e-002	6.76e+000	1.35e+000
DP-TSCM	9.70e-002	8.85e+000	8.68e+000
SOF-HYBR	6.68e-002	6.15e+000	[1.03e+003 , 5.55e-003]
GCV-HYBR	6.67e-002	6.15e+000	[1.03e+003 , 4.47e-003]
DP-HYBR	8.76e-002	7.65e+000	[2.12e+002 , 4.54e-002]

Table 6.4: Example 2 results, continued.

Method	Relative error	Average error	Lambda
SOF-SPL_lin	6.76e-002	6.27e+000	$\begin{bmatrix} -5.14e+000; 8.52e+001 \\ -4.40e+001; -2.06e+000 \\ 1.764376e+000 \end{bmatrix}$
GCV-SPL_lin	6.70e-002	6.19e+000	$\begin{bmatrix} 4.93e+002; -7.71e+002 \\ 6.85e+002; -2.80e+001 \\ 4.113429e+000 \end{bmatrix}$
DP-SPL_lin	8.87e-002	7.77e+000	$\begin{bmatrix} 1.49e-001; 7.52e-001 \\ 2.19e-001; 8.78e-001 \\ 3.925452e-001 \end{bmatrix}$
SOF-SPL_log	6.66e-002	6.15e+000	$\begin{bmatrix} -5.38e-001; -3.37e-004 \\ 9.29e-001; 5.32e-001 \\ 4.736340e-001 \end{bmatrix}$
GCV-SPL_log	6.67e-002	6.16e+000	$\begin{bmatrix} -1.08e+000; 7.06e-001 \\ -7.20e-001; 7.42e-001 \\ 3.807012e-001 \end{bmatrix}$
DP-SPL_log	1.03e-001	9.11e+000	$\begin{bmatrix} 9.03e-001; 6.48e-001 \\ 7.46e-001; -1.32e-001 \\ 3.180766e-002 \end{bmatrix}$

Figure 6.7: Example 2: Computed solutions.

Figure 6.8: Example 2: Computed solutions, continued.

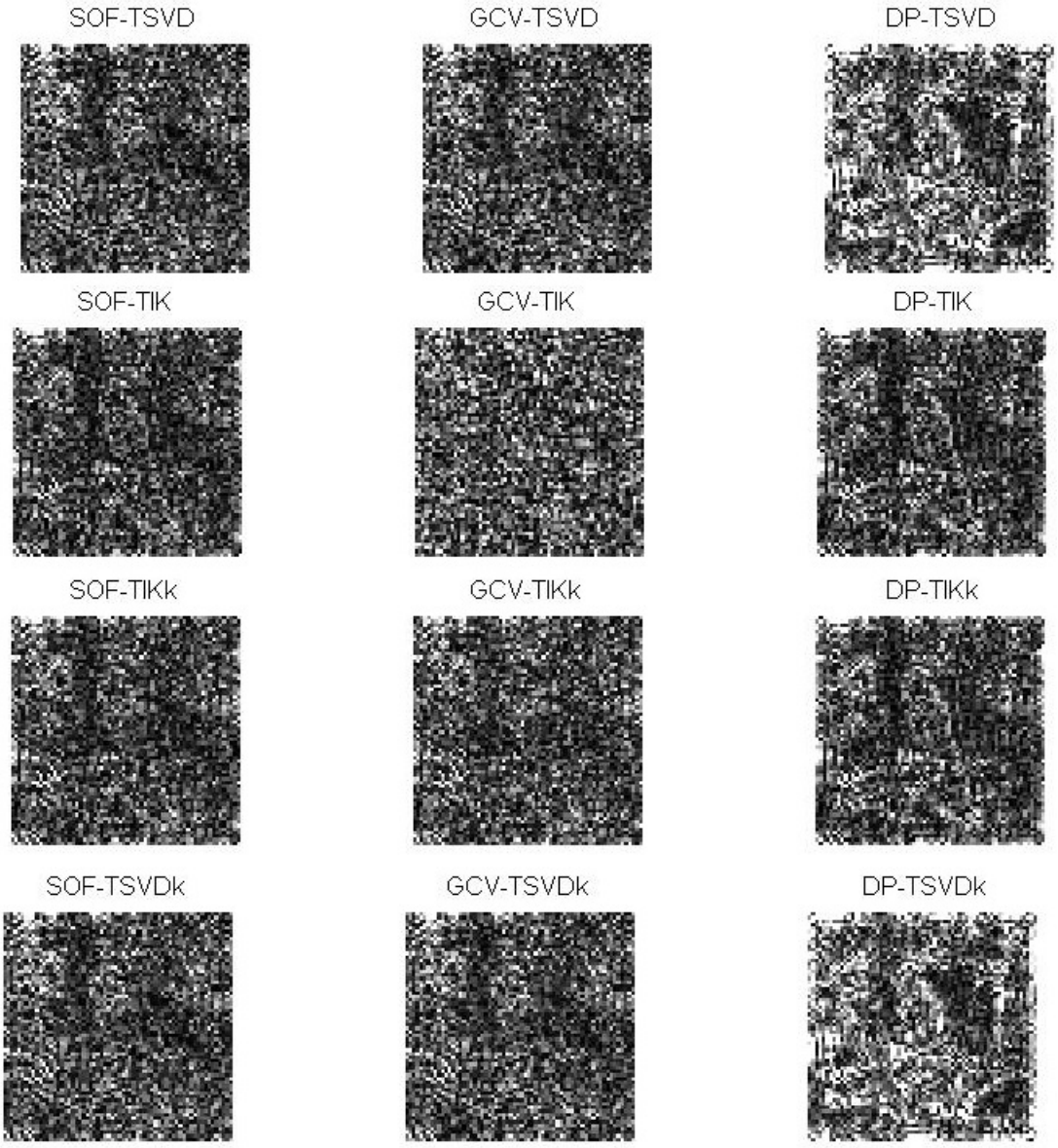

Figure 6.9: Example 2: Errors.

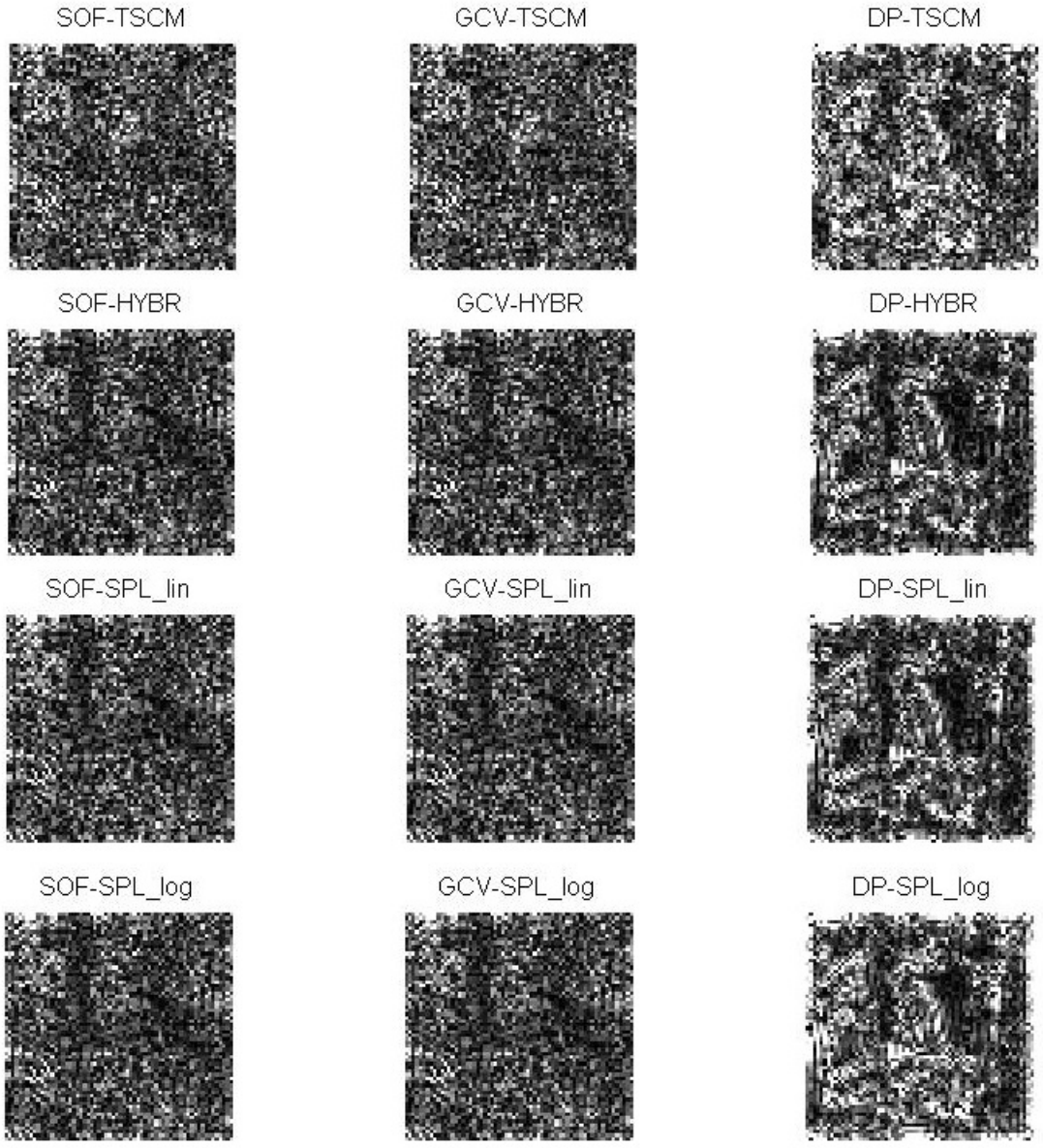

Figure 6.10: Example 2: Errors, continued.

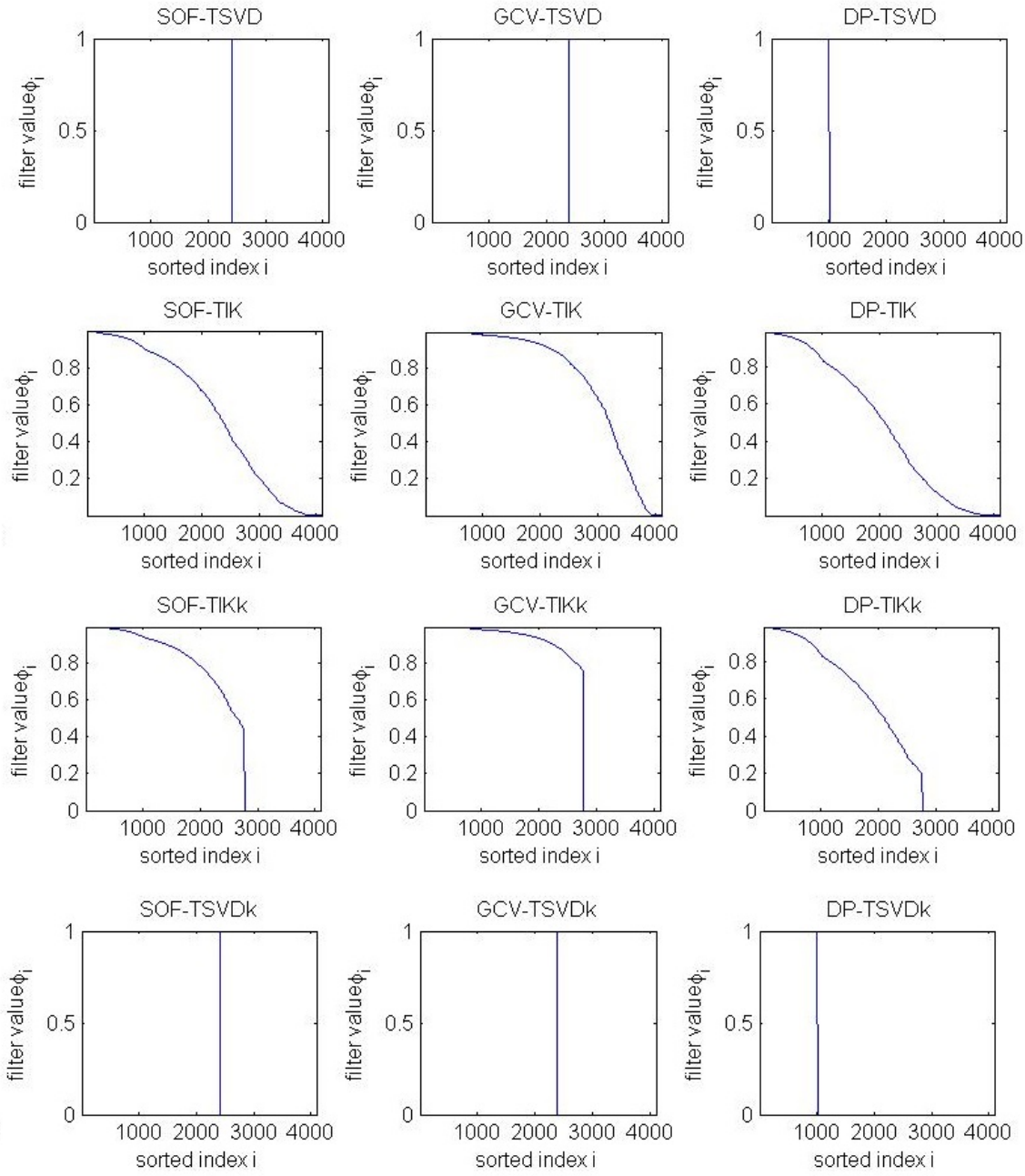

Figure 6.11: Example 2: Filters.

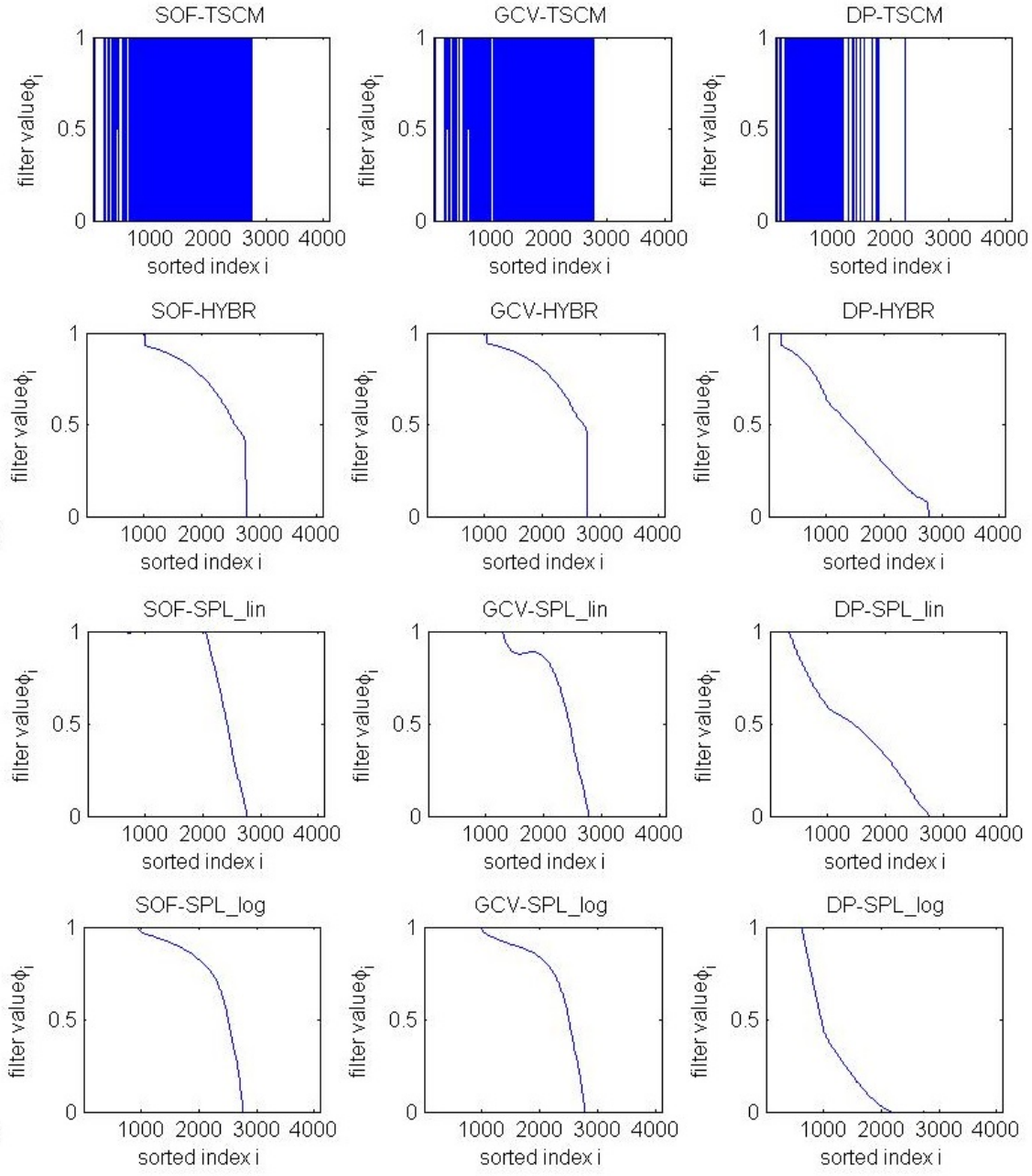

Figure 6.12: Example 2: Filters, continued.

6.2.3 Example 3

This experiment is performed using a 64×64 resolution Barbara image. The noise that is added is of standard deviation $s = 1$. The signal to noise ratio that comes from that is SNR=111.729. The Picard Parameter that is computed and used for the methods that require it is $k = 2394$ that results into an estimated standard deviation of exp_stdev = 1.147.

Tables 6.5 and 6.6 present the results.

Figures 6.13 and 6.14 show the restored images using the above methods. Figures 6.15 and 6.16 show the error images, and Figures 6.17 and 6.18 show the optimal filters.

In this example, the filter that gives the smaller relative error is the SOF-TIK. That verifies the fact that our method can give better results that the GCV and the Discrepancy Principle especially if we see that the GCV-TIK is the worst of all the filters when we use GCV. In this case GCV-TIK is worse than DP-TIK as well, something that we don't see in any of the other filters. The order of magnitude of the errors in this case is the same as the order of magnitude of the errors in the case of the Pirate image with the same parameters of the noise.

By looking at either the solution images or the error images, we can see that DP smooths the image by losing information near the edges of the object whereas the GCV-TIK solution is not as smooth as the images that are better solutions. This can be seen from the filter figures as well. The GCV-TIK gives larger weight to small singular values that are dominated by noise whereas the DP method ignores singular

values that hold important information about the image. It is also interesting to see how quickly the weight of the singular value changes in the case of the spline filter.

Figure 6.13: Example 3: Computed solutions.

Table 6.5: Example 3 results.

Method	Relative error	Average error	Lambda
SOF-TSVD	6.35e-002	6.10e+000	2.43e+003
GCV-TSVD	6.35e-002	6.10e+000	2.43e+003
DP-TSVD	1.01e-001	9.47e+000	9.43e+002
SOF-TIK	6.10e-002	5.75e+000	1.24e-002
GCV-TIK	7.09e-002	6.89e+000	1.33e-003
DP-TIK	6.47e-002	6.09e+000	1.69e-002
SOF-TIKk	6.24e-002	5.93e+000	4.43e-003
GCV-TIKk	6.29e-002	6.02e+000	9.01e-004
DP-TIKk	6.88e-002	6.45e+000	1.66e-002
SOF-TSVDk	6.40e-002	6.12e+000	2.38e+003
GCV-TSVDk	6.40e-002	6.12e+000	2.38e+003
DP-TSVDk	1.01e-001	9.47e+000	9.43e+002
SOF-TSCM	6.52e-002	6.25e+000	1.63e+000
GCV-TSCM	6.48e-002	6.22e+000	1.50e+000
DP-TSCM	9.00e-002	8.44e+000	9.68e+000
SOF-HYBR	6.23e-002	5.90e+000	[1.00e+003 , 5.40e-003]
GCV-HYBR	6.23e-002	5.89e+000	[1.00e+003 , 5.18e-003]
DP-HYBR	8.77e-002	7.96e+000	[5.56e+002 , 8.59e-002]

Table 6.6: Example 3 results, continued.

Method	Relative error	Average error	Lambda
SOF-SPL_lin	6.42e-002	6.12e+000	$-1.63e+000; 4.33e+001$ $-2.01e+001; -9.81e+000$ $6.352153e+000$
GCV-SPL_lin	6.56e-002	6.18e+000	$3.37e+002; -1.47e+002$ $2.14e+002; -2.14e+001$ $3.399844e+000$
DP-SPL_lin	8.45e-002	7.60e+000	$1.47e-001; 7.57e-001$ $2.21e-001; 8.74e-001$ $4.002758e-001$
SOF-SPL_log	6.36e-002	6.03e+000	$6.09e+001; 5.83e-005$ $-8.52e+000; -1.09e+001$ $4.013329e+000$
GCV-SPL_log	6.42e-002	6.05e+000	$7.59e-001; -4.62e-001$ $-1.60e-001; -1.35e-001$ $8.878763e-001$
DP-SPL_log	9.44e-002	8.59e+000	$9.12e-001; 6.14e-001$ $7.21e-001; 2.17e-002$ $3.078151e-002$

Figure 6.14: Example 3: Computed solutions, continued.

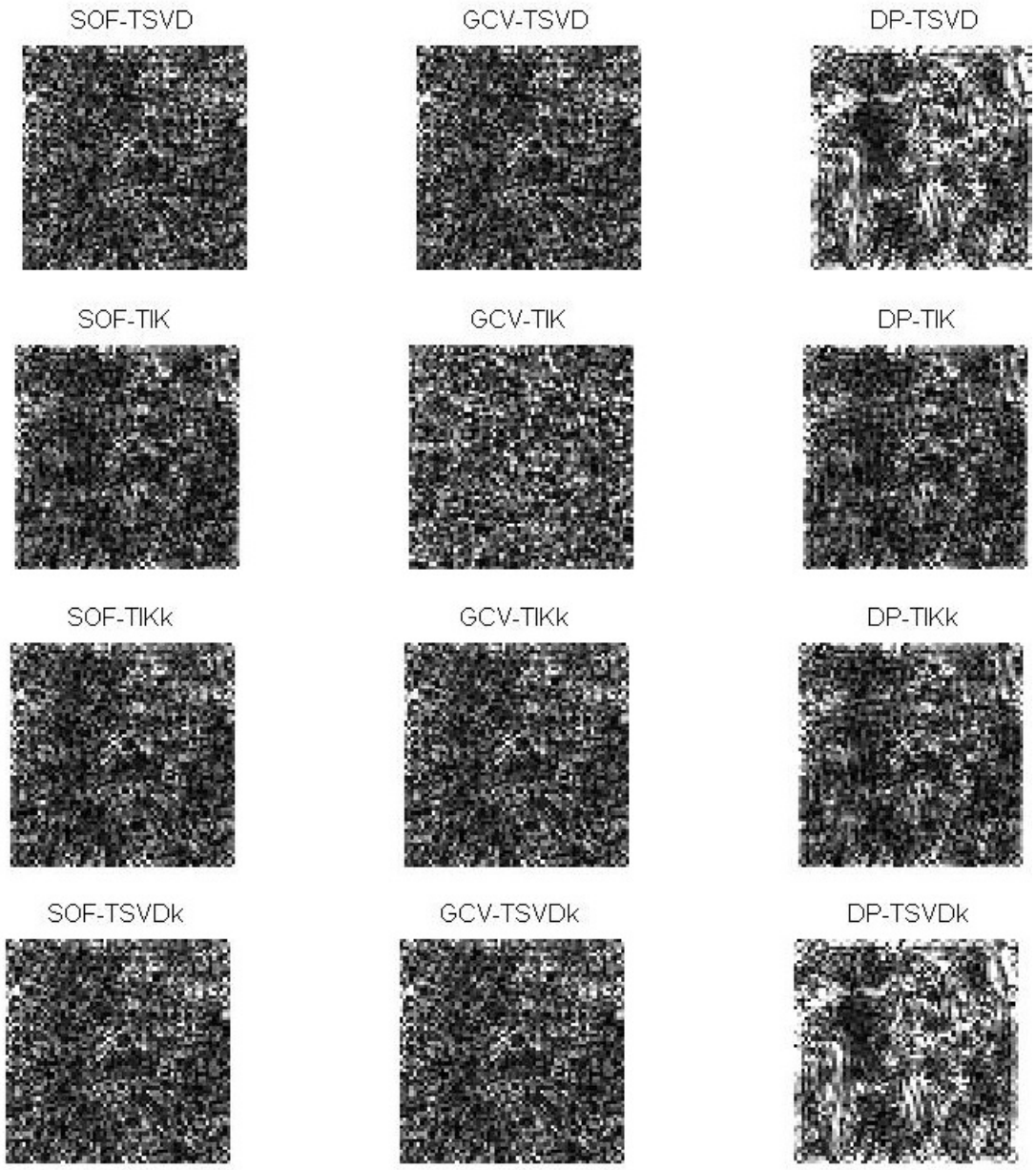

Figure 6.15: Example 3: Errors.

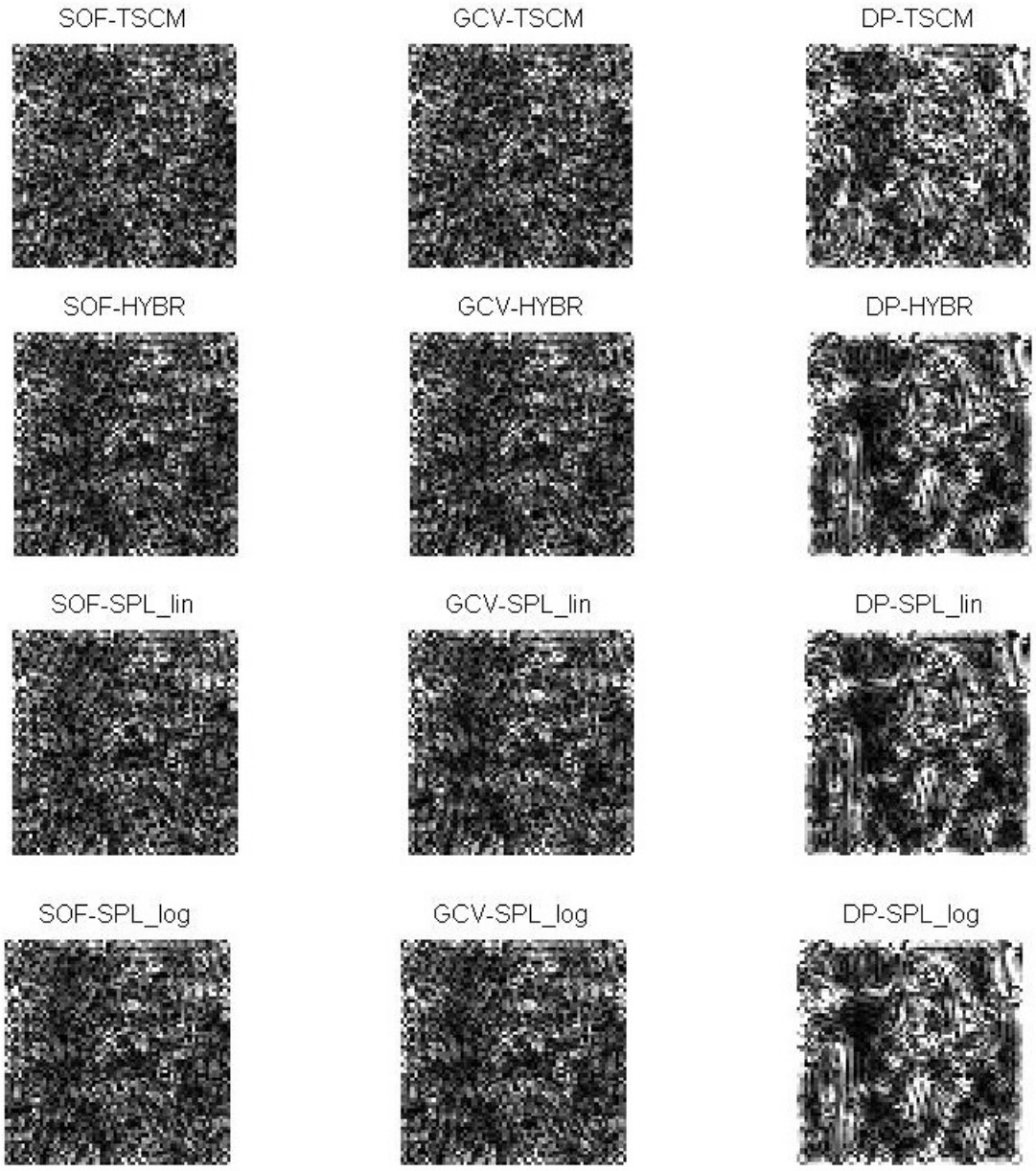

Figure 6.16: Example 3: Errors, continued.

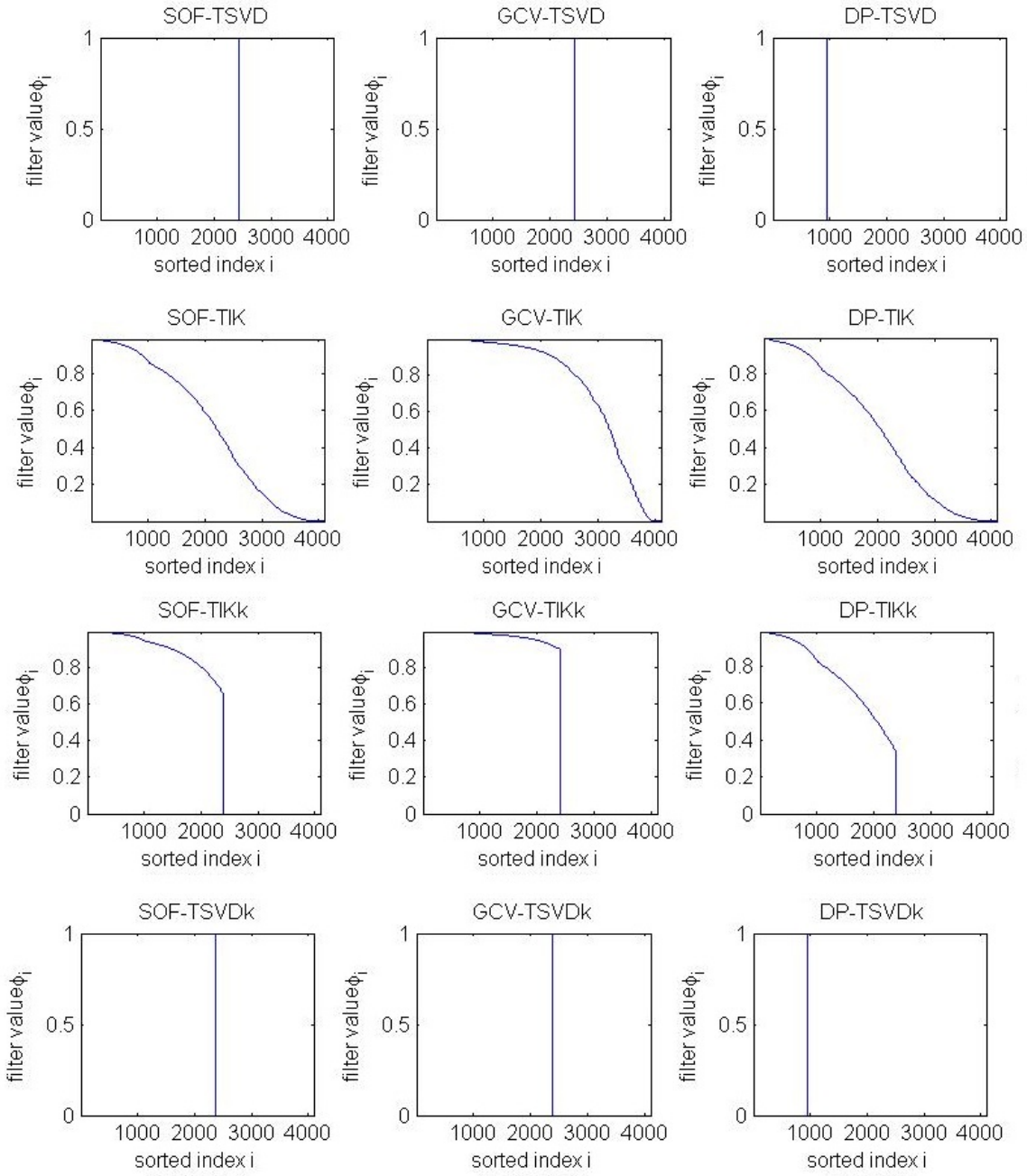

Figure 6.17: Example 3: Filters.

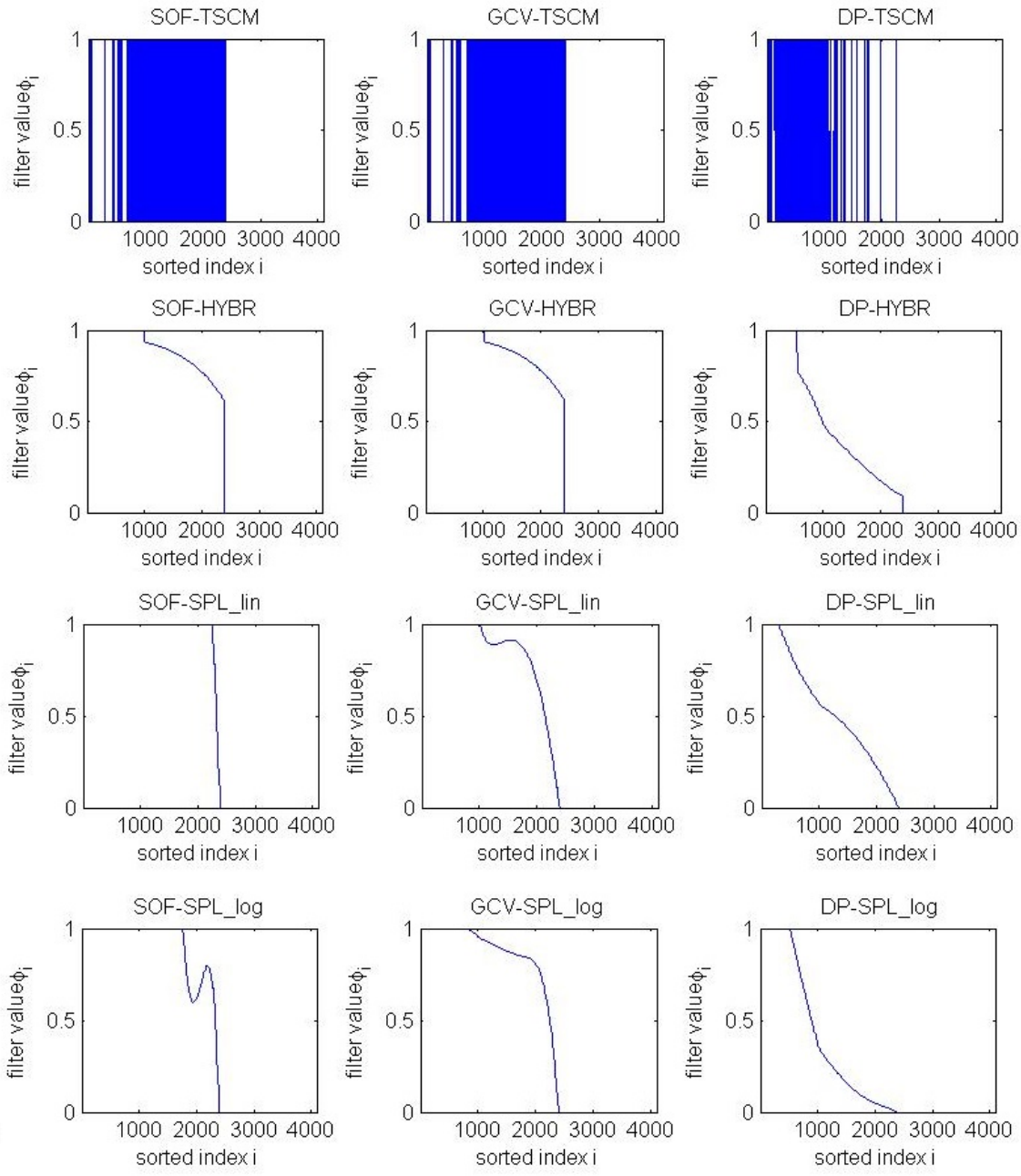

Figure 6.18: Example 3: Filters, continued.

6.2.4 Example 4

This experiment is performed using the 128×128 Pirate image. The noise that is added is of standard deviation $s = 1$. The signal to noise ratio that comes from that is SNR=110.033. The Picard Parameter that is computed and used for all the methods that require it is $k = 16013$ that results into 372 cropped eigenvalues and an estimated standard deviation of exp_stdev = 0.961.

Tables 6.7 and 6.8 present the results.

Figures 6.19 and 6.20 show the restored images using the above methods. Figures 6.21 and 6.22 show the error images, and Figures 6.23 and 6.24 show the optimal filters.

What is very interesting with this problem is the fact that the SOF method fails when paired with the TSVD. That happens because the method keeps all the singular values (see Figure 6.23). By giving the same weight in all singular values, we don't really deblur the image. The Tikhonov filter paired with the SOF method gives a very good estimate. When we combine the two methods though to the HYBR filter, the SOF method doesn't fail but it works better than each of the two individually. The same happens when we use the GCV method as well. Another interesting fact with this example is that the DP-TIK is better than the GCV-TIK and that the SOF-TSCM and GCV-TSCM perform worse than the other filters with the corresponding methods and the DP-TSCM. This is easily seen in the figures of the solution and of the error.

Our methods perform comparably to the TSVD method that is computed

Table 6.7: Example 4 results.

Method	Relative error	Average error	Lambda
SOF-TSVD	1.19e+001	1.14e+003	1.64e+004
GCV-TSVD	6.02e-002	5.60e+000	8.90e+003
DP-TSVD	8.58e-002	7.51e+000	3.96e+003
SOF-TIK	5.43e-002	5.10e+000	5.35e-003
GCV-TIK	7.10e-002	6.81e+000	1.34e-003
DP-TIK	5.69e-002	5.16e+000	1.44e-002
SOF-TIKk	5.43e-002	5.10e+000	5.35e-003
GCV-TIKk	7.10e-002	6.81e+000	1.34e-003
DP-TIKk	5.69e-002	5.16e+000	1.44e-002
SOF-TSVDk	5.89e-002	5.53e+000	9.97e+003
GCV-TSVDk	6.02e-002	5.60e+000	8.90e+003
DP-TSVDk	8.58e-002	7.51e+000	3.96e+003
SOF-TSCM	3.85e-001	3.69e+001	1.36e+000
GCV-TSCM	4.61e-001	4.42e+001	6.05e-001
DP-TSCM	7.81e-002	7.00e+000	7.64e+000
SOF-HYBR	5.37e-002	5.04e+000	[4.46e+003, 5.78e-003]
GCV-HYBR	5.39e-002	5.07e+000	[4.10e+003 , 5.47e-003]
DP-HYBR	8.44e-002	7.38e+000	[3.77e+003, 7.44e-001]

Table 6.8: Example 4 results, continued.

Method	Relative error	Average error	Lambda
SOF-SPL_lin	5.49e-002	5.16e+000	$\begin{bmatrix} 5.94e+001; 3.25e+003 \\ -3.80e+002; 6.33e+000 \\ -1.786134e-002 \end{bmatrix}$
GCV-SPL_lin	5.84e-002	5.54e+000	$\begin{bmatrix} -4.06e+000; -3.64e+000 \\ 2.13e+001; 1.27e+000 \\ 6.746483e-001 \end{bmatrix}$
DP-SPL_lin	6.51e-002	5.71e+000	$\begin{bmatrix} 1.43e-001; 7.09e-001 \\ 6.34e-001; 5.45e-001 \\ 4.663053e-001 \end{bmatrix}$
SOF-SPL_log	5.88e-002	5.55e+000	$\begin{bmatrix} 6.86e-001; -1.69e-001 \\ 6.88e-001; -5.54e-001 \\ 4.987661e-001 \end{bmatrix}$
GCV-SPL_log	5.34e-002	4.94e+000	$\begin{bmatrix} 1.01e+000; -7.96e-003 \\ -4.85e-001; -8.95e-001 \\ 5.496191e-001 \end{bmatrix}$
DP-SPL_log	7.99e-002	6.64e+000	$\begin{bmatrix} 1.00e+000; 1.81e-001 \\ 1.21e-001; -1.14e+000 \\ 4.093088e-001 \end{bmatrix}$

using all available singular values. That means that the Picard parameter has been computed correctly and we didn't lose much information. The new Hybrid method using GCV to find the solution is the best whereas the TSVD with the Discrepancy Principle behaves the worst. The Truncated Tikhonov is the best method if the GCV method is not used and very close to the Hybrid method when GCV is used.

Figure 6.19: Example 4: Computed solutions.

Figure 6.20: Example 4: Computed solutions, continued.

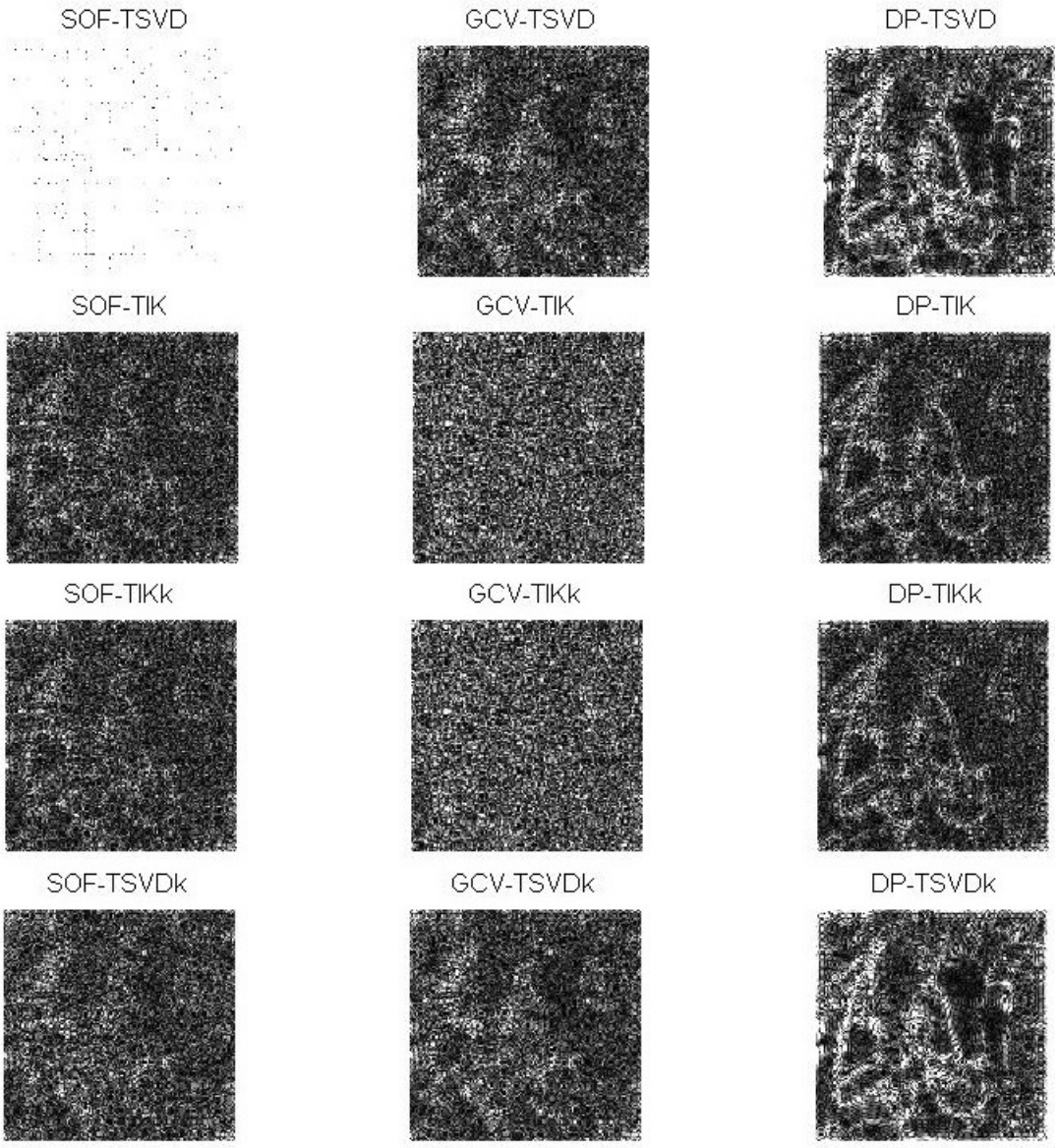

Figure 6.21: Example 4: Errors.

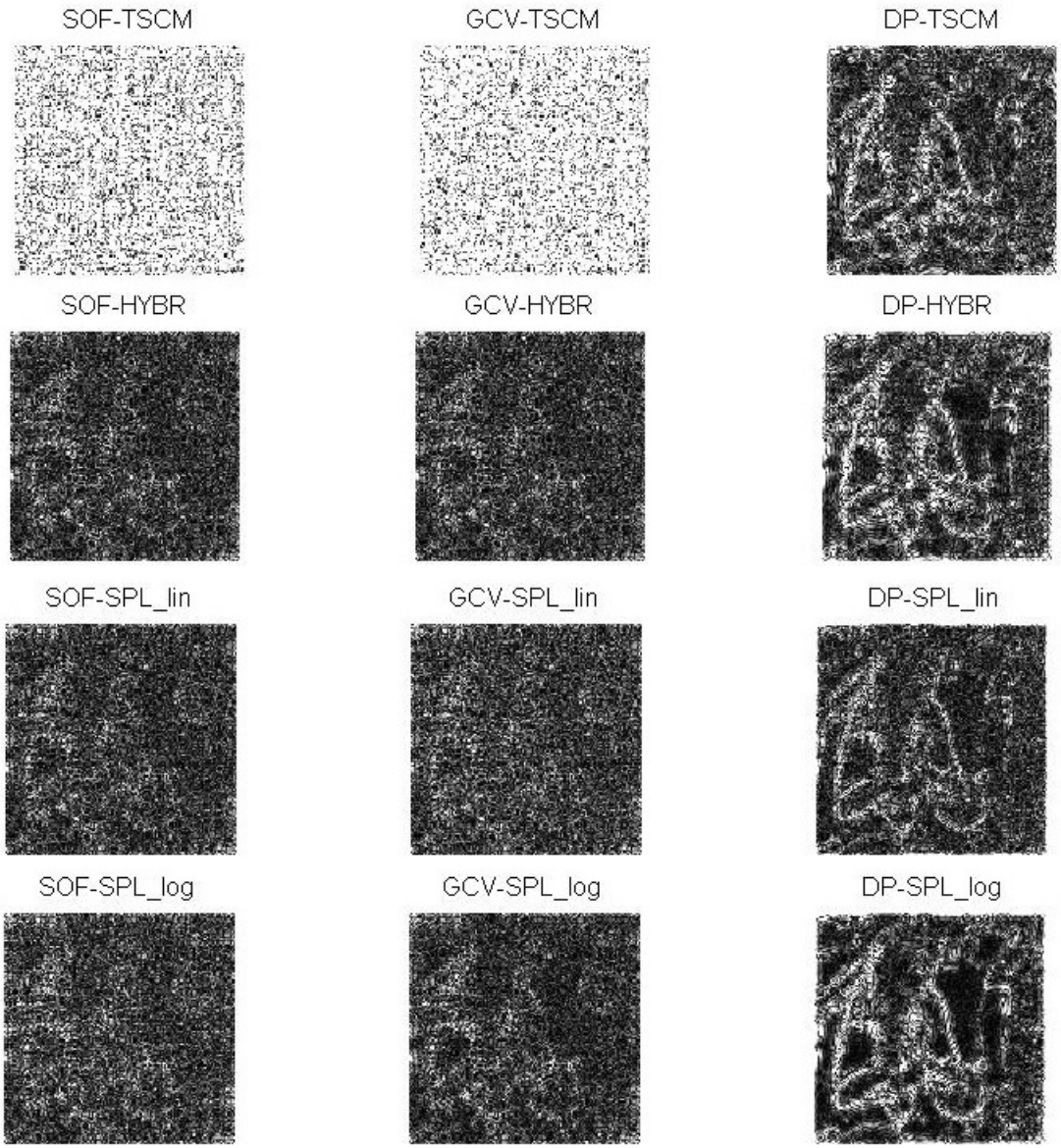

Figure 6.22: Example 4: Errors, continued.

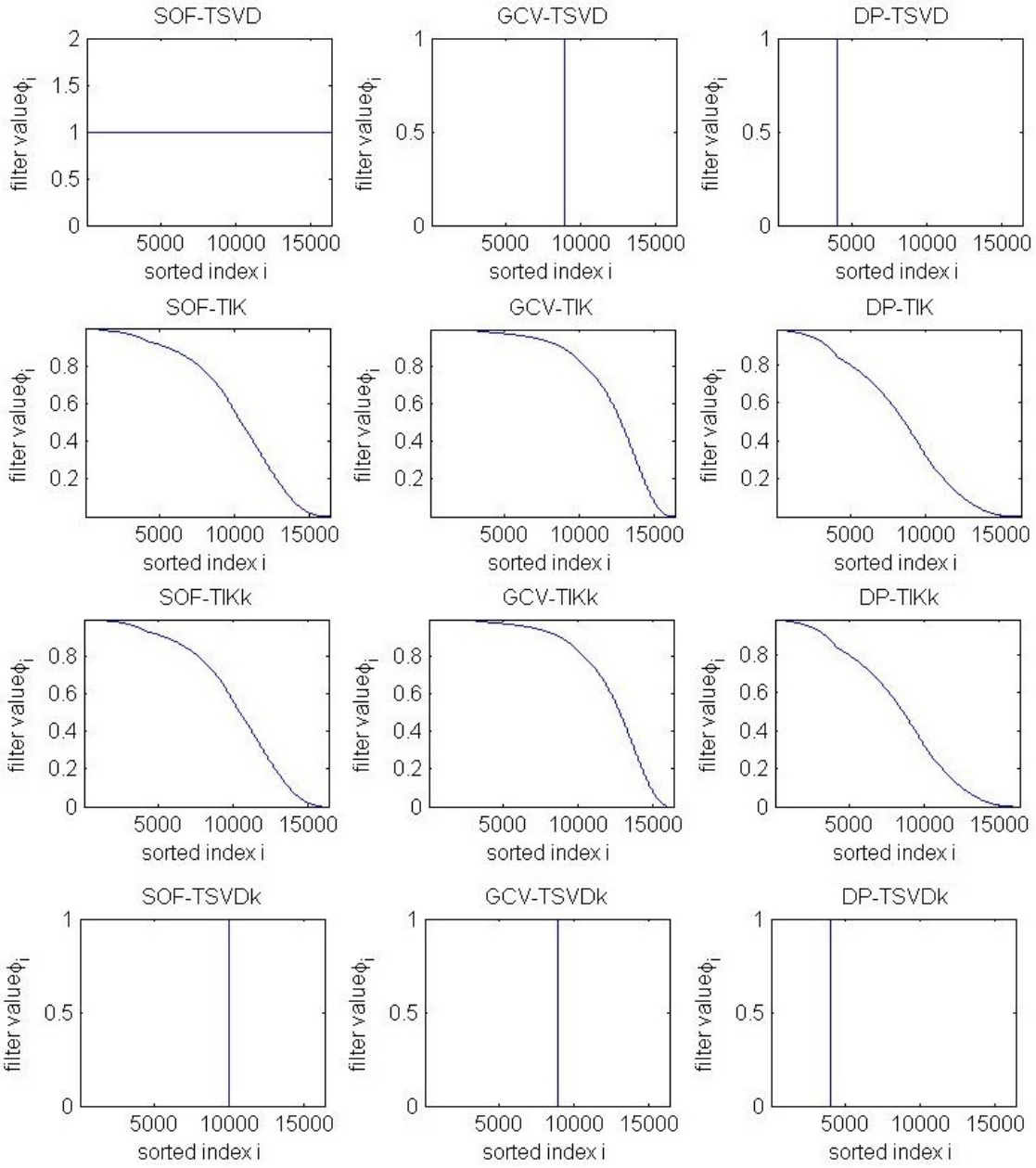

Figure 6.23: Example 4: Filters.

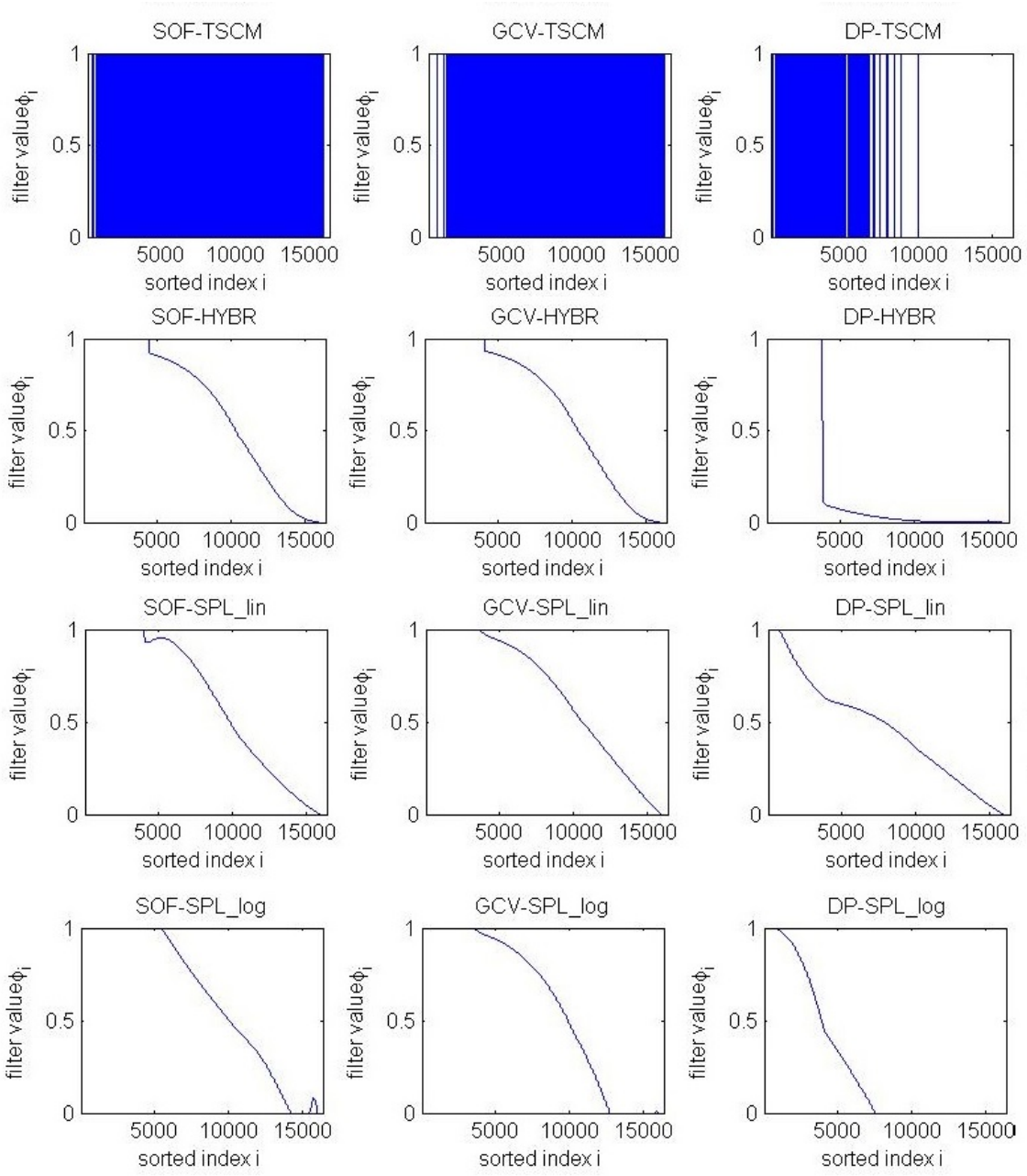

Figure 6.24: Example 4: Filters, continued.

6.2.5 Example 5

This experiment is performed using the 128×128 Barbara image. The noise that is added is of standard deviation $s = 1$. The signal to noise ratio that comes

from that is SNR=112.820. The Picard Parameter that is computed and used for all the methods that require it is $k = 9943$ that results into 6442 cropped eigenvalues and an estimated standard deviation of exp_stdev = 1.050649.

Tables 6.9 and 6.10 present the results.

Figures 6.25 and 6.26 show the restored images using the above methods. Figures 6.27 and 6.28 show the error images, and Figures 6.29 and 6.30 show the optimal filters.

Even though the SOF-TSVD does not fail in this case, the SOF-HYBR and the GCV-HYBR are better than the TSVD and the TIK filters when computed using the same methods. DP-TIK is still better than the the Hybrid though when computed using the Discrepancy Principle. The spline filters give very good results as well similar to the other methods.

It can be seen here as well that the DP method oversmooths the image and that is why the error images show the edges of Barbara.

Table 6.9: Example 5 results.

Method	Relative error	Average error	Lambda
SOF-TSVD	5.12e-002	4.87e+000	8.92e+003
GCV-TSVD	5.26e-002	4.93e+000	7.97e+003
DP-TSVD	7.72e-002	6.72e+000	3.47e+003
SOF-TIK	4.82e-002	4.50e+000	1.28e-002
GCV-TIK	6.70e-002	6.53e+000	1.38e-003
DP-TIK	4.98e-002	4.61e+000	1.59e-002
SOF-TIKk	4.85e-002	4.55e+000	7.07e-003
GCV-TIKk	5.04e-002	4.83e+000	9.65e-004
DP-TIKk	5.16e-002	4.76e+000	1.58e-002
SOF-TSVDk	5.12e-002	4.87e+000	8.92e+003
GCV-TSVDk	5.26e-002	4.93e+000	7.97e+003
DP-TSVDk	7.72e-002	6.72e+000	3.47e+003
SOF-TSCM	5.12e-002	4.88e+000	1.49e+000
GCV-TSCM	5.13e-002	4.91e+000	1.34e+000
DP-TSCM	6.95e-002	6.22e+000	8.97e+000
SOF-HYBR	4.80e-002	4.45e+000	[4.46e+003, 1.08e-002]
GCV-HYBR	4.79e-002	4.46e+000	[3.99e+003, 8.99e-003]
DP-HYBR	7.48e-002	6.41e+000	[3.14e+003, 4.61e-001]

Table 6.10: Example 5 results, continued.

Method	Relative error	Average error	Lambda
SOF-SPL_lin	4.86e-002	4.50e+000	$\begin{bmatrix} 3.87e+002; -2.17e+002 \\ -3.28e+002; -2.17e+001 \\ 3.287367e+000 \end{bmatrix}$
GCV-SPL_lin	4.88e-002	4.50e+000	$\begin{bmatrix} 1.61e+002; -1.80e+001 \\ -2.74e+002; -1.06e+001 \\ 2.153707e+000 \end{bmatrix}$
DP-SPL_lin	6.14e-002	5.05e+000	$\begin{bmatrix} 1.44e-001; 6.99e-001 \\ 5.98e-001; 5.57e-001 \\ 4.899174e-001 \end{bmatrix}$
SOF-SPL_log	4.84e-002	4.48e+000	$\begin{bmatrix} 4.85e-001; -1.07e+000 \\ 6.71e-001; 2.50e-001 \\ 5.310893e-001] \end{bmatrix}$
GCV-SPL_log	4.84e-002	4.48e+000	$\begin{bmatrix} 6.39e-001; -1.44e+000 \\ 5.67e+000; 2.21e-001 \\ 5.313338e-001 \end{bmatrix}$
DP-SPL_log	7.46e-002	6.14e+000	$\begin{bmatrix} 1.29e+000; 2.12e-001 \\ 1.25e-001; 1.12e-001 \\ -3.996123e-001 \end{bmatrix}$

Figure 6.25: Example 5: Computed solutions.

Figure 6.26: Example 5: Computed solutions, continued.

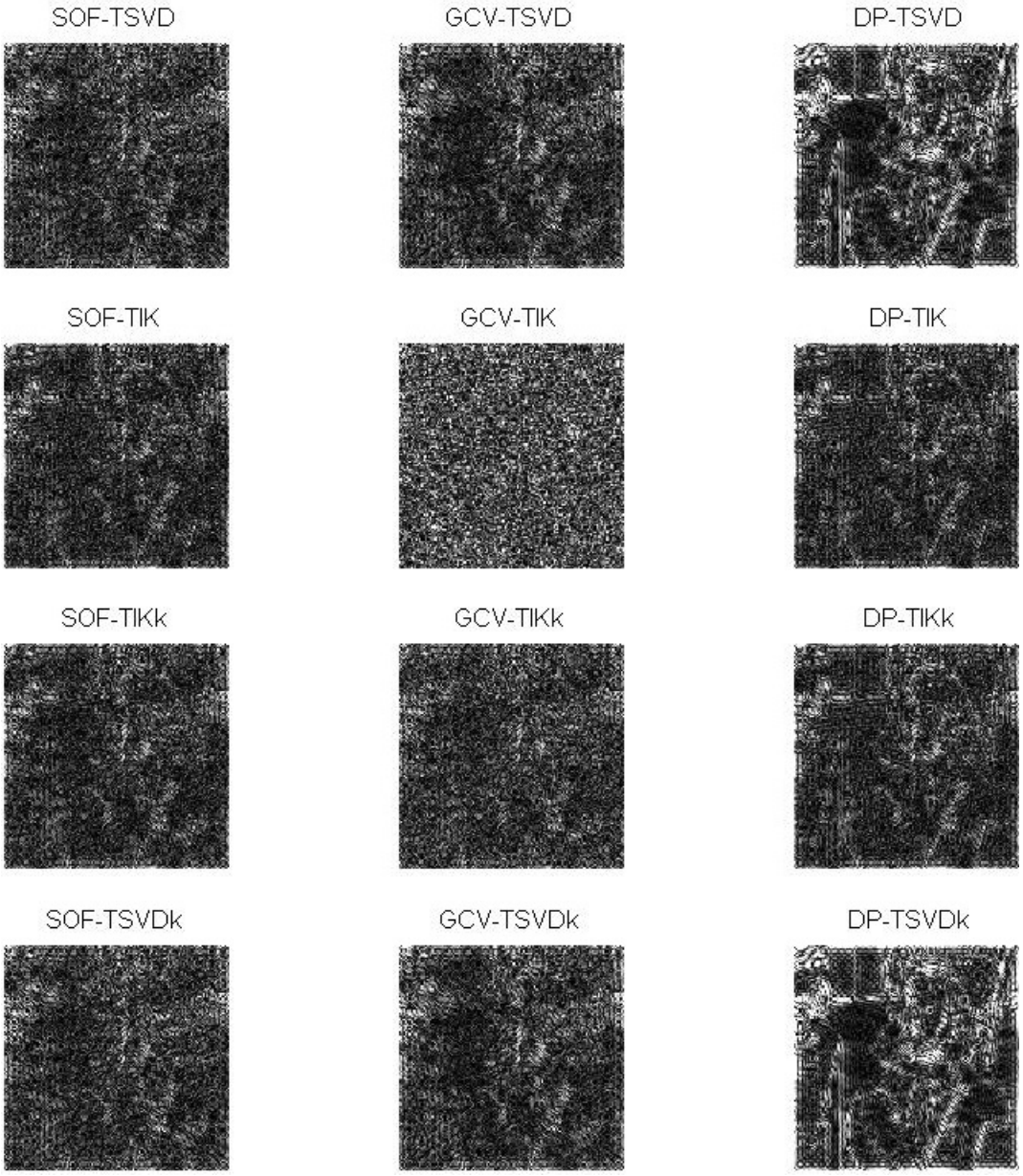

Figure 6.27: Example 5: Errors.

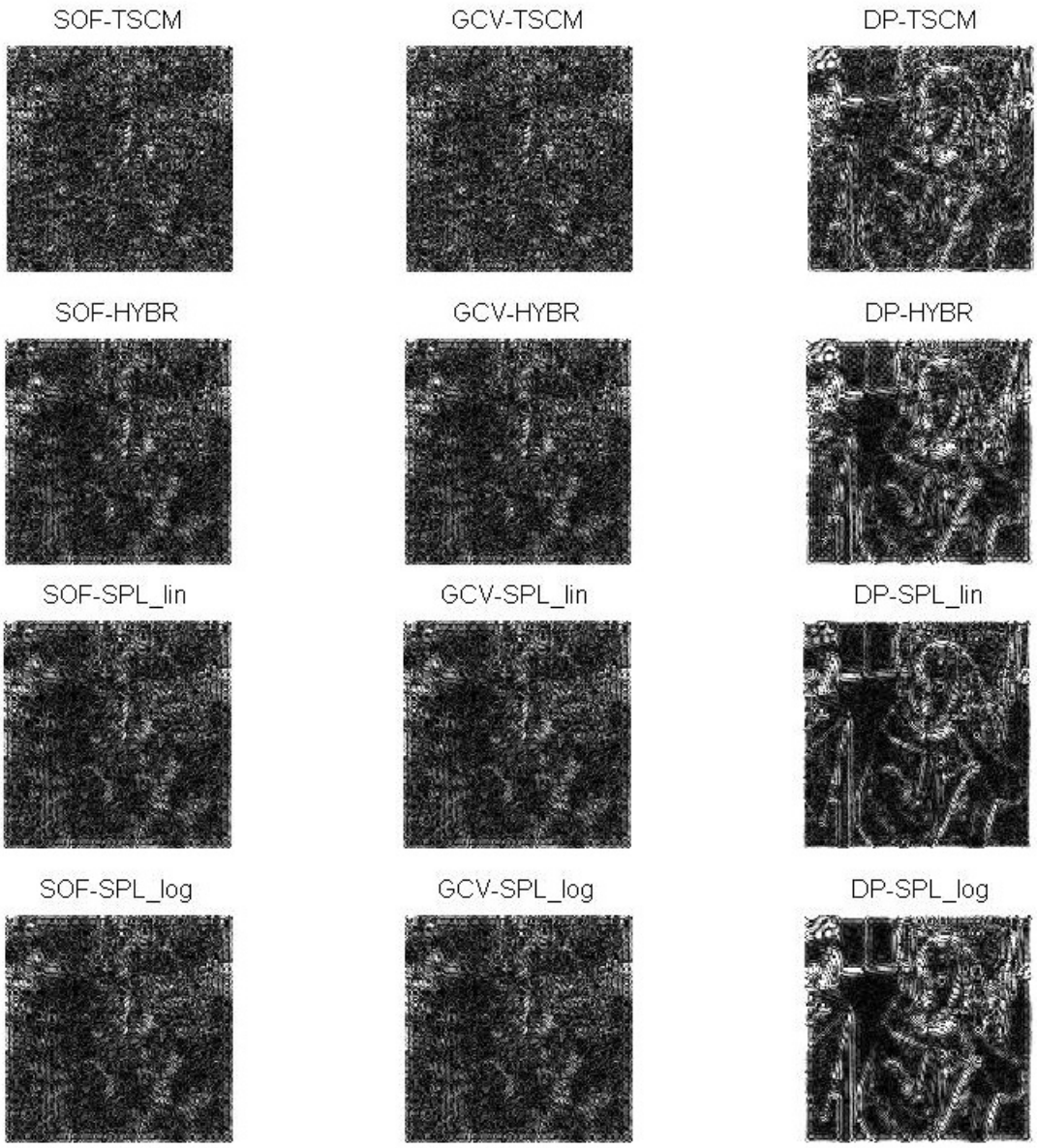

Figure 6.28: Example 5: Errors, continued.

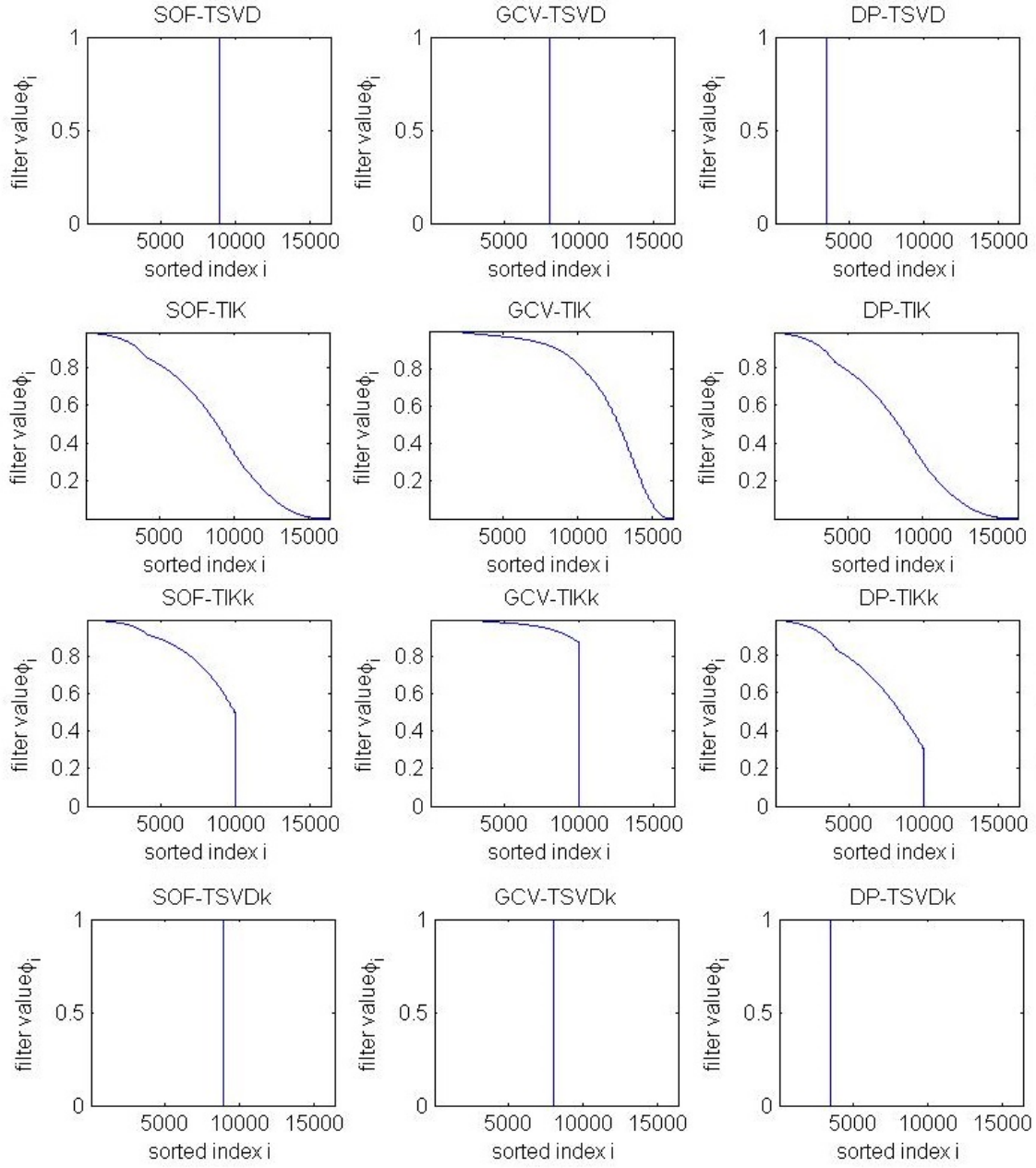

Figure 6.29: Example 5: Filters.

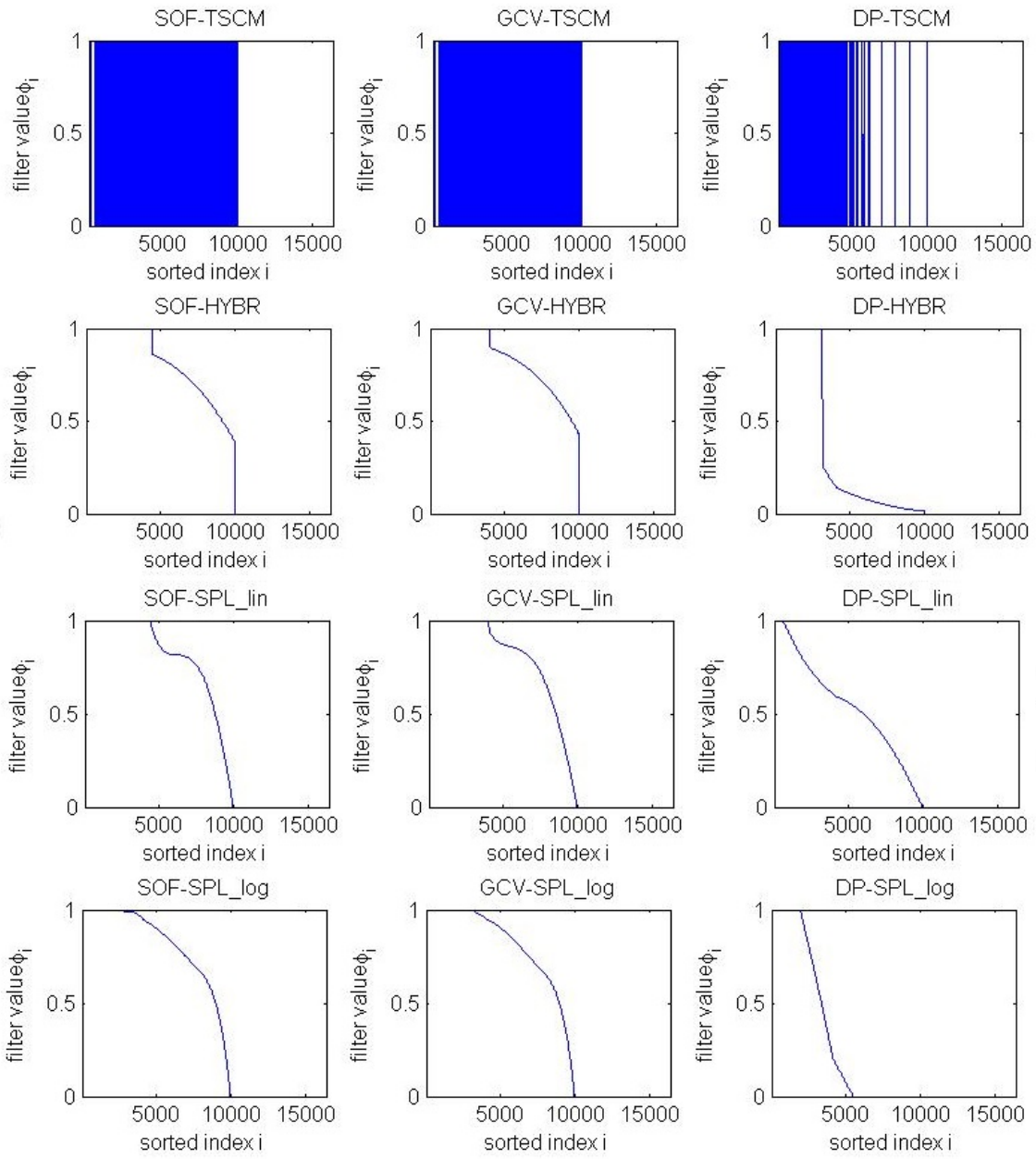

Figure 6.30: Example 5: Filters, continued.

6.2.6 Example 6

This experiment is performed using the 256×256 UMD image. The noise that is added is of standard deviation $s = 1$. The signal to noise ratio that comes from

that is SNR=205.4301. The Picard Parameter that is computed and used for all the methods that require it is $k = 64790$ that results into 747 cropped eigenvalues and an estimated standard deviation of exp_stdev = 0.98769.

Tables 6.11 and 6.12 present the results. We omitted the HYBR problem due to the computational time it requires.

Figures 6.31 and 6.32 show the restored images using the above methods. Figures 6.33 and 6.34 show the error images, and Figures 6.35 and 6.36 show the optimal filters.

In general the errors now are smaller than with the Pirate image when the same noise is used. That tells us that the image plays an important role on how the restoration is affected by the image. Something noticeable from the error figures is the border of the images that is whiter than the rest of the image including the logo itself.

In this case the SPL_ lin method gives the smallest error when computed with the SOF method or the GCV method. The TSCM filter though that gives relative errors of an order of magnitude higher than the rest filters when computed with the SOF or GCV methods, gives the best results when the DP method is used.

Table 6.11: Example 6 results.

Method	Relative error	Average error	Lambda
SOF-TSVD	2.48e-002	3.91e+000	3.54e+004
GCV-TSVD	2.53e-002	3.65e+000	3.13e+004
DP-TSVD	3.77e-002	3.86e+000	1.38e+004
SOF-TIK	2.48e-002	4.06e+000	7.37e-003
GCV-TIK	4.46e-002	7.76e+000	7.87e-004
DP-TIK	2.47e-002	4.01e+000	8.54e-003
SOF-TIKk	2.48e-002	4.06e+000	7.37e-003
GCV-TIKk	4.46e-002	7.76e+000	7.87e-004
DP-TIKk	2.47e-002	4.01e+000	8.54e-003
SOF-TSVDk	2.48e-002	3.91e+000	3.54e+004
GCV-TSVDk	2.53e-002	3.65e+000	3.13e+004
DP-TSVDk	3.77e-002	3.86e+000	1.38e+004
SOF-TSCM	3.29e-001	5.73e+001	1.40e+000
GCV-TSCM	4.20e-001	7.30e+001	4.58e-001
DP-TSCM	2.38e-002	3.62e+000	1.11e+001

Table 6.12: Example 6 results, continued.

Method	Relative error	Average error	Lambda
SOF-SPL_lin	2.36e-002	3.62e+000	$\begin{bmatrix} 1.37e+000; 3.86e-001 \\ -3.97e-003; 3.67e+000 \\ 4.773255e-002 \end{bmatrix}$
GCV-SPL_lin	2.42e-002	3.84e+000	$\begin{bmatrix} -2.45e+001; 2.56e+001 \\ 1.04e+001; 4.59e+000 \\ 3.226832e-002 \end{bmatrix}$
DP-SPL_lin	3.05e-002	4.54e+000	$\begin{bmatrix} 6.24e-001; 1.02e-001 \\ 4.23e-001; 2.45e-001 \\ 6.403497e-001 \end{bmatrix}$
SOF-SPL_log	3.44e-002	5.78e+000	$\begin{bmatrix} 4.14e-002; 3.74e-001 \\ -2.57e-002; -1.64e-001 \\ 2.620007e-001 \end{bmatrix}$
GCV-SPL_log	2.25e-002	3.19e+000	$\begin{bmatrix} 9.26e-001; 1.36e-001 \\ -2.76e-001; -1.09e+000 \\ 6.735136e-001 \end{bmatrix}$
DP-SPL_log	3.78e-002	3.56e+000	$\begin{bmatrix} 8.17e-001; 2.33e+000 \\ 7.91e+000; -4.09e+000 \\ -1.603588e-001 \end{bmatrix}$

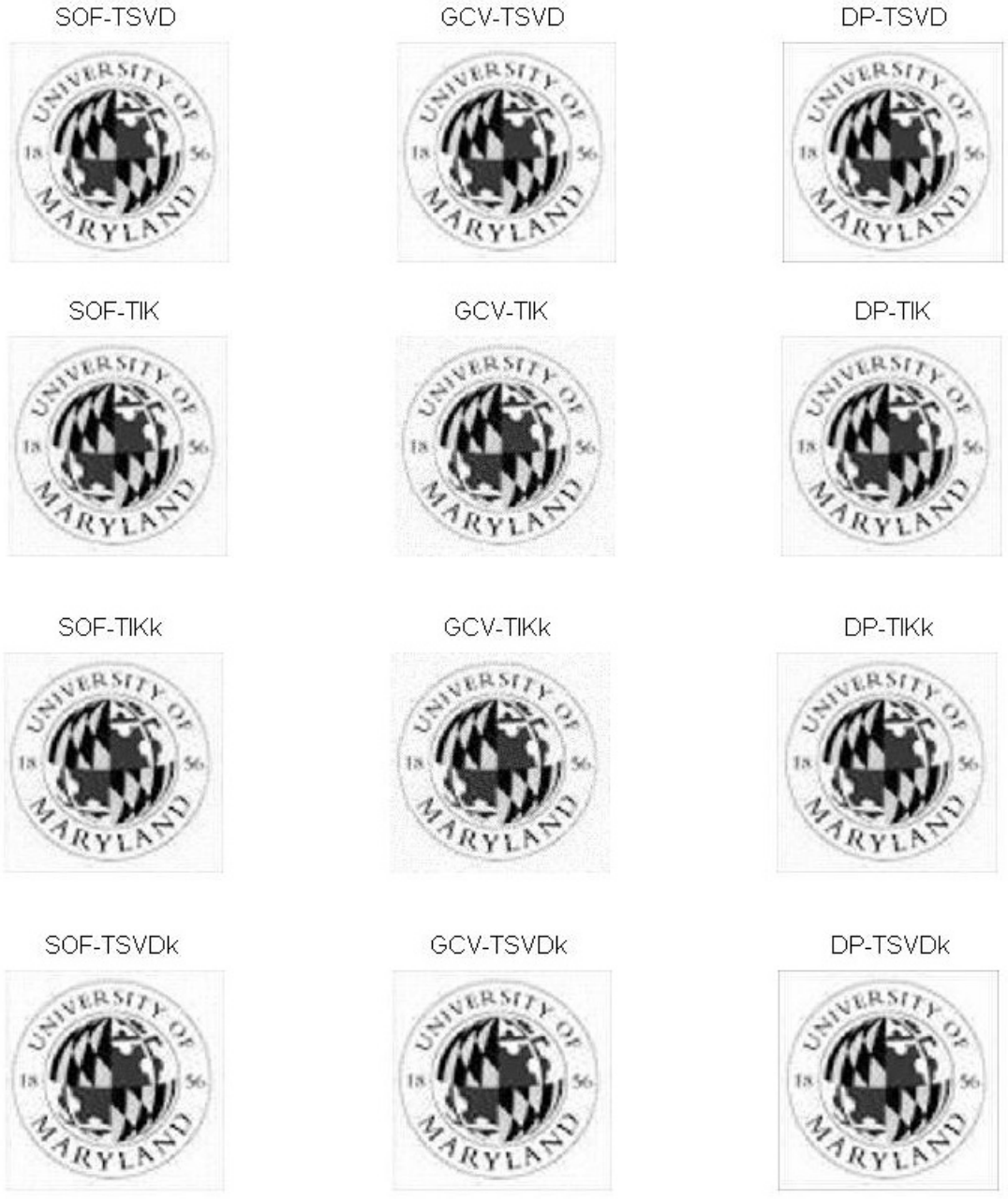

Figure 6.31: Example 6: Computed solutions.

Figure 6.32: Example 6: Computed solutions, continued.

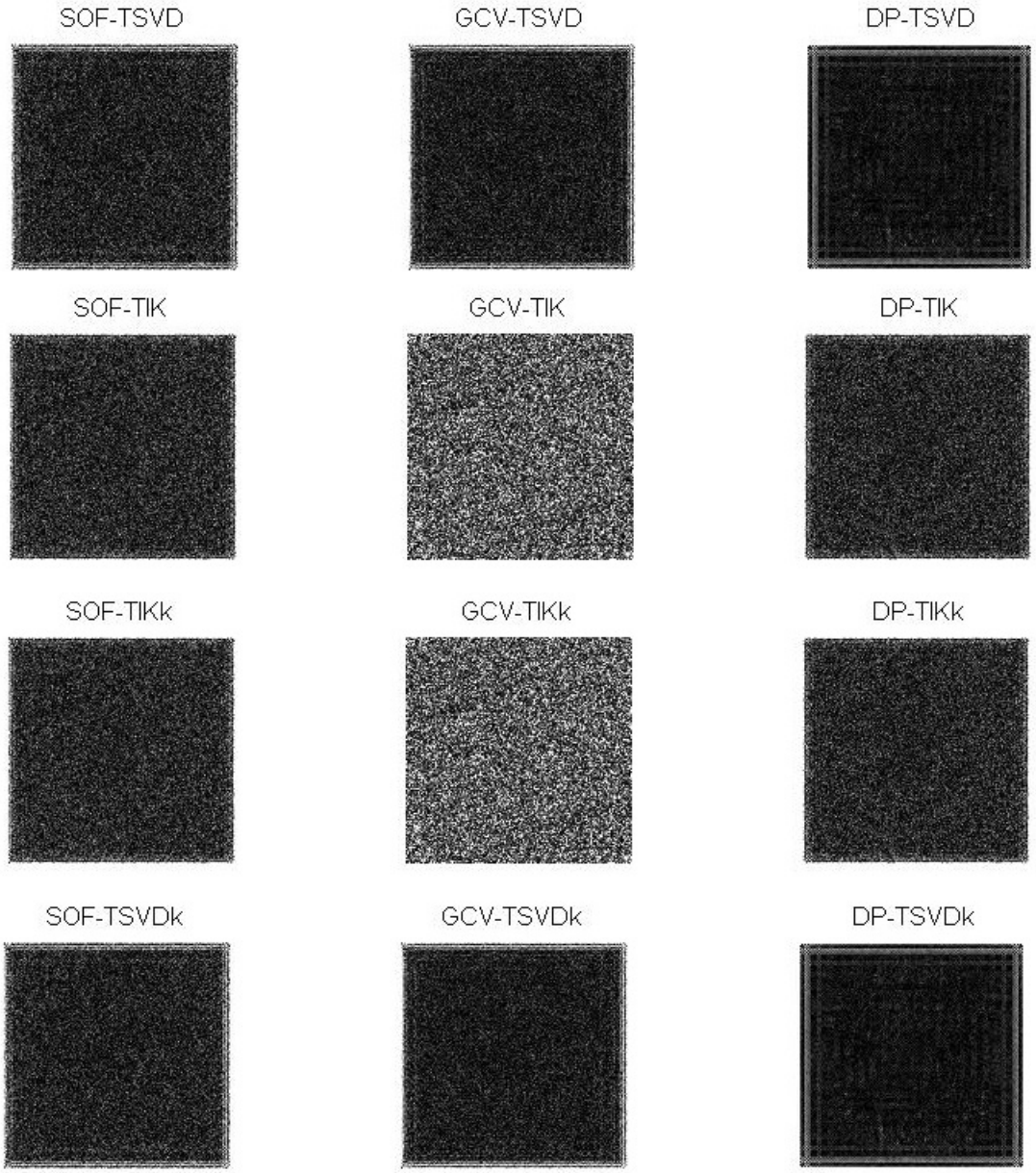

Figure 6.33: Example 6: Errors.

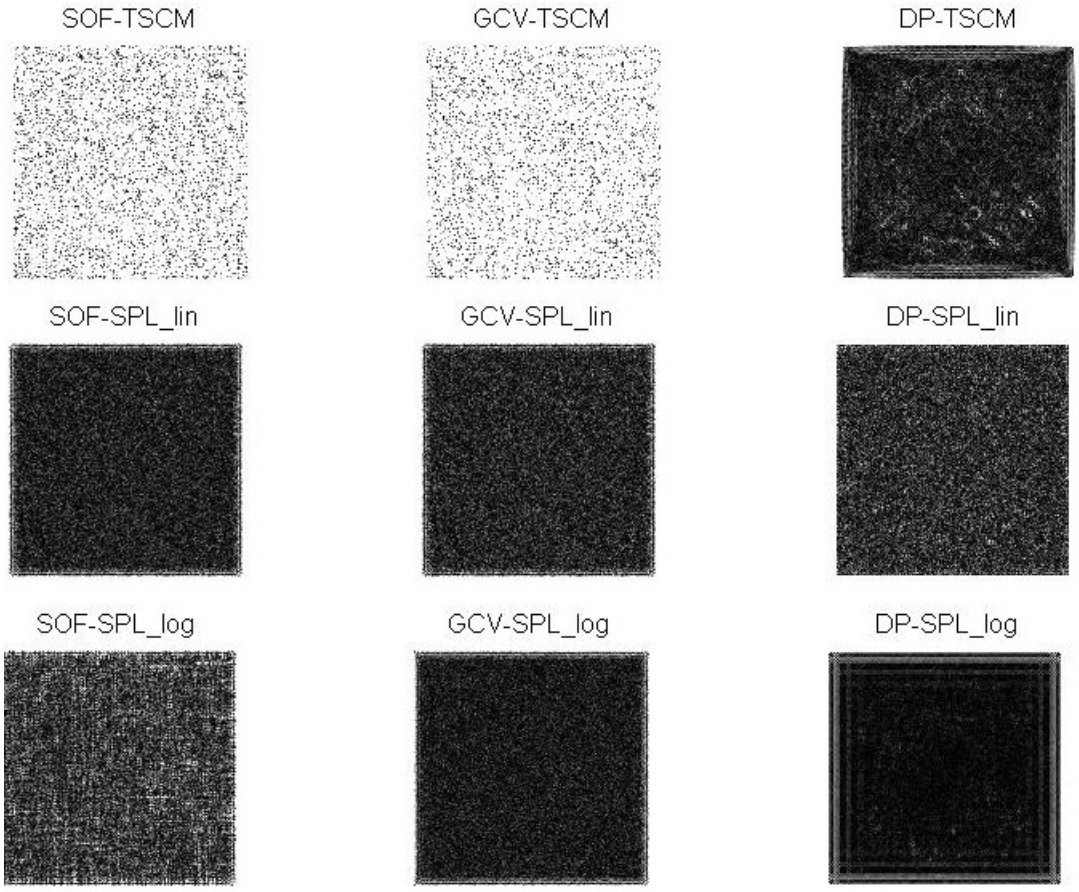

Figure 6.34: Example 6: Errors, continued.

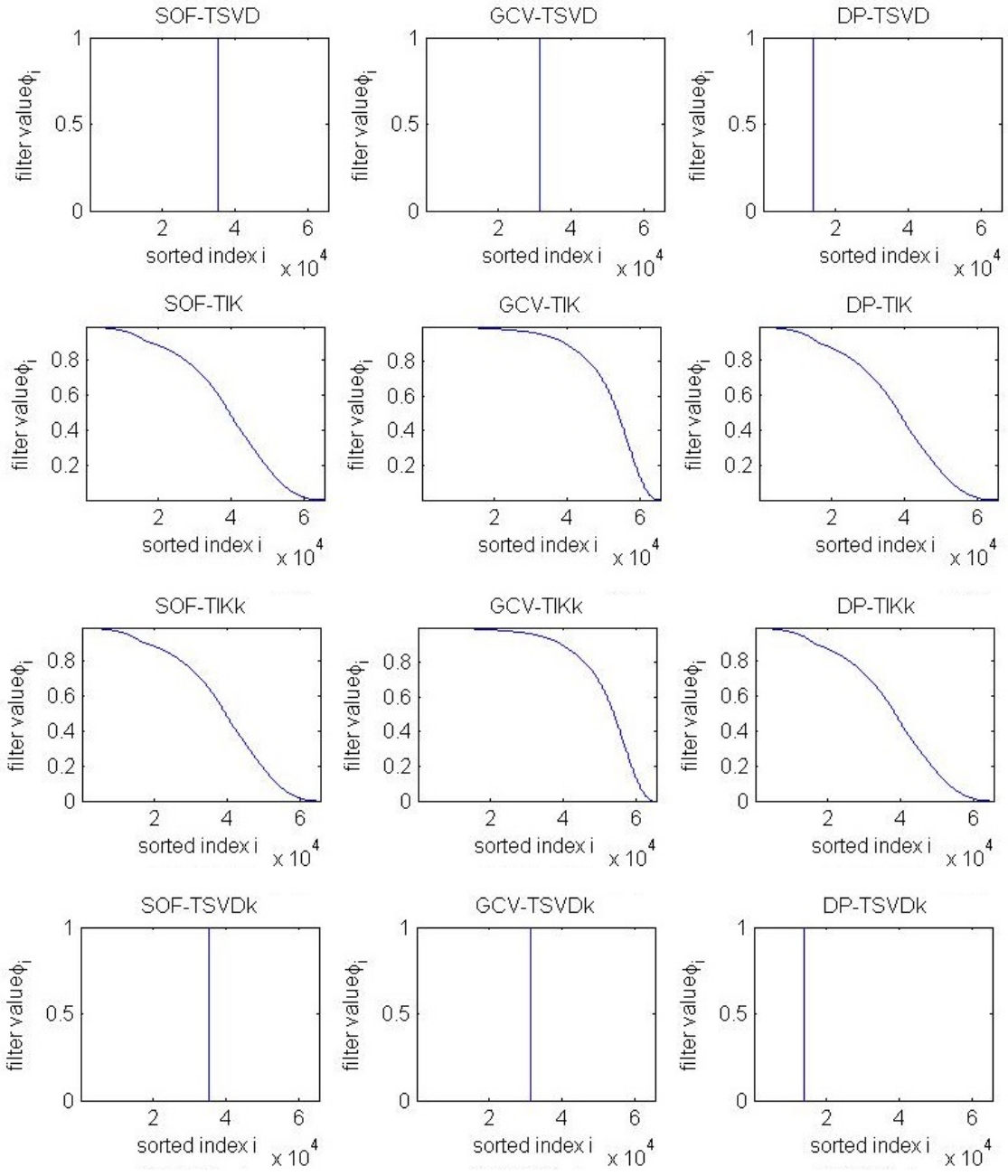

Figure 6.35: Example 6: Filters.

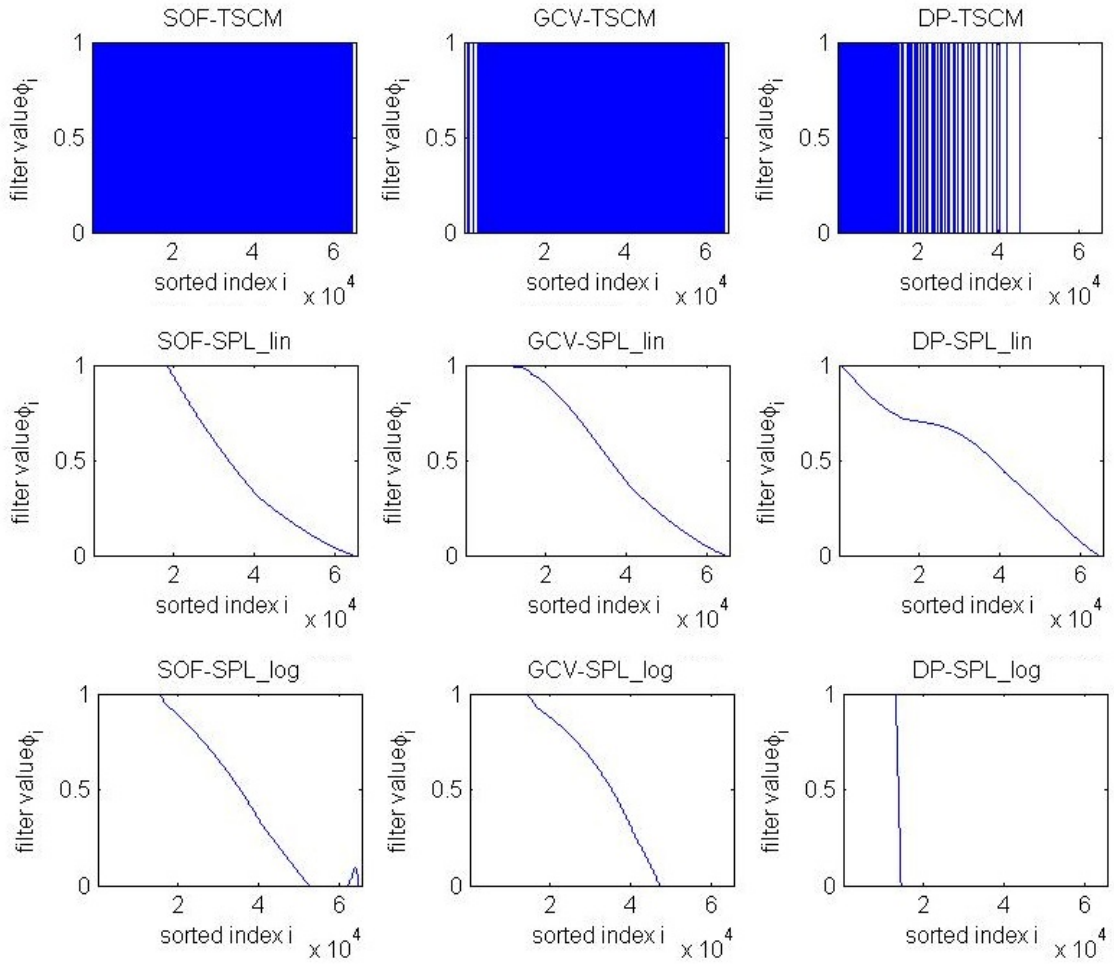

Figure 6.36: Example 6: Filters, continued.

6.2.7 Example 7

This experiment is performed using the 256×256 UMD image. The noise that is added is of standard deviation $s = 10$. The signal to noise ratio that comes from that is SNR=20.539. The Picard Parameter that is computed and used for all the methods that require it is $k = 64790$ that results into 747 cropped eigenvalues and an estimated standard deviation of exp_stdev = 9.8769.

Table 6.13: Example 7 results.

Method	Relative error	Average error	Lambda
SOF-TSVD	5.25e-002	7.92e+000	1.29e+004
GCV-TSVD	5.25e-002	7.92e+000	1.29e+004
DP-TSVD	1.16e-001	1.53e+001	2.70e+003
SOF-TIK	8.94e-002	1.55e+001	4.88e-002
GCV-TIK	1.57e-001	2.74e+001	9.82e-003
DP-TIK	1.03e-001	1.88e+001	8.99e-002
SOF-TIKk	8.94e-002	1.55e+001	4.87e-002
GCV-TIKk	1.57e-001	2.74e+001	9.82e-003
DP-TIKk	1.03e-001	1.88e+001	8.99e-002
SOF-TSVDk	5.25e-002	7.92e+000	1.29e+004
GCV-TSVDk	5.25e-002	7.92e+000	1.29e+004
DP-TSVDk	1.16e-001	1.53e+001	2.70e+003
SOF-TSCM	3.29e+000	5.73e+002	1.40e+001
GCV-TSCM	4.20e+000	7.31e+002	4.35e+000
DP-TSCM	1.04e-001	1.65e+001	1.62e+002

Table 6.14: Example 7 results, continued.

Method	Relative error	Average error	Lambda
SOF-SPL_lin	2.53e-001	4.40e+001	$\begin{bmatrix} 5.51e+004; 6.53e+004 \\ -4.44e+002; -9.18e+003 \\ 7.746834e+002 \end{bmatrix}$
GCV-SPL_lin	4.82e-002	6.71e+000	$\begin{bmatrix} -3.16e+000 \\ -9.45e+000 \\ -2.29e+000 \\ 3.73e+000 \\ -8.148776e-001 \end{bmatrix}$
DP-SPL_lin	1.18e-001	1.67e+001	$\begin{bmatrix} 5.87e-001; 1.90e-001 \\ 5.15e-001; 2.18e-001 \\ 1.408959e-001 \end{bmatrix}$
SOF-SPL_log	2.46e-001	4.28e+001	$\begin{bmatrix} 3.27e-001; -1.32e-003 \\ 7.42e-001; -5.77e-001 \\ 5.810294e-001 \end{bmatrix}$
GCV-SPL_log	4.82e-002	6.74e+000	$\begin{bmatrix} 4.84e-001; 9.32e-001 \\ 8.66e-003; -1.20e+000 \\ 2.407503e-001 \end{bmatrix}$
DP-SPL_log	1.15e-001	1.37e+001	$\begin{bmatrix} 3.18e+000; 7.85e+000 \\ 3.57e+001; -2.05e+001 \\ -1.578348e+000 \end{bmatrix}$

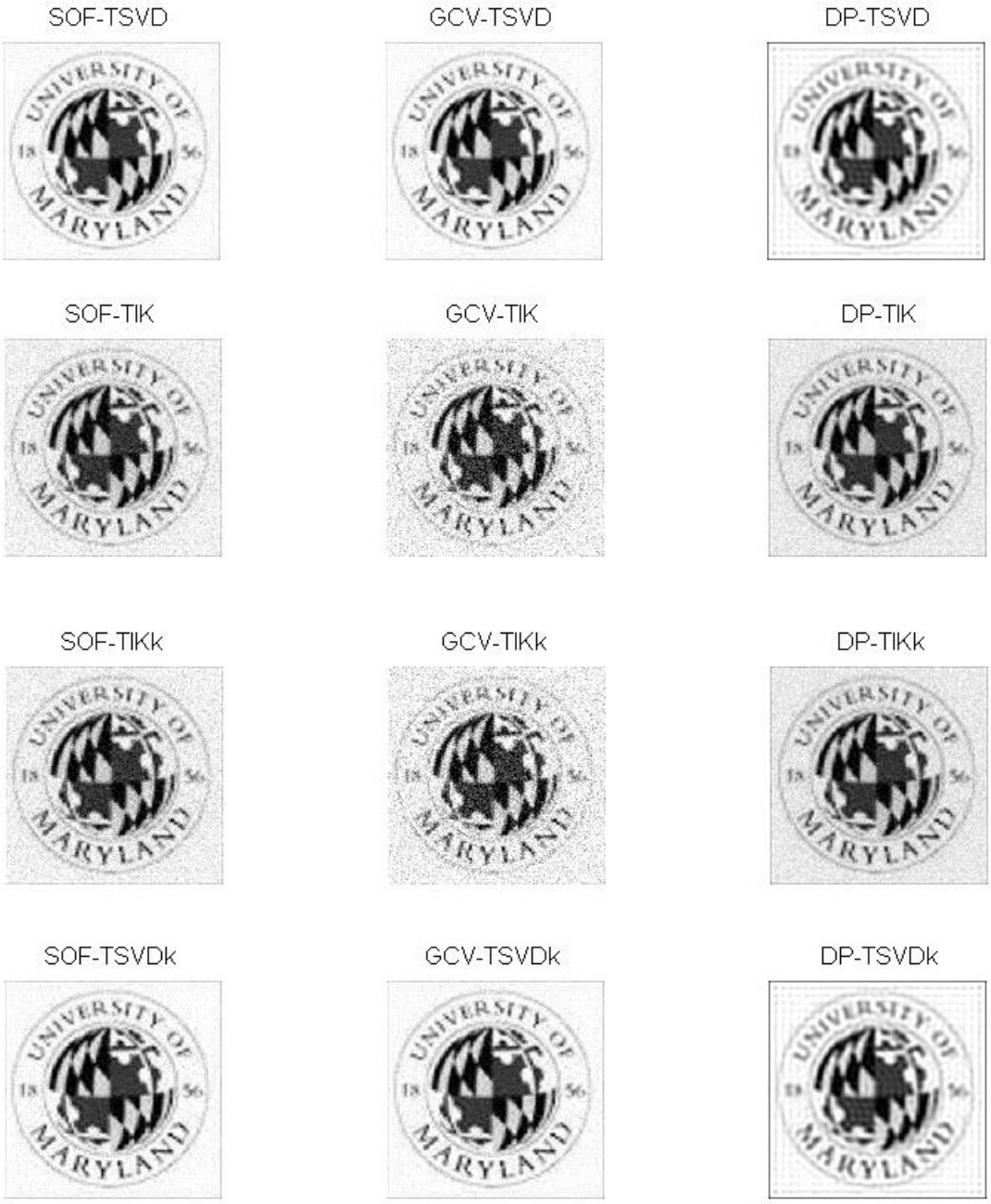

Figure 6.37: Example 7: Computed solutions.

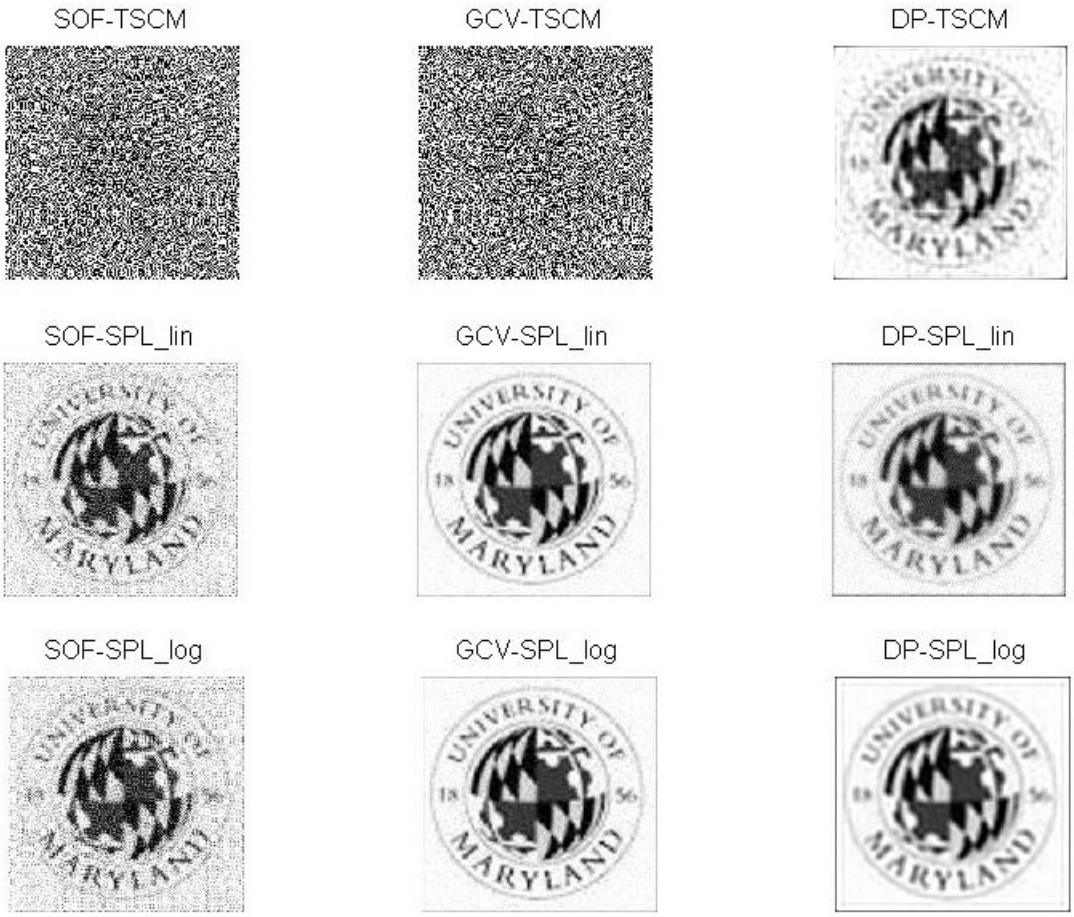

Figure 6.38: Example 7: Computed solutions, continued.

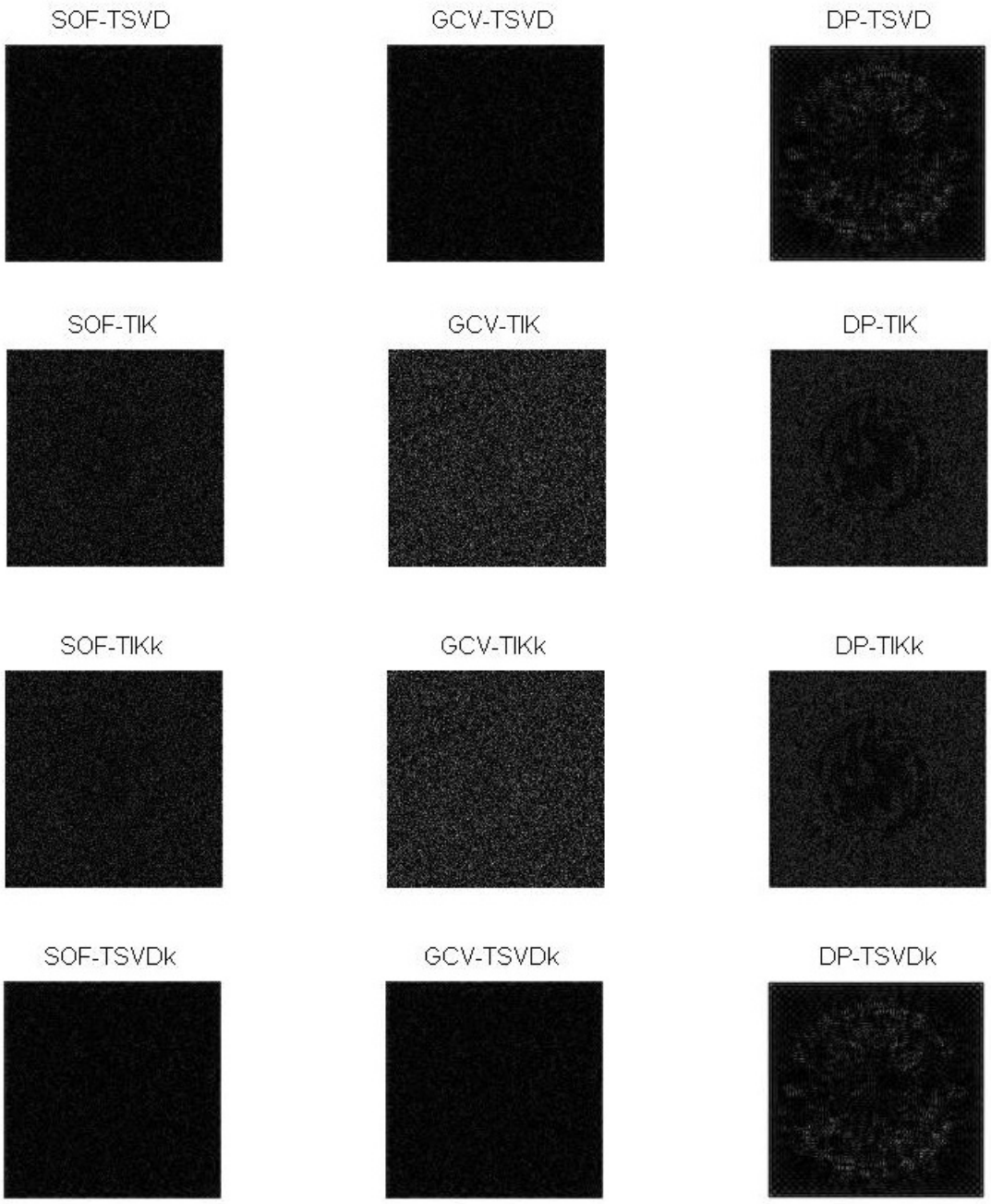

Figure 6.39: Example 7: Errors.

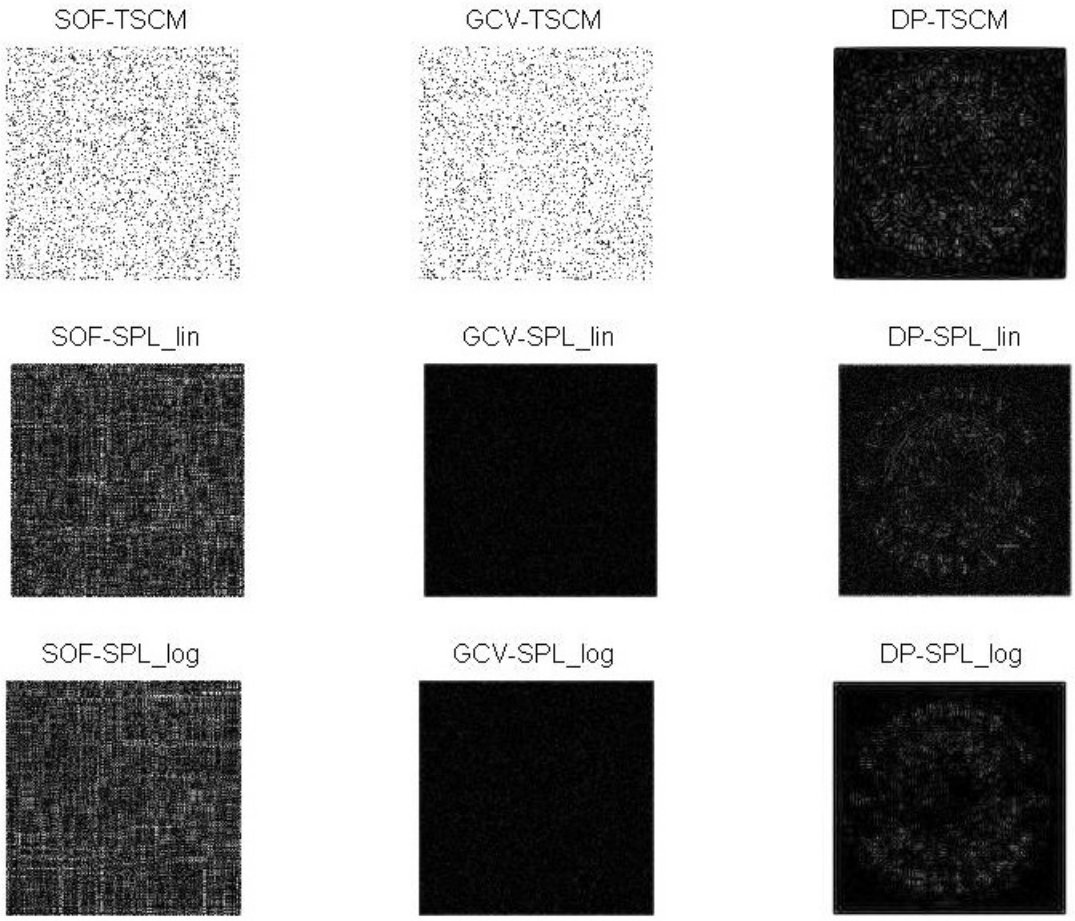

Figure 6.40: Example 7: Errors, continued.

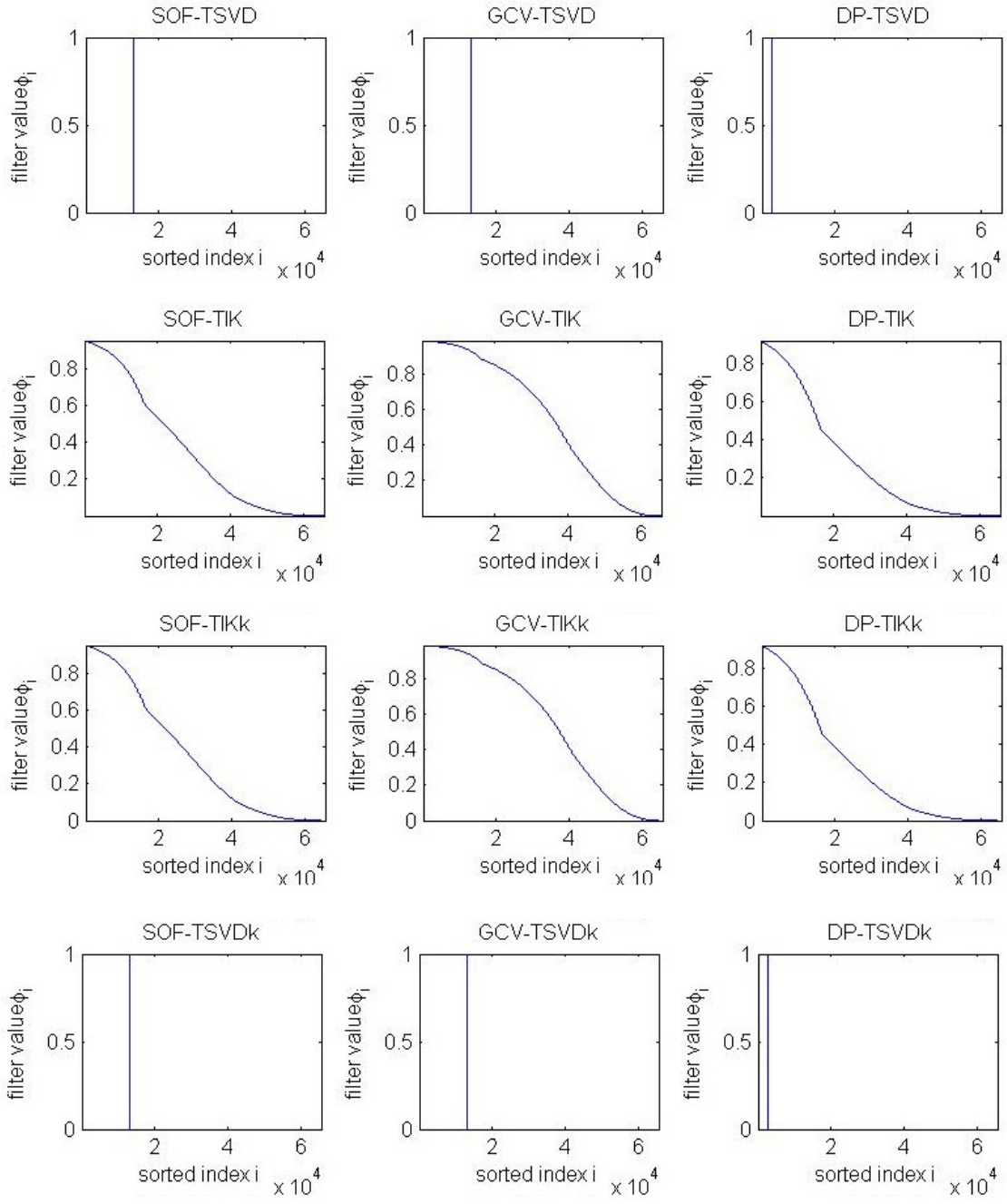

Figure 6.41: Example 7: Filters.

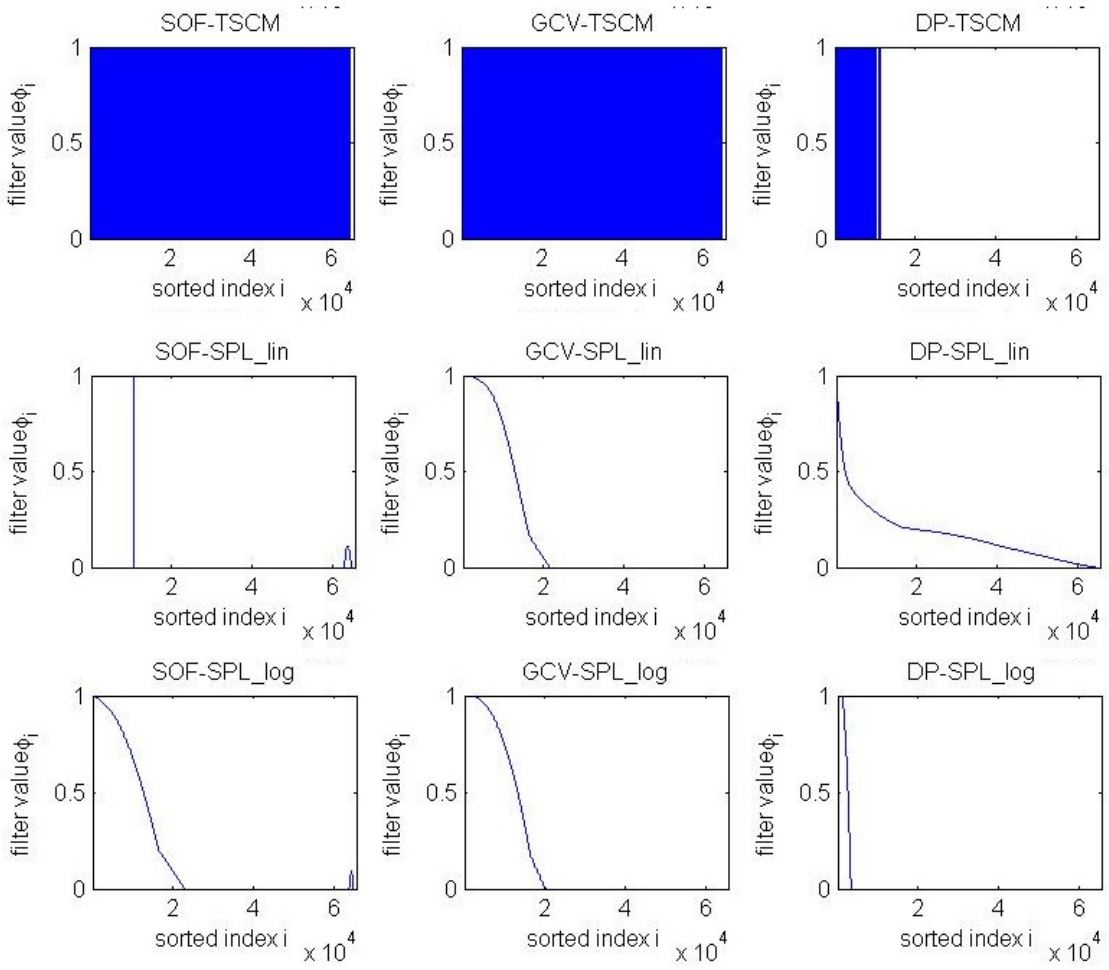

Figure 6.42: Example 7: Filters, continued.

Tables 6.13 and 6.14 present the results of seven different filters (omitting HYBR) for the computation of the solution of the image restoration problem.

Figures 6.37 and 6.38 show the restored images using the above methods. Figures 6.39 and 6.40 show the error images, and Figures 6.41 and 6.42 show the optimal filters.

As expected, the errors here increase due to the higher noise that was added to the image. In general, the SOF method produces better results than the GCV and

the DP methods except for the case of the TSCM filter and the spline filters. Once again we can conclude that different filters work better with different methods. The TSVD and TSVDk seem to work better than the rest with the SOF method whereas the TIK and TIKk seem to give better results than the other methods paired with the DP. The SPL_ lin and SPL_ log paired with the GCV give the best solutions and the least error overall.

This can be seen in the figures as well. The TSCM with SOF and GCV does not give a solution that resembles the UMD logo and different methods paired with different filters either oversmooth the image or keep part of the noise in them. In contrast with the previous example, here we see more noise in the logo than in the borders of the image.

From the filters figures, we can also conclude that we need only a small portion of the singular values to get a good estimate of the image solution. It is also interesting to see how a small spike between the SOF-SPL_ log and GCV-SPL_ lin filters creates such a different solution with different errors.

6.3 Conclusions

It seems from the experiments that the in general the SOF method works better than the Discrepancy Principle which gives error images that show the edges of the images. It seems though that the TSCMk filter does not work as well with the SOF method than it does with the Discrepancy Principle. Our TSVDk filter also performs better than the TSVD filter which sometimes fails. Different filters give

smaller errors with different methods. It is not always the same method that will give the best results and a filter might give very good results with one method but very bad results with another one. In general, our new filters and our new method are very competitive with the already known ones since they perform better in many cases. The HYBR filter combines good properties of the TSVD and the TIK filters and in many cases it gives better results than both of the filters it consists of. The trade-off for this filter though is the running time which is not reported here but is larger than that of the other filters since it combines both discrete and continuous parameters and thus it needs more operations for the minimization.

Chapter 7: Conclusions and Future Work

7.1 Conclusions

The goal of this work was to solve the problem

$$\mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{b}, \tag{7.1}$$

where $\mathbf{b}$ is the input, $\mathbf{A}$ is a known matrix, $\mathbf{e}$ is a random vector of additive Gaussian noise and $\mathbf{x}$ the unknown vector.

Assuming that the problem satisfies the discrete Picard condition, we introduced in Chapter 2 the Picard parameter which signifies the point where, in the coordinate system defined by the singular value decomposition of $\mathbf{A}$, the right hand side of (7.1) is dominated by noise. In addition, we reviewed how to estimate the Picard parameter manually and developed an algorithm that automatically computes the Picard parameter, when the noise is Gaussian, using the Lilliefors test. The results from this algorithm are similar to those we compute manually. In later chapters, we used the Picard parameter to reduce the effect of the noise while using spectral filters to minimize the error. In addition, the Picard parameter helps to reduce the computational cost of the restoration of images using spectral filters.

In Chapter 3, we use the Truncated SVD filter with the addition of the Picard

parameter (TSVDk) to estimate the solution. Our main goal was to minimize the error that the solution using the TSVDk filter would have. Since the problem is an inverse ill-posed problem, we do not know what the noise is and so, we had to use a statistical approach to eliminate the noise from our equations. Using the fact that the noise is a random and Gaussian, we were able to rewrite the norm of the error using the expected value of the noise. Doing so, we still needed to know the standard deviation of the unknown noise. The Picard parameter helped us estimate this standard deviation since we know which elements of the known image resemble noise. Using these tools, we developed a method for the TSVDk that computes the near optimal parameter for the specific filter. We compared our results with those computed using the GCV and the Discrepancy Principle. Our method worked very well and the relative and average errors were close to those that the TSVD would give if we knew the real solution. We also took advantage of the Kronecker products for the SVD of the blurring matrix in case the blur is separable. That reduced the computational time of the computation of the SVD, which is the most expensive part of the algorithm.

In Chapter 4, the same statistical analysis was used to determine the near optimal parameters for general spectral filters. Again, the importance of the Picard parameter estimation is clear here since in the process of creating the method, a function that depends on the noise in the image needs to be minimized, and the Picard parameter helps in computing that estimated value. The method we developed and called SOF also gives us a way to estimate the error without knowing anything about the noise except that it has mean zero and known variance. That

gives us a way to quantify uncertainties.

We developed more filters with varied number and continuity properties of the parameters in Chapter 5. These new filters include a Truncated Tikhonov with a continuous parameter, a TSCM that uses the Picard parameter, and a Hybrid method that consists of the Truncated Tikhonov and the TSVDk and involves one discrete and one continuous parameter. We also developed a continuous equivalent to the TSVDk filter and showed that we can create continuous filters from discrete ones. We also proposed Heaviside and tangent filters. Finally, we developed a cubic spline filter. This filter has a general number of knots that can be spaced either linearly or logarithmically.

We tested our SOF algorithm with most of the above filters in Chapter 6 to compare with the GCV and the Discrepancy Principle. The results of our statistical analysis compared to those with the optimal solution are good. The SOF method outperforms DP in most cases. It also gets results close to those of the GCV.

In general, we created a new method and new filters that very well estimate a solution to the ill-posed problem (7.1) and provide some intuition about uncertainty quantification.

7.2 Future work

There are many questions that arose from this project. Some of them were answered but there are others that still need to be considered.

7.2.1 Work on the Picard parameter

- Is there a better way to use the Lilliefors method for the computation of the discrete Picard parameter?
- How can other normality testing techniques be used to develop a method for the computation of the discrete Picard parameter?
- If the additive error is not sampled from a normal distribution, what methods can be used for the estimation of the Picard parameter and what other assumptions are needed?

7.2.2 Work on the SOF method

- The SOF method requires an estimate of the standard deviation of the noise. Are there better ways to do this?
- Could different properties of the blurring matrix reduce the computational cost of the SVD and the SOF method?
- Can other uncertainty quantification estimates be developed?
- Can we combine the SOF and DP methods to obtain better results?

7.2.3 Work on spectral filters

- What combination of filters could be used to get the good properties of both but without increasing the computational cost?

- What happens if we use higher continuity properties in the spline filter?
- Is there a best number of knots for the spline filter?
- Is the linear or the logarithmic scale better for the spline filter?
- How do the filters work without the truncation with the Picard parameter?
- How can the discrete filters be modified to become continuous?
- Are there better filters for a specific image depending on its features, like boundary, sharp, smooth, etc?
- Will future testing reveal recommendations for which particular parameter choice method should be used for each kind of filter?

Chapter A: Point Spread Functions, Construction of Blurring Matrices, and Noise

In Section A.1 we discuss how blurring matrices are constructed from point spread functions (PSFs). Then in Section A.2 we discuss how our data is generated.

A.1 Two common models of blurring

Two common models of blurring are Gaussian blur and separable blur, and we discuss each.

A.1.1 Gaussian blur

In this model, the blurring matrix $\mathbf{A}$ is constructed using spatially invariant blur and Gaussian PSFs. Usually these PSFs are of much smaller size than the original image. Let $\hat{p}$ be the size of the PSF. Then, for $k, l = 1 \dots \hat{p}$, define

$$PSF(k,l) = \exp\Big(-\frac{1}{2}\frac{(k-c_1)^2}{s_1^2} - \frac{1}{2}\frac{(l-c_2)^2}{s_2^2}\Big),$$

where c_1 and c_2 are the coordinates of the center of the PSF, usually $(\lceil \hat{p}/2 \rceil, \lceil \hat{p}/2 \rceil)$. In our experiments we set $\hat{p} = 3$ or $\hat{p} = 5$ and $s_1 = s_2 = 3$. We also set $m = n$, making $\mathbf{A}$ square.

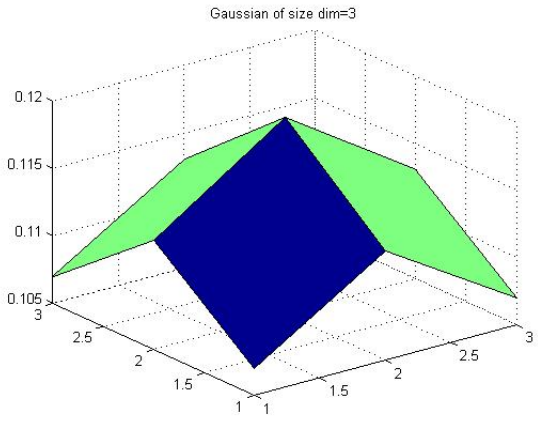

Figure A.1: Point spread functions of size 3×3.

As an example, suppose we have a true image of size 5×5 and a PSF of size 3×3. So the PSF array is of the form

$$\begin{matrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{matrix}$$

where the red denotes the center of PSF and $\times$ denotes a nonzero element. This means that a given pixel in the true image is averaged with its 8 nearest neighbors to create a blurred pixel. The elements are normalized so that they sum to 1.

Let's assume that pixels outside the true image are zero, and imagine forming an artificial image having a white color (which we normalize to 1 here instead of 255) at the first pixel and black (0) everywhere else. Then the blurred image will

look like

$$\begin{matrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

and stacking this, column by column, we get the first column of the blurring matrix **A**.

As Matlab is column oriented, we count the pixels column by column, so the second pixel is the one which is in the second row but first column. So, for the second pixel, the blurred image will look like

$$\begin{matrix} \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

and stacking this, column by column, we get the second column of the blurring matrix **A**.

If we do this for all the pixels of the image we will end up having the block-tridiagonal blurring matrix **A** shown in Figure A.2, where the elements on the main diagonal are equal to the red weight.

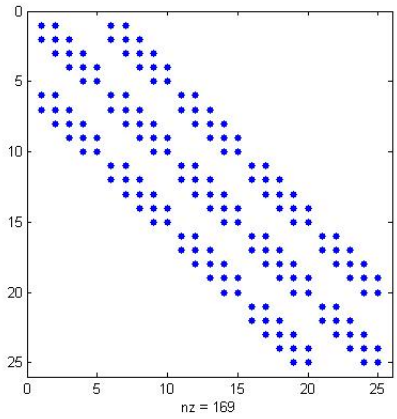

Figure A.2: Blurring matrix $\mathbf{A}$ for an image of size 5×5 with a PSF of size 3×3.

A.1.2 Separable Gaussian blur

Separable blur is a combination of two different blurs: horizontal blur $\mathbf{A}_r$ affects the rows of the image, and vertical blur $\mathbf{A}_c$ affects the columns. As described in Section 3.3.2, the blurring matrix $\mathbf{A}$ is the Kronecker product of the two blurs $\mathbf{A} = \mathbf{A}_r \otimes \mathbf{A}_c$. Since not all matrices are separable, for the purposes of this thesis, we generate a separable blur using the following steps:

1. In the experiments we use a 3×3 Gaussian PSF.

2. We pad $\mathbf{P}$ with zeros so that it has the same dimensions as the image that we blur.

3. We use the Matlab function `[Ar, Ac]=kronDecomp(P, center)` written by Hansen, Nagy, and O'Leary in their HNO package [14], to form a separable approximation to the Gaussian blur. This decomposition is not exact but we set $\mathbf{A} = \mathbf{A}_r \otimes \mathbf{A}_c$.

A.2 Blurring an image and constructing noise

After the blurring matrix $\mathbf{A}$ has been computed, we blur the image $\mathbf{X}$ by computing $\mathbf{Ax}$, where $\mathbf{x}$ is the vector formed by stacking columns of $\mathbf{X}$. But in order to simulate the real case of blurred images, we need to add random noise. For this, we construct a random vector, $\mathbf{e}$, with elements with mean 0 and standard deviation a specified number s. The variance matrix in this case is the identity matrix multiplied by the number s^2, so it is symmetric and invertible. The noisy blurred image is then $\mathbf{b} = \mathbf{Ax} + \mathbf{e}$. For reproducability, the same noise sample is used in all experiments.

Bibliography

[1] F.G. Cantelli. Sulla determinazione empirica delle leggi di probabilita. *Giorn. Ist. Ital. Attuari*, 4:421–424, 1933. Referenced by Kevin Ford, University of Illinois at Urbana-Champaign, (http://www.math.uiuc.edu/ ford/wwwpapers/kol_ engl2.pdf).

[2] S. Cho and S. Lee. Fast motion deblurring. *ACM Transactions on Graphics (TOG)*, 28(5):145, 2009.

[3] J. Chung, M. Chung, and D.P. O'Leary. Designing optimal filters for ill-posed inverse problems. *SIAM J. Sci. Comput.*, 33(6):3132 – 3152, 2011.

[4] J. Chung, M. Chung, and D.P. OLeary. Optimal filters from calibration data for image deconvolution with data acquisition error. *Journal of Mathematical Imaging and Vision*, pages 1–9, 2012.

[5] R.B. D'Agostino. An omnibus test of normality for moderate and large sample size. *Biometrika*, 58:341–348, 1971.

[6] J. Devore. *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2012.

[7] M. Elad and A. Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE Trans. on Image Processing*, 6(12):1646–1658, 1997.

[8] M. Fuhry and L. Reichel. A new Tikhonov regularization method. *Numerical Algorithms*, 59(3):433–445, 2012.

[9] V. Glivenko. Sulla determinazione empirica delle leggi di probabilita. *Giorn. Ist. Ital. Attuari*, 4:92–99, 1933. Referenced by Kevin Ford, University of Illinois at Urbana-Champaign, (http://www.math.uiuc.edu/ ford/wwwpapers/kol_ engl2.pdf).

[10] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, 2(2), 1965.

[11] G.H. Golub, M. Heath, and G. Wahba. Generalized Cross Validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

[12] P.C. Hansen. The discrete Picard condition for discrete ill-posed problems. *BIT Numerical Mathematics*, 30(4):658–672, 1990.

[13] P.C. Hansen. *Discrete Inverse Problems - Insight and Algorithms*. SIAM, Philadelphia, 2010.

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

[15] R.J. Hanson. A numerical method for solving Fredholm integral equations of the first kind using singular values. *SIAM J. Numer. Anal.*, 8(3):616–622, 1971.

[16] C.M. Jargue and A.K. Bera. A test for normality of observations and regression residuals. *International Statistical Review*, 55(2):163–172, 1987.

[17] H.W. Lilliefors. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402, 1967.

[18] F.J. Massey. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[19] V.A. Morozov. On the solution of functional equations by the method of regularization. *Soviet Math. Dokl.*, 7:414–417, 1966.

[20] J.L. Mueller and S. Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. SIAM, Philadelphia, 2012.

[21] J.G. Nagy and D. P. O'Leary. Restoring images degraded by spatially-variant blur. *SIAM J. Sci. Comput.*, 19(4):1063–1082, 1998.

[22] D.P. O'Leary. Near-optimal parameters for Tikhonov and other regularization methods. *SIAM J. Sci. Comput.*, 23(4):1161–1171, 2001.

[23] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[24] B.W. Rust. Truncating the singular value decomposition for ill-posed problems. *NISTIR 6131, U.S. Dept. of Commerce*, 1998.

[25] P.N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Rev.*, 19(3):490–501, 1977.

[26] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.