Technical University of Denmark

Bachelorproject in Mathematics and Technology

Iterative tomographic reconstruction with priorconditioning

February 13, 2017

Hjørdis Amanda Schlüter (s132757)
Bachelor student at DTU

Supervisor:
Professor Per Christian Hansen
Section for Scientific Computing
DTU Compute

Preface

This paper is a Bachelor project in the engineering field "Mathematics and Technology".

Acknowledgements I want to thank my supervisor Per Christian Hansen for guiding and supporting me throughout this project. It was a great pleasure to work with you. And I want to thank Mirko Salewski from the department of Physics at DTU for his cooporation. Here I had the possibility to gain some insight in real world tomographic problems. Finally I would like to thank Mathias Zambach for proofreading this manuscript.

Hjørdis Amanda Schlüter
Copenhagen, 2017

Abstract

For medical and research purposes the quality of tomographic reconstructions is of major importance. Priorconditioning may be a performance improving factor to iterative methods, especially for tomographic reconstruction, but has only been derived for some methods. For this purpose priorconditioned versions of the algebraic iterative methods Kaczmarz and Cimmino are derived in this paper. The priorconditioning appears in form of a priorconditioner matrix that acts as a first or second derivative operator, which causes smoothing of the solution. For this project the methods priorconditioned Kaczmarz and priorconditioned Cimmino were implemented and analyzed in Matlab. Investigations of several test problems obtained by synthetic data showed that the priorconditioned methods performed better than Kaczmarz and Cimmino if the solution was very smooth. Here we find that factors like shape and composition of the objects in the solution, amount of noise in the data and choice of the priorconditioner matrix was important for the quality of the reconstruction.

Contents

List of Symbols

Symbol	Meaning	Dimension
A	Coefficient matrix	$m \times n$
b	Right-hand side/Datavector	$m \times 1$
$\mathcal{K}_k$	Krylov subspace of dimension K	
L	Priorconditioner matrix	$(n-u) \times n$
L_d	Discretization of the $d.$ order derivative in 1D	$(n-u) \times n$
L_d^2	Discretization of the $d.$ order derivative in 2D	$(n-u) \times n$
$L^\dagger$	Moore-Penrose pseudoinverse of L	$n \times (n-u)$
$L^\#$	Oblique pseudoinverse of L	$n \times (n-u)$
W_d	Basisvectors in the null space of L_d (1D)	$n \times u$
W_d^2	Basisvectors in the null space of L_d^2 (2D)	$n \times u$
U	Left singular matrix	$m \times m$
V	Right singular matrix	$n \times n$
x	"Naive" solution	$n \times 1$
$x^{(k)}$	Iteration vector	$n \times 1$
Σ	Diagonal matrix with singular values	$n \times n$
Π	Permutation matrix	$n \times n$
Φ	Matrix containing filter factors	$n \times n$
$\|\cdot\|_2$, $\|\cdot\|_F$	2-norm, Frobenius-norm	
λ	Regularization parameter (Tikhonov)	
ω	Regularization parameter (Landweber)	

Scalar explanation:

Symbol	Meaning
N	Number of discretization intervals used to obtain x
m	Number of data points in b
u	Dimension of the null space of L

1 Introduction and Motivation

In tomography the information inside a body or an object is reconstructed using imaging techniques. One of the best known applications is Computed Tomography (CT) scan for medical use, where X-Rays are used to scan a body. Here the information from the set-up of the X-Rays and data in form of damping of the X-Rays are used to formulate a mathematical problem, that can be solved with iterative tomographic methods. These types of problems are called inverse problems, arising when recovering "interior" or "hidden" information from "outside" [6]. As the mathematical problems in tomography usually are very ill-conditioned, regularization methods are needed to obtain a solution. Here it can be advantageous to use algebraic iterative methods since these methods are well suited for tomography problems.

This is where priorconditioning enters the field: Priorconditioning is a way to optimize iterative methods using prior information about the solution image. Especially in tomography-problems priorconditioning derived for algebraic iterative methods may improve the reconstruction. So far priorconditioning has been derived for Tikhonov regularization and the Conjugate Gradient method of Least Squares (CGLS) [6]. Here the priorconditioner matrices act as a first or second derivative operator on the solution, which causes smoothing in the reconstruction. For curiosity reasons and to improve the performance of the algebraic iterative methods, we want to use the approach of priorconditioned CGLS to derive priorconditioned versions of the algebraic iterative methods Cimmino and Kaczmarz. Based on generalized Tikhonov we then want to derive priorconditioner matrices appearing as a first or second derivative operator. To test the performance of Priorconditioned Cimmino (PCimmino) and Priorconditioned Kaczmarz (PKaczmarz) we want to implement the methods in Matlab and analyze how these perform on several test problems.

As a priorconditioner matrix in form of a derivative operator is used, the matrix will cause smoothing of the solution. Thus, we expect that PKaczmarz and PCimmino will produce smooth reconstructions and may improve the performance of Kaczmarz and Cimmino on test problems, where the solution is smooth.

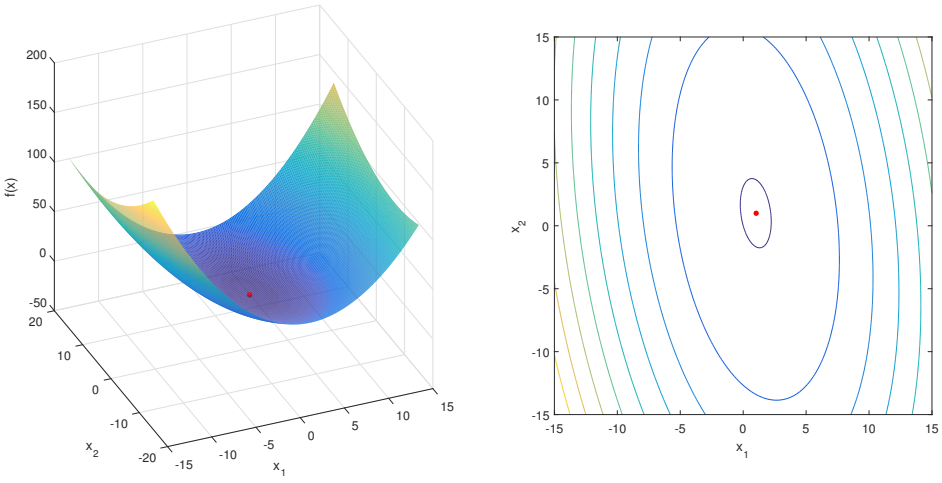

Figure 1: Quadratic form of a vector where the minimum is marked as a red dot.

2 Krylov Subspace Method

I want to motivate priorconditioning by introducing the method of Conjugate Gradients for Least Squares (CGLS) and its priorconditioned version. CGLS is an iterative method that belongs to the Krylov subspace methods. To derive the algebraic definition for this method it is easier first to consider the method of Conjugate Gradients (CG), since CGLS follows the same principle as CG. But since CG is associated with the Method of Steepest Descent I will also present this method.

The derivation of the mentioned methods will follow the principles used by Jonathan Richard Shewchuk [11].

CG is an iterative method to solve large systems of linear equations on the form

$$Ax = b, \tag{1}$$

with respect to x. Where x is an unknown vector, b is a vector containing some data and A is the coefficient matrix of size $n \times n$. Here A must be symmetric and positive-definite.

As a way to solve equation (1) I consider the quadratic form of a vector:

$$f(x) = \frac{1}{2}x^{\mathsf{T}}Ax - b^{\mathsf{T}}x + c, \tag{2}$$

where c is a scalar constant. For A being positive-definite and symmetric, the quadratic form f has the property, that the minimum of this function is exactly the solution to $Ax = b$. This can be illustrated by figure 1 for a given matrix A and data vector b.
Remark that the minimum of f is unique which is caused by the fact that A is a positive-definite matrix and thus has full rank. To investigate this minimum I consider the gradient of the quadratic form

$$\nabla f(x) = \frac{1}{2}A^{\mathsf{T}}x + \frac{1}{2}Ax - b.$$

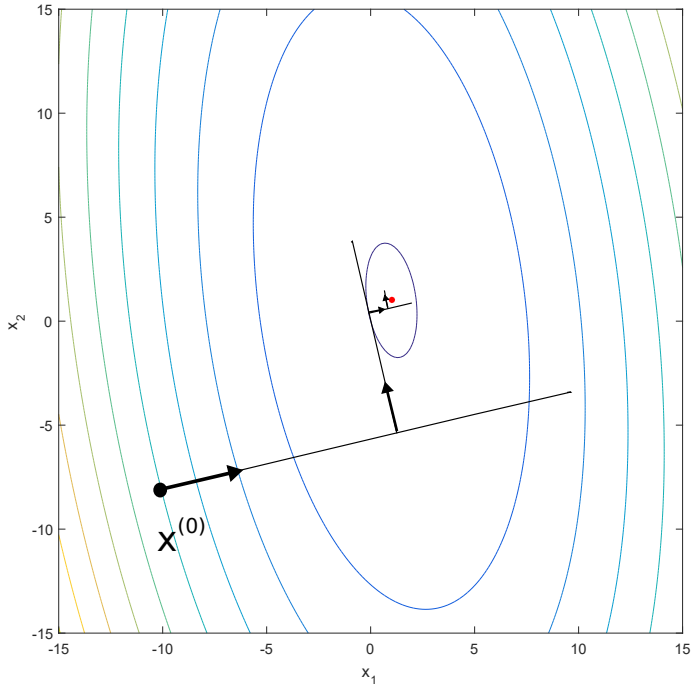

Figure 2: Method of Steepest Descent. The black lines represent the search lines, while the arrows points in the opposite direction of the gradient.

But since A is symmetric the gradient can be rewritten as

$$\nabla f(x) = Ax - b. \tag{3}$$

Setting $\nabla f(x) = 0$ we arrive at equation (1), which is the original problem we want to solve. So for A being symmetric and positive-definite we can find a solution to (1) by minimizing $f(x)$ in (2) with respect to x.

2.1 The Method of Steepest Descent

There are several methods using the minimum of the function f to solve the problem $Ax = b$. For example the iterative Method of Steepest Descent: Starting at any point $x^{(0)}$ we choose the next iteration vector $x^{(1)}$ based on the direction, where f decreases most quickly (this is opposite to the direction $\nabla f(x^{(i)})$). From this direction vector we draw a search line, where we find the next iteration vector at the minimum of the line. The minimum will be found at the point where the gradient vector is orthogonal to the search line. This procedure is illustrated for the previous example in figure 2 for the first iteration vectors $x^{(0)}, x^{(1)}, x^{(2)}, x^{(3)}$ and $x^{(4)}$.

Mathematically the expression for the iteration vector $x^{(k+1)}$ can be written as

$$x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)}, \tag{4}$$

where $r^{(k)}$ is the direction vector and α_k is the step length for the iteration k. Since the direction will be opposite to the direction $\nabla f(x^{(k)})$ we get from equation (3) that

$$r^{(k)} = -\nabla f(x^{(k)}) = b - Ax. \tag{5}$$

The step length α_k is chosen such that the previous and the present gradient vectors are orthogonal to each other, so ${r^{(k)}}^\mathsf{T} r^{(k+1)} = 0$. This leads to following expression for α_k

$$\alpha_k = \frac{{r^{(k)}}^\mathsf{T} r^{(k)}}{{r^{(k)}}^\mathsf{T} A r^{(k)}}. \tag{6}$$

Depending on the problem, this method may converge very slowly towards the naive solution x, since the gradient vectors for two following iterations must always be orthogonal to each other. This often leads to a zigzag path towards the solution. Here the Conjugate Gradient method may be a good alternative.

2.2 The Method of Conjugate Gradients

While the Method of Steepest Descent takes a lot of steps in the same direction, CG tries to avoid this procedure and just take as many steps as there are orthogonal search directions $d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}$. The idea behind CG is to transform a system like the one shown in figure 2 where the contour lines are shaped like ellipses to a system where the contour lines are shaped like circles. In the latter case two orthogonal projections would lead to the solution for the 2-dimensional system if we use the coordinate axes as search directions as seen in figure 3. So in this case the number of steps equals the number of orthogonal search directions.
But for a system, where the contour lines are shaped like circles, two direction vectors being orthogonal ${d^{(k)}}^\mathsf{T} d^{(k+1)} = 0$ means that they are A-conjugate for the elliptical form

$${d^{(k)}}^\mathsf{T} A d^{(k+1)} = 0. \tag{7}$$

In this way we can transfer the properties of the circular form to the elliptical form. The mathematical expression for an iteration in CG should follow the same set-up as for the Method of Steepest Descent

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)},$$

where $r^{(k)}$ is the direction vector and α_k is the step length for the iteration k. But the direction vectors $d^{(k)}$ should be chosen in a smarter way. So instead of satisfying ${d^{(k)}}^\mathsf{T} d^{(k+1)} = 0$ they should now satisfy (7). This leads to the following equations for CG:

$$d^{(0)} = r^{(0)} = b - Ax^{(0)}, \tag{8}$$

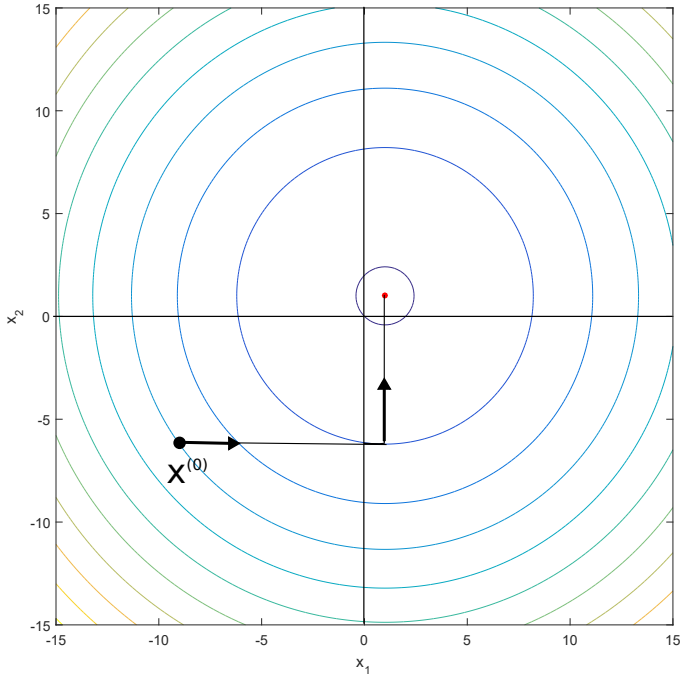

Figure 3: Method of orthogonal projections, where the coordinate axes are used as search lines.

$$\alpha_k = \frac{r^{(k)\mathsf{T}} r^{(k)}}{d^{(k)\mathsf{T}} A d^{(k)}}, \tag{9}$$

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}, \tag{10}$$

$$r^{(k+1)} = r^{(k)} - \alpha_k A d^{(k)}, \tag{11}$$

$$\beta_{k+1} = \frac{r^{(k+1)\mathsf{T}} r^{(k+1)}}{r^{(k)\mathsf{T}} r^{(k)}}, \tag{12}$$

$$d^{(k+1)} = r^{(k+1)} + \beta_{k+1} d^{(k)}. \tag{13}$$

Figure 4 illustrates how the Conjugate Gradients method performs on the example used for figure 1.

Considering the zero vector as the starting point $x^{(0)}$ I want to calculate the iteration vectors $x^{(1)}, x^{(2)}$ and $x^{(3)}$ for the first iterations of CG:

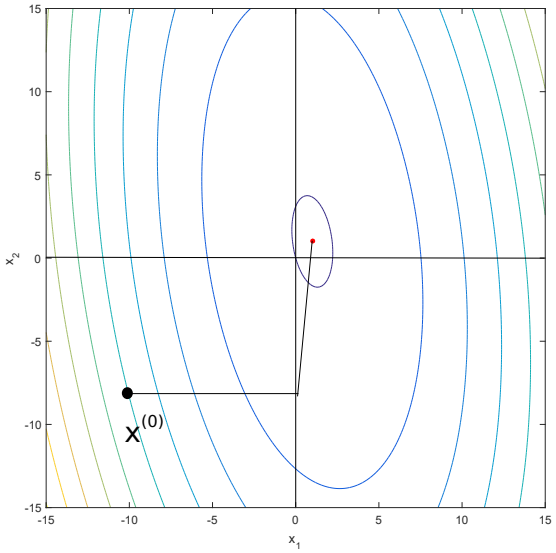

Figure 4: The method of Conjugate Gradients.

$$
\begin{aligned}
x^{(0)} &= 0 \\
d^{(0)} &= r^{(0)} = b - Ax^{(0)} = b,
\end{aligned}
$$

$$
\begin{aligned}
x^{(1)} &= \alpha_0 b \\
r^{(1)} &= r^{(0)} - \alpha_0 A d^{(0)} = b - \alpha_0 Ab \\
d^{(1)} &= r^{(1)} + \beta_1 d^{(0)} = b - \alpha_0 Ab + \beta_1 b = (1-\beta_1)b - \alpha_0 Ab
\end{aligned} \tag{14}
$$

$$
\begin{aligned}
x^{(2)} &= x^{(1)} + \alpha_1 d^{(1)} = \alpha_0 b + \alpha_1((1-\beta_1)b - \alpha_0 Ab) = (\alpha_0 + \alpha_1(1-\beta_1))b - \alpha_1\alpha_0 Ab \\
r^{(2)} &= r^{(1)} - \alpha_1 A d^{(1)} = b - \alpha_0 Ab - \alpha_1 A((1-\beta_1)b - \alpha_0 Ab) \\
&= b - (\alpha_0 + \alpha_1(1-\beta_1))Ab + \alpha_0\alpha_1 A^2 b \\
d^{(2)} &= r^{(2)} + \beta_2 d^{(1)} = b - (\alpha_0 + \alpha_1(1-\beta_1))Ab + \alpha_0\alpha_1 A^2 b + \beta_2((1-\beta_1)b - \alpha_0 Ab) \\
&= (1 + \beta_2(1-\beta_1))b - (\alpha_0 + \alpha_1(1-\beta_1) + \beta_2\alpha_0)Ab + \alpha_0\alpha_1 A^2 b \\
x^{(3)} &= x^{(2)} + \alpha_2 d^{(2)} = (\alpha_0 + \alpha_1(1-\beta_1))b - \alpha_1\alpha_0 Ab \\
&\quad + \alpha_2((1 + \beta_2(1-\beta_1))b - (\alpha_0 + \alpha_1(1-\beta_1) + \beta_2\alpha_0)Ab + \alpha_0\alpha_1 A^2 b) \\
&= (\alpha_0 + \alpha_1(1-\beta_1) + \alpha_2(1 + \beta_2(1-\beta_1)))b \\
&\quad - (\alpha_1\alpha_0 + \alpha_2(\alpha_0 + \alpha_1(1-\beta_1) + \beta_2\alpha_0))Ab + \alpha_0\alpha_1\alpha_2 A^2 b.
\end{aligned} \tag{15, 16}
$$

Common for all parts in the sum of $x^{(1)}, x^{(2)}$ and $x^{(3)}$ are elements on the form

$$\begin{aligned} x^{(1)} &= \_b \\ x^{(2)} &= \_b + \_Ab \\ x^{(3)} &= \_b + \_Ab + \_A^2b, \end{aligned}$$

where the spaces represent constants depending on the given iteration. Thus, all iteration vectors contains linear combinations of $A^{i-1}b$ for $i = 0, \ldots, k$, where k represents the given iteration. For the k^{th} iteration we then get an expression on the form

$$x^{(k)} = \sum_{i=1}^{k} \gamma_i A^{i-1}b = \gamma_1 b + \gamma_2 Ab + \ldots + \gamma_k A^{k-1}b,$$

where the constants γ_i are different for each iteration. Since the matrix A has full rank, the vectors $A^{i-1}b$ for $i = 0, \ldots, k$ will be linearly independent and as A has dimension $n \times n$, the vectors $A^{i-1}b$ will span a subspace of $\mathbb{R}^n$

$$x^{(k)} \in \text{span}\left\{b, Ab, A^2b, \ldots, A^{k-1}b\right\} \subset \mathbb{R}^n. \tag{17}$$

The definition of the Krylov subspace of order k is given by

$$\mathcal{K}_k(A, b) = \text{span}\left\{b, Ab, A^2b, \ldots, A^{k-1}b\right\},$$

so $x^{(k)} \in \mathcal{K}_k(A, b)$.

CG thus finds a solution $x^{(k)}$ satisfying

$$x^{(k)} = \arg\min_x f(x) \qquad \text{s.t } x^{(k)} \in \mathcal{K}_k(A, b), \tag{18}$$

where x is a solution to (1).

2.3 CGLS

In general CG is suited for solving expression (1) when A is square, symmetric and positive definite. If A is of size $m \times n$ and $m > n$ there no longer exists a unique solution x to the problem $Ax = b$ and thus we must solve a Least Squares problem:

$$x = \arg\min_x \|Ax - b\|_2^2 .$$

We can transform the least squares problem to a system of linear equations on the form $Ax = b$ if we investigate $\|Ax - b\|_2^2$:

$$\|Ax - b\|_2^2 = (Ax - b)^\mathsf{T}(Ax - b)$$

$$
\begin{aligned}
&= (x^\mathsf{T} A^\mathsf{T} - b^\mathsf{T})(Ax - b) \\
&= x^\mathsf{T} A^\mathsf{T} Ax - 2x^\mathsf{T} A^\mathsf{T} b + b^\mathsf{T} b.
\end{aligned}
$$

By differentiating $\|Ax - b\|_2^2$ with respect to x and solving when this expression is equal to zero, we are able to find the minimum of $\|Ax - b\|_2^2$:

$$
\frac{d\,\|Ax - b\|_2^2}{dx} = 2A^\mathsf{T} Ax - 2A^\mathsf{T} b.
$$

Setting $\frac{d\,\|Ax - b\|_2^2}{dx} = 0$ the solution to the least squares problem is obtained for $A^\mathsf{T} Ax = A^\mathsf{T} b$. This is the so called normal equation, because $b - Ax$ is normal to the range of A. Now the Least Squares problem can be rewritten as

$$
(A^\mathsf{T} A)x = (A^\mathsf{T} b). \tag{19}
$$

The expression is on the same form as $Ax = b$, where in this case the coefficient matrix is given by $A^\mathsf{T} A$ and the data vector is given by $A^\mathsf{T} b$.

In the following we introduce the Conjugate Gradient method for Least Squares (CGLS). This method solves the Least Squares problem by solving (19). By defining $A^\mathsf{T} A$ as my coefficient matrix and $A^\mathsf{T} b$ as my right-hand side we can follow the same procedure as for CG to find the solution x.
The expression for the quadratic function we want to minimize in this case becomes

$$
f(x) = \frac{1}{2} x^\mathsf{T} A^\mathsf{T} Ax - b^\mathsf{T} Ax + c. \tag{20}
$$

The minimum of f is then the solution to (19) with respect to x.
By replacing all occurrences of the matrix A by $A^\mathsf{T} A$ and b by $A^\mathsf{T} b$, we can arrive at a system, we are able to solve with this method.
The expression for the k^{th} iteration will here be on the form

$$
x^{(k)} = \sum_{i=1}^{k} \gamma_i (A^\mathsf{T} A)^{i-1} A^\mathsf{T} b = \gamma_1 A^\mathsf{T} b + \gamma_2 (A^\mathsf{T} A) A^\mathsf{T} b + \ldots + \gamma_k (A^\mathsf{T} A)^{k-1} A^\mathsf{T} b.
$$

Where $x^{(k)}$ now satisfies

$$
\begin{aligned}
x^{(k)} &\in \operatorname{span}\left\{ A^\mathsf{T} b, (A^\mathsf{T} A) A^\mathsf{T} b, (A^\mathsf{T} A)^2 A^\mathsf{T} b, \ldots, (A^\mathsf{T} A)^{k-1} A^\mathsf{T} b \right\} \\
&= \mathcal{K}_k(A^\mathsf{T} A, A^\mathsf{T} b).
\end{aligned}
$$

Arriving at the minimization problem

$$x^{(k)} = \arg\min_x f(x) \qquad \text{s.t } x^{(k)} \in \mathcal{K}_k(A^\mathsf{T}A, A^\mathsf{T}b),$$

where $f(x)$ is the quadratic function defined in (20) and x solves (19).

This can also be written as

$$x^{(k)} = \arg\min_x \|Ax - b\|_2^2 \qquad \text{s.t } x^{(k)} \in \mathcal{K}_k(A^\mathsf{T}A, A^\mathsf{T}b). \tag{21}$$

2.3.1 PCGLS

To introduce the Preconditioned Conjugate Gradient method for Least Squares (PCGLS), I first take a look at the standard form of the Tikhonov regularization. Here I consider the continuous formulation

$$\begin{aligned}&\min_f \left\{ \left\| \int_0^1 K(s,t)f(t)dt - g(s) \right\|_2^2 + \lambda^2 \|f\|_2^2 \right\} \\ &= \min_f \left\{ \int_0^1 \left(\int_0^1 K(s,t)f(t)dt - g(s) \right)^2 ds + \lambda^2 \int_0^1 (f(t))^2 dt \right\}, \end{aligned} \tag{22}$$

where $\int_0^1 K(s,t)f(t)dt = g(s)$ is known as the first-kind Fredholm integral equation and λ^2 is a regularization parameter. Discretizing epxression (22) will lead to Tikhonov regularization on discrete form

$$\min_x \left\{ \|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2 \right\}.$$

In general this method can be improved by using prior information about the solution to the problem we are dealing with. For Tikhonov regularization this can be done by introducing a matrix L containing the prior information. Leading to the discrete generalized form

$$\min_x \left\{ \|Ax - b\|_2^2 + \lambda^2 \|Lx\|_2^2 \right\}. \tag{23}$$

The matrix L is a finite difference approximation of a derivative of the function $f(t)$ defined in expression (22) [6].

Based on expression (23) it is possible to establish a preconditioned version of CGLS following the same idea as for Tikhonov regularization. But since the expression for CGLS is given by

$$x^{(k)} = \arg\min_x \|Ax - b\|_2^2 \qquad \text{s.t } x^{(k)} \in \mathcal{K}_k(A, b),$$

I want to reformulate (23) by introducing a variable $\xi = Lx$. Now x can be written as $x = L^{-1}\xi$ leading to the expression:

$$\min_\xi \left\{ \left\| (AL^{-1})\xi - b \right\|_2^2 + \lambda^2 \|\xi\|_2^2 \right\}.$$

Here the minimum is found with respect to ξ but rewriting x as $x = L^{-1}\xi$ this expression also gives us the minimum with respect to x. By introducing the variable $\xi = LX$ the problem $Ax = b$ we want to solve becomes $AL^{-1}\xi = b$. Thus we can replace A by AL^{-1} in (21) to obtain a preconditioned solution for CGLS. Now the solution for any iteration k using CGLS is given by

$$\xi^{(k)} = \arg\min_{\xi} \left\| (AL^{-1})\xi - b \right\|_2^2 \qquad \text{s.t } \xi^{(k)} \in \mathcal{K}_k((AL^{-1})^T AL^{-1}, (AL^{-1})^T b), \quad (24)$$

where $x^{(k)} = L^{-1}\xi^{(k)}$. The solution to (24) can be rewritten in terms of $x^{(k)}$. For this purpose I want to check the first 3 iterations of $x^{(k)}$ using PCGLS. From (14),(15) and (16) I know the first iteration vectors for $\xi^{(k)}$ replacing A by $(AL^{-1})^T AL^{-1}$ and b by $(AL^{-1})^T b$

$$\begin{aligned}
x^{(1)} = L^{-1}\xi^{(1)} =& \alpha_0 L^{-1}(AL^{-1})^T b \\
=& \alpha_0 L^{-1}L^{-T}A^T b \\
x^{(2)} = L^{-1}\xi^{(2)} =& (\alpha_0 + \alpha_1(1-\beta_1))L^{-1}(AL^{-1})^T b - \alpha_1\alpha_0 L^{-1}(AL^{-1})^T AL^{-1}(AL^{-1})^T b \\
=& (\alpha_0 + \alpha_1(1-\beta_1))L^{-1}L^{-T}A^T b - \alpha_1\alpha_0 L^{-1}L^{-T}A^T AL^{-1}L^{-T}A^T b \\
x^{(3)} = L^{-1}\xi^{(3)} =& (\alpha_0 + \alpha_1(1-\beta_1) + \alpha_2(1+\beta_2(1-\beta_1)))L^{-1}(AL^{-1})^T b \\
& - (\alpha_1\alpha_0 + \alpha_2(\alpha_0 + \alpha_1(1-\beta_1) + \beta_2\alpha_0))L^{-1}(AL^{-1})^T AL^{-1}(AL^{-1})^T b \\
& + \alpha_0\alpha_1\alpha_2 L^{-1}((AL^{-1})^T AL^{-1})^2(AL^{-1})^T b \\
=& (\alpha_0 + \alpha_1(1-\beta_1) + \alpha_2(1+\beta_2(1-\beta_1)))L^{-1}L^{-T}A^T b \\
& - (\alpha_1\alpha_0 + \alpha_2(\alpha_0 + \alpha_1(1-\beta_1) + \beta_2\alpha_0))L^{-1}L^{-T}A^T AL^{-1}L^{-T}A^T b \\
& + \alpha_0\alpha_1\alpha_2(L^{-1}L^{-T}A^T A)^2 L^{-1}L^{-T}A^T b.
\end{aligned}$$

For the last part of $x^{(3)}$ we have that

$$\begin{aligned}
L^{-1}((AL^{-1})^T AL^{-1})^2(AL^{-1})^T b &= L^{-1}L^{-T}A^T AL^{-1}L^{-T}A^T AL^{-1}L^{-T}A^T b \\
&= (L^{-1}L^{-T}A^T A)^2 L^{-1}L^{-T}A^T b.
\end{aligned}$$

We see that our iteration vectors are on the form

$$\begin{aligned}
x^{(1)} &= ___L^{-1}L^{-\mathsf{T}}A^T b \\
x^{(2)} &= ___L^{-1}L^{-\mathsf{T}}A^T b + ___(L^{-1}L^{-\mathsf{T}}A^T A)L^{-1}L^{-\mathsf{T}}A^T b \\
x^{(3)} &= ___L^{-1}L^{-\mathsf{T}}A^T b + ___(L^{-1}L^{-\mathsf{T}}A^T A)L^{-1}L^{-\mathsf{T}}A^T b + ___(L^{-1}L^{-\mathsf{T}}A^T A)^2 L^{-1}L^{-\mathsf{T}}A^T b.
\end{aligned}$$

Thus $x^{(k)} \in \mathcal{K}_k(L^{-1}L^{-\mathsf{T}}A^T A, L^{-1}L^{-\mathsf{T}}A^T b)$.

Therefore the expression for the Preconditioned Conjugate Gradient method is given by

$$\xi^{(k)} = \arg\min_{\xi} \left\| (AL^{-1})\xi - b \right\|_2^2 \qquad \text{s.t } \xi^{(k)} \in \mathcal{K}_k((AL^{-1})^T AL^{-1}, (AL^{-1})^T b), \quad (25)$$

where ξ solves $(AL^{-1})\xi = b$. And $x^{(k)} = L^{-1}\xi^{(k)}$, with subject to
$x^{(k)} \in \mathcal{K}_k(L^{-1}L^{-\mathsf{T}}A^T A, L^{-1}L^{-\mathsf{T}}A^T b)$.

Since L acts as a preconditioner matrix for this problem we call this method Preconditioned CGLS. In general we can call L a priorconditioner, since this matrix contains the prior information of the solution to the problem [3]. Therefore from now on the term "priorconditioner" will be used for the matrix L.

3 Algebraic iterative methods

As an alternative to the Krylov Subspace method CGLS I want to introduce the algebraic iterative methods. Here we differ between the so called row-action methods, that access one row of the matrix A at a time and methods that access the rows simultaneously. In this section I want to present one method from each of the two groups.

3.1 Kaczmarz's Method

The following theory is based on work of Per Christian Hansen et al. [2].

Kaczmarz's method is a iterative method for solving $Ax = b$ involving computations on one row of A at a time. Therefore we interpret the linear system $Ax = b$ as:

$$\begin{aligned} r_1 \cdot x &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &&= b_1 \\ r_2 \cdot x &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &&= b_2 \\ &\vdots \\ r_m \cdot x &= a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &&= b_m. \end{aligned}$$

Note that r_i for $i = 1, 2, \ldots, m$ is a row vector and each equation $r_i \cdot x = b_i$ defines an affine hyperplane in $\mathbb{R}^n$. If the system $Ax = b$ is consistent and has an unique solution x, this will be the point in $\mathbb{R}^n$ where the affine hyperplanes intersect. The Kaczmarz's method uses an intuitive approach to find this intersection point. To begin with we project the initial vector orthogonal on a given hyperplane, this vector we project orthogonal on another hyperplane, continuing this procedure for all hyperplanes; this is called the Kaczmarz sweep. One sweep is then one iteration compared to simultaneous methods and for the next sweep the last projection vector is used as the start vector. The order in which the hyperplanes are accessed is cyclic and could influence the speed of convergence. Often the row ordering is cyclic in the following way: $1, 2, \ldots, m, 1, 2, \ldots, m, 1, 2, \ldots, m, \ldots$.

The method can be derived algebraically by interpreting the projection of x on a given hyperplane i by taking a step Δx such that $x+\Delta x$ satisfies the equation $b_i - r_i \cdot (x+\Delta x) = 0$:

$$b_i - r_i \cdot (x + \Delta x) = 0 \qquad \Leftrightarrow \qquad r_i \cdot \Delta x = b_i - r_i \cdot x. \tag{26}$$

To find an expression for Δx we must solve equation (26) with respect to Δx

$$\Delta x = (r_i)^\dagger (b_i - r_i \cdot x).$$

Here $(r_i)^\dagger$ denotes the Moore-Penrose pseudoinverse matrix of r_i, where the matrix in this case is a row vector. The Moore-Penrose pseudoinverse of an arbitrary matrix A is the unique matrix that satisfies the four Moore-Penrose conditions [4]:

1. $AA^\dagger A = A$

2. $A^\dagger A A^\dagger = A^\dagger$

3. $\left(AA^\dagger\right)^\mathsf{T} = AA^\dagger$

4. $\left(A^\dagger A\right)^\mathsf{T} = A^\dagger A$

In the following we make sure that $\dfrac{r_i^\mathsf{T}}{\|r_i\|_2^2}$ is the pseudoinverse of r_i and satisfies the Moore-Penrose conditions:

1.
$$r_i r_i^\dagger r_i = r_i \frac{r_i^\mathsf{T}}{\|r_i\|_2^2} r_i = \frac{r_i r_i^\mathsf{T}}{\|r_i\|_2^2} r_i = \frac{\|r_i\|_2^2}{\|r_i\|_2^2} r_i = r_i \tag{27}$$

2.
$$r_i^\dagger r_i r_i^\dagger = \frac{r_i^\mathsf{T}}{\|r_i\|_2^2} r_i \frac{r_i^\mathsf{T}}{\|r_i\|_2^2} = \frac{r_i^\mathsf{T}}{\|r_i\|_2^2} \frac{r_i r_i^\mathsf{T}}{\|r_i\|_2^2} = \frac{r_i^\mathsf{T}}{\|r_i\|_2^2} \frac{\|r_i\|_2^2}{\|r_i\|_2^2} = r_i^\dagger \tag{28}$$

3.
$$\left(r_i r_i^\dagger\right)^\mathsf{T} = \left(r_i \frac{r_i^\mathsf{T}}{\|r_i\|_2^2}\right)^\mathsf{T} = \left(\frac{r_i^\mathsf{T}}{\|r_i\|_2^2}\right)^\mathsf{T} r_i^\mathsf{T} = \frac{r_i}{\|r_i\|_2^2} r_i^\mathsf{T} = r_i \frac{r_i^\mathsf{T}}{\|r_i\|_2^2} = r_i r_i^\dagger \tag{29}$$

4.
$$\left(r_i^\dagger r_i\right)^\mathsf{T} = \left(\frac{r_i^\mathsf{T}}{\|r_i\|_2^2} r_i\right)^\mathsf{T} = r_i^\mathsf{T} \left(\frac{r_i^\mathsf{T}}{\|r_i\|_2^2}\right)^\mathsf{T} = r_i^\mathsf{T} \frac{r_i}{\|r_i\|_2^2} = \frac{r_i^\mathsf{T}}{\|r_i\|_2^2} r_i = r_i^\dagger r_i \tag{30}$$

Thus, $(r_i)^\dagger = \dfrac{r_i^\mathsf{T}}{\|r_i\|_2^2}$ must be the pseudoinverse matrix of r_i, which is uniquely defined. We now get the following expression for the step size Δx:

$$\Delta x = \frac{r_i^\mathsf{T}}{\|r_i\|_2^2}(b_i - r_i \cdot x).$$

And the update of an iteration vector for Kaczmarz's method will then be given by:

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)} = x^{(k)} + \frac{b_i - r_i \cdot x^{(k)}}{\|r_i\|_2^2} r_i^\mathsf{T}.$$

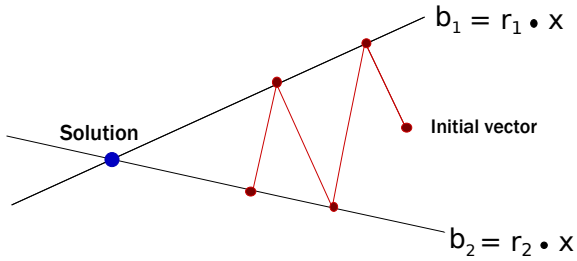

Figure 5: Illustration of Kaczmarz's method for $n = 2$.

Thus we obtain the algebraic formulation of Kaczmarz's method:

$x^{(0)} =$ initial vector
for $k = 0, 1, 2, \ldots$
    $i = k \pmod m$
    $x^{(k+1)} = x^{(k)} + \dfrac{b_i - r_i \cdot x^{(k)}}{\|r_i\|_2^2} r_i^{\mathsf{T}}$
end

Where m iterations corresponds to one sweep over all rows of the matrix A. The algorithm is illustrated for $n = 2$ in figure 5.

3.2 Cimmino's Method

The following theory is based on work of Per Christian Hansen et al. [2].

Cimmino's method is an iterative method for solving $Ax = b$ involving all rows simultaneously. Compared to CGLS this method also solves a Least Squares problem but a weighted one:

$$x = \arg\min_x \left\| M^{1/2}(Ax - b) \right\|_2^2,$$

where M is a diagonal matrix containing weighting factors.

Like Kaczmarz's this method uses orthogonal projections on the affine hyperplanes. But instead of projecting a vector on one hyperplane, the next iteration vector is found by an average between the projection of the previous iteration vector on all hyperplanes

$$\begin{aligned} x^{(k+1)} &= \frac{1}{m}\sum_{i=1}^{m}\left(x^{(k)} + \Delta x^{(k)}\right) \\ &= \frac{1}{m}\sum_{i=1}^{m}\left(x^{(k)} + \frac{b_i - r_i \cdot x^{(k)}}{\|r_i\|_2^2} r_i^{\mathsf{T}}\right) \\ &= x^{(k)} + \frac{1}{m}\sum_{i=1}^{m}\frac{b_i - r_i \cdot x^{(k)}}{\|r_i\|_2^2} r_i^{\mathsf{T}} \end{aligned}$$

This can be rewritten in matrix-form

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \frac{1}{m}\sum_{i=1}^{m}\frac{b_i - r_i \cdot x^{(k)}}{\|r_i\|_2^2} r_i^{\mathsf{T}} \\ &= x^{(k)} + \frac{1}{m}\begin{pmatrix} \frac{r_1^{\mathsf{T}}}{\|r_1\|_2^2} & \frac{r_2^{\mathsf{T}}}{\|r_2\|_2^2} & \cdots & \frac{r_m^{\mathsf{T}}}{\|r_m\|_2^2} \end{pmatrix} \begin{pmatrix} b_1 - r_1 \cdot x^{(k)} \\ b_2 - r_2 \cdot x^{(k)} \\ \vdots \\ b_m - r_m \cdot x^{(k)} \end{pmatrix} \\ &= x^{(k)} + \frac{1}{m}\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \frac{1}{\|r_1\|_2^2} & & & \\ & \frac{1}{\|r_2\|_2^2} & & \\ & & \ddots & \\ & & & \frac{1}{\|r_m\|_2^2} \end{pmatrix} \left(b - \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} x^{(k)} \right) \\ &= x^{(k)} + A^{\mathsf{T}} M^{-1}(b - Ax^{(k)}) \end{aligned}$$

Where we defined the diagonal matrix M =diag$(m\,\|r_i\|_2^2)$. Resulting in the algebraic formulation of Cimmino's method:

$x^{(0)} =$ initial vector
for $k = 0, 1, 2, \ldots$
    $x^{(k+1)} = x^{(k)} + A^{\mathsf{T}} M^{-1}(b - Ax^{(k)})$
end

4 Priorconditioned Versions

We now have investigated priorconditioning for both Tikhonov regularization and CGLS. The algebraic iterative method Cimmino shares similarities with CGLS. Thus, based on the knowledge from PCGLS we derive a priorconditioned version of Cimmino, using this new information to obtain a priorconditioned version of Kaczmarz.

4.1 Priorconditioned Cimmino

The methods Cimmino and CGLS are similar since they both solve a Least Squares problem. Therefore it seems reasonable to follow the same principles in the derivation of Priorconditioned Cimmino (PCimmino) used in PCGLS. I introduce a new variable $\xi = Lx$, where L is the priorconditioner matrix containing prior information about the problem. The linear system of equations I want to solve, $Ax = b$, now becomes $AL^{-1}\xi = b$. I want to derive Cimmino's method such that the system $AL^{-1}\xi = b$ can be solved with respect to ξ. The update of x using Cimmino's method is given by

$$x^{(k+1)} = x^{(k)} + \frac{1}{m}\sum_{i=1}^{m} \frac{b_i - r_i \cdot x^{(k)}}{\|r_i\|_2^2} r_i^{\mathsf{T}}.$$

Replacing r_i by $r_i L^{-1}$ and x by ξ we can obtain the update for ξ:

$$\begin{aligned}
\xi^{(k+1)} &= \xi^{(k)} + \frac{1}{m}\sum_{i=1}^{m} \frac{b_i - r_i L^{-1} \cdot \xi^{(k)}}{\|r_i L^{-1}\|_2^2} \left(r_i L^{-1}\right)^{\mathsf{T}} \\
&= \xi^{(k)} + \frac{1}{m}\begin{pmatrix} \frac{L^{-\mathsf{T}} r_1^{\mathsf{T}}}{\|r_1 L^{-1}\|_2^2} & \frac{L^{-\mathsf{T}} r_2^{\mathsf{T}}}{\|r_2 L^{-1}\|_2^2} & \cdots & \frac{L^{-\mathsf{T}} r_m^{\mathsf{T}}}{\|r_m L^{-1}\|_2^2} \end{pmatrix} \begin{pmatrix} b_1 - r_1 L^{-1} \cdot \xi^{(k)} \\ b_2 - r_2 L^{-1} \cdot \xi^{(k)} \\ \vdots \\ b_m - r_m L^{-1} \cdot \xi^{(k)} \end{pmatrix} \\
&= \xi^{(k)} + L^{-\mathsf{T}} \frac{1}{m} \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \frac{1}{\|r_1 L^{-1}\|_2^2} & & & \\ & \frac{1}{\|r_2 L^{-1}\|_2^2} & & \\ & & \ddots & \\ & & & \frac{1}{\|r_m L^{-1}\|_2^2} \end{pmatrix} \left(b - \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} L^{-1} \xi^{(k)} \right) \\
&= \xi^{(k)} + L^{-\mathsf{T}} A^{\mathsf{T}} \hat{M}^{-1} (b - AL^{-1}\xi^{(k)}).
\end{aligned}$$

Where we defined the diagonal matrix $\hat{M}$ =diag$(m \|r_i L^{-1}\|_2^2)$.

We want to find the solution of $AL^{-1}\xi = b$ with respect to ξ in terms of x. Therefore we multiply L^{-1} by $\xi^{(k+1)}$:

$$L^{-1}\xi^{(k+1)} = L^{-1}\xi^{(k)} + L^{-1}L^{-\mathsf{T}} A^{\mathsf{T}} \hat{M}^{-1} (b - AL^{-1}\xi^{(k)}).$$

Thus, the solution in terms of x will be given by

$$x^{(k+1)} = x^{(k)} + L^{-1}L^{-\mathsf{T}}A^{\mathsf{T}}\hat{M}^{-1}(b - Ax^{(k)}).$$

Therefore the algebraic formulation of PCimmino is on the form

$x^{(0)} =$ initial vector
for $k = 0, 1, 2, \ldots$
    $x^{(k+1)} = x^{(k)} + L^{-1}L^{-\mathsf{T}}A^{\mathsf{T}}\hat{M}^{-1}(b - Ax^{(k)})$
end.

Note that the matrices L^{-1} and $L^{-\mathsf{T}}$ aren't defined explicitly in the implementation, since in case L is large, L^{-1} and $L^{-\mathsf{T}}$ would use too much memory.

4.2 Priorconditioned Kaczmarz

As Cimmino and Kaczmarz are close related to each other by the use of affine hyperplanes, we derive Priorconditioned Kaczmarz (PKaczmarz) following the same principle used in PCimmino. I introduce the new variable $\xi = Lx$ and instead of solving $Ax = b$ I want to solve the linear system $AL^{-1}\xi = b$. I want to derive Kaczmarz's method such that the system $AL^{-1}\xi = b$ can be solved with respect to ξ. The step length $\Delta\xi$ now has to satisfy the equation $b_i - r_i L^{-1} \cdot (\xi + \Delta\xi) = 0$:

$$b_i - r_i L^{-1} \cdot (\xi + \Delta\xi) = 0 \qquad \Leftrightarrow \qquad r_i L^{-1} \Delta\xi = b_i - r_i L^{-1} \cdot \xi.$$

Thus the expression for $\Delta\xi$ will be given by

$$\Delta\xi = \left(r_i L^{-1}\right)^{\dagger} \left(b_i - r_i L^{-1}\xi\right).$$

When we replace r_i by $r_i L^{-1}$ in the equations (27), (28), (29) and (30) we actually get the result, $\left(r_i L^{-1}\right)^{\dagger} = \dfrac{\left(r_i L^{-1}\right)^{\mathsf{T}}}{\|r_i L^{-1}\|_2^2}$, that satisfies the Moore-Penrose conditions. Therefore the step length will be given by

$$\Delta\xi = \frac{\left(r_i L^{-1}\right)^{\mathsf{T}}}{\|r_i L^{-1}\|_2^2} \left(b_i - r_i L^{-1} \cdot \xi\right) = \frac{L^{-\mathsf{T}} r_i^{\mathsf{T}}}{\|r_i L^{-1}\|_2^2} \left(b_i - r_i L^{-1} \cdot \xi\right).$$

And the update of an iteration vector will be on the form

$$\xi^{(k+1)} = \xi^{(k)} + \Delta\xi^{(k)} = \xi^{(k)} + \frac{b_i - r_i L^{-1} \cdot \xi^{(k)}}{\|r_i L^{-1}\|_2^2} L^{-\mathsf{T}} r_i^{\mathsf{T}}.$$

This expression can be transformed to a system that can be rewritten in terms of x. So we multiply L^{-1} by ξ^{k+1} yielding:

$$L^{-1}\xi^{(k+1)} = L^{-1}\xi^{(k)} + L^{-1} \frac{b_i - r_i L^{-1} \cdot \xi^{(k)}}{\|r_i L^{-1}\|_2^2} L^{-\mathsf{T}} r_i^{\mathsf{T}}.$$

This is equivalent with

$$L^{-1}\xi^{(k+1)} = L^{-1}\xi^{(k)} + \frac{b_i - r_i L^{-1} \cdot \xi^{(k)}}{\|r_i L^{-1}\|_2^2} L^{-1} L^{-\mathsf{T}} r_i^{\mathsf{T}}. \tag{31}$$

And since x equals $L^{-1}\xi$, equation (31) can be rewritten as

$$x^{(k+1)} = x^{(k)} + \frac{b_i - r_i \cdot x^{(k)}}{\|r_i L^{-1}\|_2^2} L^{-1} L^{-\mathsf{T}} r_i^{\mathsf{T}}. \tag{32}$$

Here the expression (32) will solve the linear system $AL^{-1}\xi = b$ with respect to ξ and for simplicity this expression is rewritten in terms of x.

Thus, the algebraic formulation of Pkaczmarz will be on the form

$x^{(0)} =$ initial vector
for $k = 0, 1, 2, \ldots$
    $i = k \pmod m$
    $$x^{(k+1)} = x^{(k)} + \frac{b_i - r_i \cdot x^{(k)}}{\|r_i L^{-1}\|_2^2} L^{-1} L^{-\mathsf{T}} r_i^{\mathsf{T}}$$
end.

Here m iterations will correspond to one sweep over all rows of the matrix AL^{-1}.

5 Priorconditioner matrix L

The priorconditioner matrix L contains prior information of the exact solution x to the problem $Ax = b$. As stated in section 2.3.1 common choices of the L-matrix are finite difference approximations to derivatives of the function $f(t)$ which is the continuous form of the vector x. These L-matrices have the property that the solution obtained will be smooth [6].

5.1 Motivation by Tikhonov regularization

Our derivation of the L-matrices is motivated by the expression for generalized Tikhonov regularization. Therefore I want to explain the relation between generalized Tikhonov and the iterative methods PCimmino and PKaczmarz. When considering the expression for generalized Tikhonov regularization:

$$\min_x \left\{ \|Ax - b\|_2^2 + \lambda^2 \|Lx\|_2^2 \right\},$$

the solution will be on the form

$$x_{\lambda,L} = \left(A^\mathsf{T} A + \lambda^2 L^\mathsf{T} L \right)^{-1} A^\mathsf{T} b.$$

Using Generalized Singular Value Decomposition (GSVD) I want to analyze the solution $x_{\lambda,L}$. I decompose A and L into the matrices U_A, Σ_A, Θ and U_L, Σ_L, Θ respectively such that

$$\begin{aligned} A &= U_A \Sigma_A \Theta^{-1} \\ L &= U_L \Sigma_L \Theta^{-1}. \end{aligned}$$

Here U_A and U_L contain the left singular vectors of A and L respectively while the two matrices share the right singular vectors contained in Θ. The matrices Σ_A and Σ_L are non-negative, diagonal and stores the singular values for A, α_i, and the singular values for U, β_i, in the diagonal. Using the decomposition of A and L we can obtain another expression for $x_{\lambda,L}$

$$\begin{aligned} x_{\lambda,L} &= \left(\Theta^{-\mathsf{T}} \Sigma_A^\mathsf{T} U_A^\mathsf{T} U_A \Sigma_A \Theta^{-1} + \lambda^2 \Theta^{-\mathsf{T}} \Sigma_L^\mathsf{T} U_L^\mathsf{T} U_L \Sigma_L \Theta^{-1} \right)^{-1} \Theta^{-\mathsf{T}} \Sigma_A^\mathsf{T} U_A^\mathsf{T} b \\ &= \left(\Theta^{-\mathsf{T}} \Theta^{-1} \left(\Sigma_A^\mathsf{T} \Sigma_A + \lambda^2 \Sigma_L^\mathsf{T} \Sigma_L \right) \right)^{-1} \Theta^{-\mathsf{T}} \Sigma_A^\mathsf{T} U_A^\mathsf{T} b \\ &= \left(\Theta^{-\mathsf{T}} \Theta^{-1} \left(\Sigma_A^\mathsf{T} \Sigma_A + \lambda^2 \Sigma_L^\mathsf{T} \Sigma_L \right) \right)^{-1} \Theta^{-\mathsf{T}} \Sigma_A^\mathsf{T} U_A^\mathsf{T} b \\ &= \operatorname{diag}\left(\frac{1}{\alpha_i^2 + \lambda^2 \beta_i^2} \right) \Theta \Theta^\mathsf{T} \Theta^{-\mathsf{T}} \Sigma_A^\mathsf{T} U_A^\mathsf{T} b \\ &= \operatorname{diag}\left(\frac{\alpha_i^2}{\alpha_i^2 + \lambda^2 \beta_i^2} \right) \Sigma_A^{-1} \Theta U_A^\mathsf{T} b \end{aligned}$$

$$= \Theta\hat{\Phi}\Sigma_A^{-1}U_A^{\mathsf{T}}b, \tag{33}$$

where $\hat{\Phi} = \text{diag}\left(\frac{\alpha_i^2}{\alpha_i^2 + \lambda^2\beta_i^2}\right)$ and $\hat{\Phi}$ is a diagonal matrix that stores the specified coefficients in the diagonal.

To show the similarity between Tikhonov regularization and Cimmino I will now investigate Landweber's method, which like Cimmino is an iterative algebraic method, in which it is easier to express the filtered SVD solution. The minimization problem for Landweber is a simple least squares equation

$$\min_x \|Ax - b\|_2^2,$$

and by introducing the priorconditioner matrix L we can obtain the transformed problem

$$\min_\xi \left\| \left(AL^{-1}\right)\xi - b \right\|_2^2. \tag{34}$$

The update of the iteration vector for the standard Landweber is given by

$$x^{(k+1)} = x^{(k)} + \omega A^{\mathsf{T}}\left(b - Ax^{(k)}\right).$$

Where the update for priorconditioned Landweber can be obtained by substituting A by $\overline{A} = AL^{-1}$ and $x^{(k)}$ by $\xi^{(k)}$. I now use Singular Value Decomposition (SVD) to rewrite the solution $\xi^{(k)}$. I decompose $\overline{A}$ in the matrices $\overline{U}$ containing the left singular vectors, $\overline{V}$ containing the right singular vectors and $\overline{\Sigma}$ containing the singular values $\overline{\sigma}_i$ in the diagonal. Now $\overline{A}$ satisfies $\overline{A} = \overline{U}\,\overline{\Sigma}\,\overline{V}^{\mathsf{T}}$. Following the same procedure as for Tikhonov regularization we can express the priorconditioned Landweber solution $x^{(k)}$ in terms of the SVD components. Using the expression for the update of the iteration vector that solves the minimization problem (34), we get

$$\xi^{(k)} = \overline{V}\,\overline{\Phi}\,\overline{\Sigma}^{-1}\,\overline{U}^{\mathsf{T}}b,$$

where $\overline{\Phi}$ =diag$(1 - (1 - \omega\overline{\sigma}_i^2)^k)$. Since $x^{(k)} = L^{-1}\xi^{(k)}$ it follows that

$$x^{(k)} = L^{-1}\overline{V}\,\overline{\Phi}\,\overline{\Sigma}^{-1}\,\overline{U}^{\mathsf{T}}b.$$

To show that there is a relation between Tikhonov and Landweber I use the GSVD components for A and L to express the SVD components for the matrix AL^{-1}:

$$AL^{-1} = U_A\Sigma_A\Theta^{-1}\left(U_L\Sigma_L\Theta^{-1}\right)^{-1} = U_A\Sigma_A\Theta^{-1}\Theta\Sigma_L^{-1}U_L^{\mathsf{T}} = U_A\left(\Sigma_A\Sigma_L^{-1}\right)U_L^{\mathsf{T}}.$$

We see that U_A and U_L are unitary square matrices and $\Sigma_A\Sigma_L^{-1}$ is a non-negative diagonal matrix. Thus, the SVD of AL^{-1} will be on the form $AL^{-1} = U_A\left(\Sigma_A\Sigma_L^{-1}\right)U_L^{\mathsf{T}}$, where $U_A = \overline{U}$, $\Sigma_A\Sigma_L^{-1} = \overline{\Sigma}$ and $U_L^{\mathsf{T}} = \overline{V}^{\mathsf{T}}$. Using these expressions we can now rewrite the solution $x^{(k)}$:

$$\begin{aligned} x^{(k)} &= L^{-1}U_L\text{diag}\left(1-\left(1-\omega\frac{\alpha_i^2}{\beta_i^2}\right)\right)\Sigma_L\Sigma_A^{-1}U_A^\mathsf{T}b \\ &= \Theta\ \Sigma_L^{-1}\ U_L^\mathsf{T}\ U_L\ \Phi\ \Sigma_L\ \Sigma_A^{-1}\ U_A^\mathsf{T}\ b \\ &= \Theta\ \Sigma_L^{-1}\ \Phi\ \Sigma_L\ \Sigma_A^{-1}\ U_A^\mathsf{T}\ b \\ &= \Theta\ \Phi\ \Sigma_A^{-1}\ U_A^\mathsf{T}\ b, \end{aligned}$$

where $\Phi = \text{diag}\left(1-\left(1-\omega\frac{\alpha_i^2}{\beta_i^2}\right)\right)$ and remark that it is not important in which order the diagonal matrices are multiplied by each other. Comparing this expression to the solution $x^{(k)}$ for Tikhonov in terms of the GSVD components (33) we see that the only difference in the solution using generalized Tikhonov and priorconditioned Landweber are the filter factors $\hat{\Phi}$ and Φ.

The conclusion is that generalized Tikhonov regularization does not differ very much from algebraic iterative methods like priorconditioned Landweber or PCimmino. Therefore it is a reasonable assumption that L-matrices derived for generalized Tikhonov regularization may also work for PCimmino. However we do not know if this approach will work for PKaczmarz, since PKaczmarz doesn't access the rows in the A-matrix simultaneously. Thus we are not able to express the solution in terms of the SVD components and therefore it is not possible to compare the PKaczmarz solution to the generalized Tikhonov solution. Note that we use the L-matrices that we derive for Tikhonov, but we don't know if these matrices are a good choice for Kaczmarz's method.

5.2 In one dimension

To obtain finite difference approximations I investigate $f(\bar{t}+\Delta t)$ and $f(\bar{t}-\Delta t)$ at a fixed point $\bar{t}$ using Taylor expansions [8]. Where Δt is the step size between the discretization points that are used to obtain x.

$$f(\bar{t}+\Delta t) = f(\bar{t}) + \Delta t\cdot f'(\bar{t}) + \frac{1}{2}(\Delta t)^2\cdot f''(\bar{t}) + O(\Delta t^3) \tag{35}$$

$$f(\bar{t}-\Delta t) = f(\bar{t}) - \Delta t\cdot f'(\bar{t}) + \frac{1}{2}(\Delta t)^2\cdot f''(\bar{t}) + O(\Delta t^3). \tag{36}$$

Here $O(\Delta t^3)$ is the truncation error of order 3. Using expression (35) I get the following approximation of the first derivative:

$$f'(\bar{t}) \approx \frac{f(\bar{t}+\Delta t)-f(\bar{t})}{\Delta t}. \tag{37}$$

Using both expressions (35) and (36) we can obtain an approximation of the second derivative at the point $\bar{t}$.

$$f''(\bar{t}) \approx \frac{f(\bar{t}+\Delta t)-2f(\bar{t})+f(\bar{t}-\Delta t)}{(\Delta t)^2}. \tag{38}$$

We discretize the variable t and define a vector T that contains all discretization points:

$$T = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix},$$

where $\Delta t = t_{i+1} - t_i$ for $i = 1, 2, \ldots, N-1$. The discretization of $f(t)$ using this vector T will then give us the vector x:

$$x = f(T) = \begin{bmatrix} f(t_1) \\ f(t_2) \\ \vdots \\ f(t_N) \end{bmatrix}.$$

An approximation of the first derivative of $f(t)$ can now be obtained by using expression (37)

$$f'(t) \approx \frac{1}{\Delta t} \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix} x = L_1\, x.$$

Here L_1 will be the first derivative matrix of size $(N-1) \times N$.

On the same way we can obtain an approximation of the second derivative of $f(t)$ by using equation (38)

$$f''(t) \approx \frac{1}{(\Delta t)^2} \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix} x = L_2\, x.$$

Where L_2 is the second derivative matrix of size $(N-2) \times N$.

For one-dimensional problems the matrices L_1 and L_2 are used as priorconditioners to test the performance of PKaczmarz and PCimmino. Both L_1 and L_2 are not quadratic and therefore they are not invertible. For this reason we introduce the nullspace of L_1 and L_2.

From the rank-nullity Theorem we can deduce the dimension of the null space [1]. The L-matrices are linear transformations $L : x \longrightarrow y$, where $x \in \mathbb{R}^{N\times 1}$, $y \in \mathbb{R}^{(N-d)\times 1}$ and d is the d^{th} derivative of $f(t)$. The rank nullity Theorem gives us the dimension of the null space of L

$$\dim(x) = \dim(Lx) + \dim(\mathcal{N}(L)).$$

Thus,

$$\dim(\mathcal{N}(L)) = \dim(x) - \dim(Lx) = N - (N - d) = d.$$

Therefore the null space of the L-matrices will be spanned by d vectors of size $N \times 1$. I define the matrix W which stores the vectors that span the null space of the L-matrices. For the first derivative matrix d equals one so there must be a vector W_1 such that $L_1 W_1 = 0$. Therefore the function $f(t)$ must satisfy that $f'(t) = 0$. Integrating over $f'(t) = 0$ we get that $f(t) = k$, where $k \in \mathbb{R}$. Therefore I set $k = 1/\sqrt{N}$ and can now obtain the normalized vector W_1 in the 2-norm that spans the null space of L_1:

$$W_1 = \begin{bmatrix} \frac{1}{\sqrt{N}} \\ \frac{1}{\sqrt{N}} \\ \vdots \\ \frac{1}{\sqrt{N}} \end{bmatrix}.$$

For the second derivative matrix d equals two so there must be two vectors w_1^2 and w_2^2 such that $L_2 W_2 = L_2 \begin{bmatrix} w_2^1 & w_2^2 \end{bmatrix} = 0$. Again we see that the constant vector must be a solution, since $f(t) = k$ differentiated twice will give the zero vector. Thus

$$w_1^2 = \begin{bmatrix} \frac{1}{\sqrt{N}} \\ \frac{1}{\sqrt{N}} \\ \vdots \\ \frac{1}{\sqrt{N}} \end{bmatrix}$$

will satisfy that $L_2 w_2^1 = 0$. The other vector can be found by investigating the function $f(t)$. If $f''(t) = 0$, integrating twice over f with respect to t gives us, that $f(t) = k \cdot t + c$ for $k, c \in \mathbb{R}$. Setting $k = 1$ and $c = 0$ we get that $f(t) = t$. With step size $\Delta t = 1$ and start point $t = 1$ discretizing $f(t)$ gives us the vector

$$w_{22}^2 = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ N \end{bmatrix}.$$

I normalize this vector and can now obtain w_2^2: $w_2^2 = \dfrac{w_{22}^2}{\|w_{22}^2\|}$.
We now have that the two vectors w_2^1 and w_2^2 are normalized and linearly independent, so they will span the null space of L_2. Thus $W_2 = \begin{bmatrix} w_1^2 & w_2^2 \end{bmatrix}$.

5.3 In two dimensions

For two-dimensional problems we still want to solve the system $Ax = b$, but x now stores the information of a two-dimensional matrix X. Here X is a discretization of the function $f(t, v)$ which takes two variables as an input. We want to define the discretization vectors T and V. T is the same vector as stated in the previous section. We define a new vector V, that stores the discretization points for v. This vector has the same length as T:

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix},$$

where $\Delta v = v_{i+1} - v_i$ for $i = 1, 2, \ldots, N-1$. Discretizing f with respect to these two vectors will then give us the matrix X:

$$X = f(T, V) = \begin{bmatrix} f(t_1, v_1) & f(t_2, v_1) & \ldots & f(t_N, v_1) \\ f(t_1, v_2) & f(t_2, v_2) & \ldots & f(t_N, v_2) \\ \vdots & \vdots & \ddots & \vdots \\ f(t_1, v_N) & f(t_2, v_N) & \ldots & f(t_N, v_N) \end{bmatrix} = \begin{bmatrix} x_1 & x_{N+1} & \ldots & x_{(N-1)N+1} \\ x_2 & x_{N+2} & \ldots & x_{(N-1)N+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_N & x_{2N} & \ldots & x_{N^2} \end{bmatrix}.$$

X can now be transformed to the vector x by storing all columns of X in a long column vector of size $N^2 \times 1$:

$$x = \mathrm{vec}(X) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \\ x_{N+1} \\ x_{N+2} \\ \vdots \\ x_{2N} \\ \vdots \\ \vdots \\ x_{(N-1)N+1} \\ x_{(N-1)N+2} \\ \vdots \\ x_{N^2} \end{bmatrix}.$$

Before I now introduce the L-matrices for two dimensions I want to return to the definition of the generalized Tikhonov regularization:

$$\min_x \left\{ \|Ax - b\|_2^2 + \lambda^2 \|Lx\|_2^2 \right\}.$$

I start to find the second derivative matrix L_2^2 for two dimensions. I consider the continuous case of x and define the second derivative of $f(t,v)$ to be the sum of f differentiated twice with respect to t and f differentiated twice with respect to v:

$$\nabla^2 f(t,v) = \frac{\partial^2 f}{\partial t^2} + \frac{\partial^2 f}{\partial v^2}.$$

The second derivative of $f(t,v)$ with respect to v on discrete form can now be found by multiplying the L_2-matrix for one dimension with X. This can be explained by the fact that every column vector in the matrix X corresponds to a fixed value of t, while v is varying. Now L_2 multiplied by each column vector will give the second derivative with respect to v for this fixed value of t. A similar principle works for the second derivative of $f(t,v)$ with respect to t on discrete form. We multiply X with the transposed of L. In this way we find the second derivative with respect to t, since v is fixed in each row. We now have

$$\frac{\partial^2 f}{\partial v^2} \approx L_2 X I$$
$$\frac{\partial^2 f}{\partial t^2} \approx I X L_2^\mathsf{T}.$$

I now return to the expression for the generalized Tikhonov regularization. As we want to find the 2-norm of Lx we have two possibilities:

$$\|Lx\|_2^2 = \left\|L_2 X I + I X L_2^\mathsf{T}\right\|_F^2 \qquad \text{or} \qquad \|Lx\|_2^2 = \|L_2 X I\|_F^2 + \left\|I X L_2^\mathsf{T}\right\|_F^2,$$

where F denotes the Frobenius-norm. The norm $\left\|L_2 X I + I X L_2^\mathsf{T}\right\|_F^2$ has a very large null space due to the fact that there are many functions satisfying $\nabla^2 f = 0$ [6]. Therefore I choose $\|L_2 X I\|_F^2 + \left\|I X L_2^\mathsf{T}\right\|_F^2$ as the 2-norm of Lx squared. Since the Frobenius-norm is a matrix-norm that takes the square root of the sum of all elements squared, this will be the same as storing the column vectors of the matrix in one long column vector and take the 2-norm. Therefore the expression $\|L_2 X I\|_F^2 + \left\|I X L_2^\mathsf{T}\right\|_F^2$ can be further reduced by

$$\|L_2 X I\|_F^2 + \left\|I X L_2^\mathsf{T}\right\|_F^2 = \|\mathrm{vec}(L_2 X I)\|_2^2 + \left\|\mathrm{vec}(I X L_2^\mathsf{T})\right\|_2^2, \tag{39}$$

where $\mathrm{vec}(L_2 X I)$ denotes the long column vector that stores all columns of the matrix $L_2 X I$ below each other. Another way to express $\mathrm{vec}(L_2 X I)$ could be using a Kronecker product [7]. For this purpose I first investigate the vector $\mathrm{vec}(L_2 X I)$.

$$\mathrm{vec}(L_2 X I) = \mathrm{vec}\left(\begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_{N+1} & \dots & x_{(N-1)N+1} \\ x_2 & x_{N+2} & \dots & x_{(N-1)N+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_N & x_{2N} & \dots & x_{N^2} \end{bmatrix}\right)$$

$$= \begin{bmatrix} x_1 - 2x_2 + x_3 \\ x_2 - 2x_3 + x_4 \\ \vdots \\ x_{N-2} - 2x_{N-1} + x_N \\ x_{N+1} - 2x_{N+2} + x_{N+3} \\ \vdots \\ \vdots \\ x_{N^2-2} - 2x_{N^2-1} + x_{N^2} \end{bmatrix}$$

Separating this vector into N column vectors of identical magnitude we see that each of these vectors correspond to L_2 multiplied by one column in the matrix X. Investigating the Kronecker product $I \otimes L_2$ gives

$$I \otimes L_2 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & \ddots \\ & & & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 & 1 \end{bmatrix} & & & \\ & \ddots & & \\ & & \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 & 1 \end{bmatrix} \end{bmatrix}.$$

Multiplying this matrix $I \otimes L_2$ with the vector x yields the same expression as $\text{vec}(L_2XI)$, since L_2 is multiplied by each of the columns in the matrix X. Therefore we know $\text{vec}(L_2XI) = (I \otimes L_2)x$. In a similar way we can use a Kronecker product to express the vector $\text{vec}(IXL_2^\mathsf{T})$. Here $(L_2 \otimes I)x$ will satisfy that $\text{vec}(IXL_2^\mathsf{T}) = (L_2 \otimes I)x$. For further details and illustrations see Appendix.

We can now rewrite expression (39):

$$\|\text{vec}(L_2XI)\|_2^2 + \left\|\text{vec}(IXL_2^\mathsf{T})\right\|_2^2 = \|(I \otimes L_2)x\|_2^2 + \|(L_2 \otimes I)x\|_2^2 .$$

Since the sum between the 2-norm squared of two vectors will be the same as defining a new long vector that contains the two vectors one below the other and taking the 2-norm squared of this vector, we have that:

$$\|(I \otimes L_2)x\|_2^2 + \|(L_2 \otimes I)x\|_2^2 = \left\| \begin{pmatrix} (I \otimes L_2)x \\ (L_2 \otimes I)x \end{pmatrix} \right\|_2^2 .$$

Then we can define a new matrix $\hat{L}$ such that

$$\left\| \begin{pmatrix} (I \otimes L_2) \\ (L_2 \otimes I) \end{pmatrix} x \right\|_2^2 = \left\| \hat{L} x \right\|_2^2 .$$

Using QR-factorization we can now decompose the matrix $\hat{L}$ into two matrices Q_2 and R_2: $\hat{L} = Q_2 R_2$. Here Q_2 satisfies that $Q_2^\mathsf{T} Q_2 = I$ and R_2 is a upper triangular matrix that has the same rank and null space like $\hat{L}$. We now get

$$\left\| \hat{L} x \right\|_2^2 = \| Q_2 R_2 x \|_2^2 = (Q_2 R_2 x)^\mathsf{T} (Q_2 R_2 x) = x^\mathsf{T} R_2^\mathsf{T} Q_2^\mathsf{T} Q_2 R_2 x = x^\mathsf{T} R_2^\mathsf{T} R_2 x = \| R_2 x \|_2^2 ,$$

since $Q_2^\mathsf{T} Q_2 = I$. But if $\left\| \hat{L} x \right\|_2^2 = \| R_2 x \|_2^2$, we can choose our second derivative matrix L_2^2 for two-dimensional problems to equal the matrix R_2 that is composed by QR-factorization of the matrix $\hat{L} = \begin{pmatrix} (I \otimes L_2) \\ (L_2 \otimes I) \end{pmatrix}$.

The first derivative matrix L_1^2 for two-dimensional problems can be found on a similar way. In the continuous case of x I define the first derivative of $f(t, v)$ to be the sum between the partial derivative with respect to t and the partial derivative with respect to v:

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial v}.$$

But the first derivative with respect to t and v will be similar to the expressions for the second derivative, where L_2 is replaced by L_1 to obtain the first derivative in each direction:

$$\frac{\partial f}{\partial v} \approx L_1 X I$$
$$\frac{\partial f}{\partial t} \approx I X L_1^\mathsf{T}.$$

Therefore the derivation of L_1^2 will follow the same principle as for L_2^2, with the main difference that L_2 is replaced by L_1. Thus, the first derivative matrix L_1^2 can be obtained by QR-factorization of the matrix $\hat{L}_1 = \begin{pmatrix} (I \otimes L_1) \\ (L_1 \otimes I) \end{pmatrix}$, where $\hat{L}_1 = Q_1 R_1$. Therefore

$$\left\| \hat{L}_1 x \right\|_2^2 = \| R_1 x \|_2^2 ,$$

since $Q_1^\mathsf{T} Q_1 = I$. But then the first derivative matrix in two dimensions L_1^2 will be given by $L_1^2 = R_1$.

I want to define the null space of the L-matrices in two dimensions. We have that $L_1^2 = R_1$ and $L_2^2 = R_2$, where the R-matrices are upper triangular matrices of size $N^2 \times N^2$. But for R_1 the last row is equal to zero, while for R_2 the last four rows are equal to zero. Thus, we have that

$$\dim(\mathcal{N}(L_1^2)) = dim(x) - dim(L_1^2 x) = N^2 - (N^2 - 1) = 1.$$

And

$$\dim(\mathcal{N}(L_2^2)) = dim(x) - dim(L_2^2 x) = N^2 - (N^2 - 4) = 4.$$

For the first derivative I consider the continuous case of x, the function $f(t, v)$. This function must satisfy that $\frac{\partial f}{\partial t} + \frac{\partial f}{\partial v} = 0$. Integrating over f gives $f(t, v) = c_1 + c_2 = c$. Setting $c = 1/N$ we can now obtain the normalized vector W_1^2 of size $N^2 \times 1$ that spans the null space of L_1^2:

$$W_1^2 = \begin{bmatrix} \frac{1}{N} \\ \frac{1}{N} \\ \vdots \\ \frac{1}{N} \end{bmatrix}.$$

For the second derivative $f(t, v)$ there will be four vectors that span the null space W_2^2 of L_2^2. Here the function f must satisfy $\nabla^2 f(t, v) = \frac{\partial^2 f}{\partial t^2} + \frac{\partial^2 f}{\partial v^2} = 0$. Integrating twice over f gives $f(t, v) = c_1 + c_2 + c_3 t + c_4 v + c_5 tv = c + c_3 t + c_4 v + c_5 tv$. We see that there are four different groups of functions that appear in the expression of $f(t, v)$. Thus we can obtain four different vectors that span the null space of L_2^2. Again we see that the constant vector $w_1^2 = W_1^2$ is part of the null space. When defining $c = c_4 = c_5 = 0$ and $c_3 = 1$ we obtain the function $f(t, v) = t$. Discretizing $f(t, v)$ with step size $\Delta t = 1$ and start point $t = 1$ gives us the $N^2 \times 1$ vector

$$w_{22}^2 = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ N \\ \vdots \\ \vdots \\ 1 \\ 2 \\ \vdots \\ N \end{bmatrix}.$$

Normalizing this vector I can obtain the vector $w_2^2 = \frac{w_{22}^2}{\|w_{22}^2\|_2}$. The function $f(t, v) = v$ will also be a solution to $\nabla^2 f(t, v) = 0$ and discretizing this function with $\Delta v = 1$ and start point $v = 1$ gives us the vector

$$w_{33}^2 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 2 \\ 2 \\ \vdots \\ 2 \\ \vdots \\ \vdots \\ N \\ N \\ \vdots \\ N \end{bmatrix}.$$

Normalizing u yields another vector in the null space of L_2^2: $w_3^2 = \frac{w_{33}^2}{\|w_{33}^2\|_2}$. The last vector in the null space can now be obtained from the function $f(t, v) = tv$. Discretizing in t- and v-direction the same way as before, we get

$$w_{44}^2 = \begin{bmatrix} 1 \cdot 1 \\ 2 \cdot 1 \\ \vdots \\ N \cdot 1 \\ 1 \cdot 2 \\ 2 \cdot 2 \\ \vdots \\ N \cdot 2 \\ \vdots \\ \vdots \\ 1 \cdot N \\ 2 \cdot N \\ \vdots \\ N \cdot N \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ N \\ 2 \\ 4 \\ \vdots \\ 2N \\ \vdots \\ \vdots \\ N \\ 2N \\ \vdots \\ N^2 \end{bmatrix}.$$

We can obtain w_4^2 by normalizing this vector: $w_4^2 = \frac{w_{44}^2}{\|w_{44}^2\|_2}$. Now the null space of L_2^2 will be given by $W_2^2 = \begin{bmatrix} w_1^2 & w_2^2 & w_3^2 & w_4^2 \end{bmatrix}$ since all vectors w_1^2, w_2^2, w_3^2 and w_4^2 are linearly independent and normalized.

5.4 Inverse of L

For both PKaczmarz and PCimmino the inverse of the matrix L is used in the update of the iteration vector $x^{(k)}$. But considering the L-matrices for one and two dimensions as defined above, we see that these matrices are not square and thus, we can not easily invert them. Therefore I want to specify how we deal with L-matrices that are rectangular. Here I focus on the expression for generalized Tikhonov regularization:

$$\min_x \left\{ \|Ax - b\|_2^2 + \lambda^2 \|Lx\|_2^2 \right\}. \tag{40}$$

5.4.1 Case 1: L is square

Considering the case where $L \in \mathbb{R}^{n \times n}$ is a square matrix and L has full rank: $\text{rank}(L) = n$, the variable $\xi = Lx$ can be introduced and (40) can be rewritten as

$$\min_\xi \left\{ \left\| (AL^{-1})\xi - b \right\|_2^2 + \lambda^2 \|\xi\|_2^2 \right\} \tag{41}$$

and solved with respect to ξ. To find the solution with respect to x, we must multiply L^{-1} by ξ: $x = L^{-1}\xi$.

5.4.2 Case 2: L is rectangular with more rows than columns

In this case I consider a rectangular matrix $L \in R^{p \times n}$, where $p > n$ and $\text{rank}(L) = n$. My idea is to introduce the Moore-Penrose pseudoinverse of L as defined in section 3.1 and replace L^{-1} by $L^\dagger$ in expression (41). I use QR-factorization to decompose L into the matrices Q and R: $L = QR$. Now I want to check if the matrix $R^{-1}Q^\mathsf{T}$ satisfies the conditions to be the pseudoinverse of L.

1. $LL^\dagger L = QRR^{-1}Q^\mathsf{T}QR = QR = L$

2. $L^\dagger LL^\dagger = R^{-1}Q^\mathsf{T}QRR^{-1}Q^\mathsf{T} = R^{-1}Q^\mathsf{T} = L^\dagger$

3. $\left(LL^\dagger\right)^\mathsf{T} = \left(QRR^{-1}Q^\mathsf{T}\right)^\mathsf{T} = I^\mathsf{T} = I = QRR^{-1}Q^\mathsf{T} = LL^\dagger$

4. $\left(L^\dagger L\right)^\mathsf{T} = \left(R^{-1}Q^\mathsf{T}QR\right)^\mathsf{T} = I^\mathsf{T} = I = R^{-1}Q^\mathsf{T}QR = L^\dagger L$

We see that all four Moore-Penrose conditions are satisfied so $R^{-1}Q^\mathsf{T}$ must be the pseudoinverse of L. Thus we replace L^{-1} by $L^\dagger = R^{-1}Q^\mathsf{T}$ in expression (41) and can now obtain the minimization problem we want to solve

$$\min_\xi \left\{ \left\| (AR^{-1}Q^\mathsf{T})\xi - b \right\|_2^2 + \lambda^2 \|\xi\|_2^2 \right\}. \tag{42}$$

This can be simplified by introducing the variable $\hat{\xi} = Q^\mathsf{T}\xi$, where $\xi = Q\hat{\xi}$. We have that

$$\|\xi\|_2^2 = \left\|Q\hat{\xi}\right\|_2^2 = \left\|\hat{\xi}\right\|_2^2.$$

We can now replace ξ by $Q\hat{\xi}$ in expression (42)

$$\min_{\xi} \left\{ \left\|(AR^{-1})\hat{\xi} - b\right\|_2^2 + \lambda^2 \left\|\hat{\xi}\right\|_2^2 \right\}, \tag{43}$$

where we have the relation $\left\|\hat{\xi}\right\|_2^2 = \|\xi\|_2^2$. And $\hat{\xi}$ can be rewritten as

$$\hat{\xi} = Q^{\mathsf{T}}\xi = Q^{\mathsf{T}}Lx = Q^{\mathsf{T}}QRx = Rx \qquad \Leftrightarrow \qquad x = R^{-1}\hat{\xi}.$$

We can now solve expression (43) with respect to $\hat{\xi}$ and find the solution in terms of x: $x = R^{-1}\hat{\xi}$.

5.4.3 Case 3: L is rectangular with fewer rows than columns

This section is based on theory derived by Per Christian Hansen [6].

In this case I consider $L \in \mathbb{R}^{p\times n}$ being a rectangular matrix where $p < n$ and $\text{rank}(L) = p$. From the rank-nullity Theorem we know that these L-matrices have a non-empty null space. Therefore any Tikhonov solution x to (40) can be splitted into two components: A component $x_0 \in \mathcal{N}(L)$ that belongs to the null space of L and a component z that does not. Now x equals $z + x_0$ and the Tikhonov problem (40) becomes:

$$\left\{ \|Az + Ax_0 - b\|_2^2 + \lambda^2 \|Lz\|_2^2 \right\}.$$

Note that $Lx_0 = 0$ since $x_0 \in \mathcal{N}(L)$. By requiring that Az is orthogonal to Ax_0 we can obtain two independent problems that can be solved with respect to z and x_0:

$$\min_z \left\{ \|Az - b\|_2^2 + \lambda^2 \|Lz\|_2^2 \right\} \qquad \min_{x_0} \|Ax_0 - b\|_2^2.$$

Since $Az \perp Ax_0$ we can split $\mathbb{R}^n$ into the two components $\mathcal{N}(L)$ and $\mathcal{N}(L)^{\perp_A}$ so that $z \in \mathcal{N}(L)^{\perp_A}$ and $x_0 \in \mathcal{N}(L)$.

The derivation of the pseudo inverse of L in this case is very technical and requires definitions about oblique projectors and oblique pseudo inverses [6]. Therefore I will skip this part and only mention information most relevant to my project.

By introducing oblique projectors associated to $\mathcal{N}(L)^{\perp_A}$ and $\mathcal{N}(L)$ we can rewrite our minimization problems. Using the matrix W that spans the null space of L we can obtain an expression of x_0:

$$x_0 = W(AW)^{\dagger}b.$$

From this definition of x_0 we can derive an expression for the oblique projector that is associated to the null space of L. Using this information we end up with an expression for $L^{\#}$ such that our minimization problem for Tikhonov is on the form:

$$\min_{\xi} \left\{ \left\| (AL^{\#})\xi - b \right\|_2^2 + \lambda^2 \left\| \xi \right\|_2^2 \right\}. \tag{44}$$

The expression for $L^{\#}$ will be given by:

$$L^{\#} = (I - W(AW)^{\dagger}A)L^{\dagger},$$

where $L^{\dagger}$ depends on the problem and the structure of L. In this case we consider L having full rank and therefore L can be splitted into $L = (L_{left}, L_{right})$, where L_{left} is square and has full rank. Thus, $L^{\dagger}$ will in this case be on the form:

$$L^{\dagger} = \begin{pmatrix} L_{left}^{-1} \\ 0 \end{pmatrix}.$$

And an expression for $(AW)^{\dagger}$ can be obtained by QR-factorization. Decomposing AW into Q_{AW} and R_{AW} we get: $AW = Q_{AW}R_{AW}$. In case 2 we derived the Moore-Penrose pseudo inverse for the matrix L which was given by $L^{\dagger} = R^{-1}Q^{\mathsf{T}}$. The same principle will work in this case and thus, $(AW)^{\dagger} = R_{AW}^{-1}Q_{AW}^{\mathsf{T}}$. Now we arrive at the following expression for $L^{\#}$:

$$L^{\#} = (I - W(R_{AW}^{-1}Q_{AW}^{\mathsf{T}})A) \begin{pmatrix} L_{left}^{-1} \\ 0 \end{pmatrix}.$$

We can now solve the Tikhonov problem (44) with respect to ξ and find the solution in terms of x: $x = L^{\#}\xi$.

For the second derivative matrix in two dimensions, L_2^2, we encounter problems when separating the matrix into L_{left} and L_{right} and inverting the quadratic matrix L_{left}, since L_2^2 has a structure as shown in the left part of figure 6. We see that most elements in the diagonal and upper diagonal are nonzero, while some elements are equal to zero. Thus L_{left} will be rank-deficient and have no finite inverse. Considering N discretization intervals for x, the zero elements appear in the diagonal as number $N^2 - N - 1$ and in the upper diagonal as number $N^2 - N$. To get rid of the zero elements in the diagonals we introduce a permutation matrix Π that repositions column number $N^2 - N - 1$ and $N^2 - N$ to the last columns in the matrix, while the last $N^2 - N + 1$ columns are moved two columns to the left. This proces is illustrated in figure 6. I define $\overline{L} = L\Pi$ and can now separate this matrix into $\overline{L}_{left}$ and $\overline{L}_{right}$. $\overline{L}^{\dagger}$ will here be given by:

$$\overline{L}^{\dagger} = \begin{pmatrix} \overline{L}_{left}^{-1} \\ 0 \end{pmatrix}.$$

To obtain the pseudo inverse with respect to L we permute $\overline{L}^{\dagger}$: $L^{\dagger} = \Pi\overline{L}^{\dagger}$. Using this expression for $L^{\dagger}$ we can follow the same principle as for L-matrices that didn't have zeros in the diagonal to obtain the oblique pseudo inverse $L^{\#}$:

$$L^{\#} = (I - W(R_{AW}^{-1}Q_{AW}^{\mathsf{T}})A)\Pi \begin{pmatrix} \overline{L}_{left}^{-1} \\ 0 \end{pmatrix}.$$

This approach opens the possibility to work with the matrix L_2^2. Note that this only works for this specific L-matrix. Further approaches could aim toward finding a procedure that works for arbitrary L-matrices.

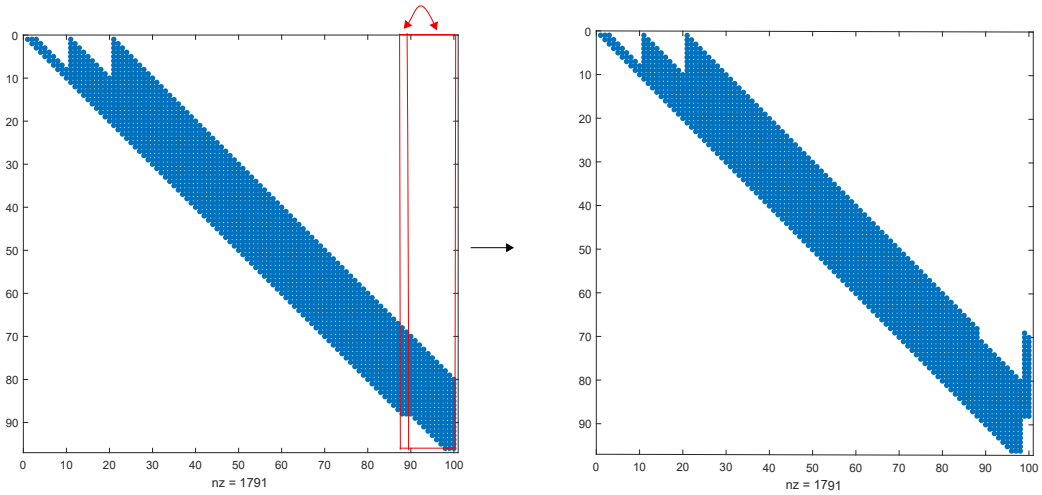

Figure 6: Nonzero elements in the L_2^2 matrix to the left and in the $\overline{L}^{\dagger}$-matrix obtained by using a permutation matrix to the right.

6 Matlab implementation

The methods PCimmino and PKaczmarz are implemented in Matlab named `PCimmino.m` and `PKaczmarz.m`. These methods use the function `get_l.m` from the AIR Tools package to define the L-matrices for one-dimensional problems and the function `get_L2.m` to define the L-matrices for two-dimensional problems. Multiplications with the inverse and the transpose of the inverse of L in one dimension are performed using the functions `lsolve.m` and `ltsolve.m` from the AIR Tools package. For two dimensions Per Christian Hansen added modifications for the second derivative matrix as derived in section 5.4.3, resulting in the functions `lsolve2D.m` and `ltsolve2D.m`.

Since I don't use large test problems, where the resolution is high, I define the matrix $L^{\#}\left(L^{\#}\right)^{\mathsf{T}} A^{\mathsf{T}}$ explicitly to save computation time. A more general approach would be to calculate $L^{\#}\left(L^{\#}\right)^{\mathsf{T}} A^{\mathsf{T}}$ and $L^{\#}\left(L^{\#}\right)^{\mathsf{T}} r_i^{\mathsf{T}}$ for PCimmino and PKaczmarz respectively in each iteration to save memory space. In this case it would be possible to use the method on larger test problems than possible with my approach. Note that for the last approach the computation time will be K times longer than for the my approach. Here K is the number of iterations for PCimmino and the number of sweeps over the rows of $AL^{\#}$ for PKaczmarz. Also note that longer computation time would especially be a disadvantage for PCimmino since PCimmino, particularly for low noise levels, needs many iterations to obtain a good reconstruction. A more efficient way could therefore be to write code that uses the first approach for small and the last approach for large test problems.

As Matlab is optimized for matrix and vector operations but is slow at performing loops, PKaczmarz performs really slow since we need a `for` loop to sweep over the rows of $AL^{\#}$ and a `for` loop to perform a given number of iterations. Therefore the code could be improved by writing in a programming language that supports loops more effectively.

7 Performance of PKaczmarz & PCimmino

To test the performance of PKaczmarz and PCimmino we created and used different test problems in one and two dimensions. In the following I name Kaczmarz's and Cimmino's method the standard version of PKaczmarz and PCimmino. In general the solution for a reconstruction problem in tomography is a two-dimensional image that represents a slice of a three-dimensional object. Therefore the main purpose of the one-dimensional problems is highlighting the difference between the priorconditioned methods and their standard versions. Another advantage is the simplicity of these problems which makes it easier to analyze the effect of using a priorconditioner matrix.

7.1 Test problems from the "AIR Tools" package

The following test problems are created using the "AIR Tools" package in Matlab [5]. Note that in all cases the datavector b is obtained by multiplying the coefficient matrix A by the known solution x and adding an amount of noise; this phenomena is called "inverse crime" since the same model is used to compute the reconstruction and to get test data [6]. Therefore the following test problems may not be representative for how PKaczmarz and PCimmino operate on real data. However the purpose of the following tests is to give an impression of their performance; if the methods don't yield good results for synthetic data they won't yield good results for real data either.

7.1.1 One dimension

In one dimension $n = 900$ discretization points are used to obtain the test problems. The naive solution x will then have dimension 900×1, the datavector b will have dimension 900×1 and thus, the coefficient matrix A will have dimension 900×900.

7.1.1.1 "Deriv2"

The test problem "Deriv2" is a discretization of a first kind Fredholm equation and the solution appears as a straight line from 0 to 0.0333. As there are no discontinuities in the solution, the problem will be quite smooth. I used Kaczmarz, PKaczmarz, Cimmino and PCimmino to reconstruct the solution at a 1% noise-level, results are illustrated in figure 7. Especially PCimmino was performing well for both the first and second derivative used as L-matrices, but the second derivative actually produced an even better result. PKaczmarz had some problems with the boundary points and produced worse results than Kaczmarz. A similar trend can be recognized at the boundary points for Cimmino and the reason for this behavior of Cimmino can be explained by the basis vectors of the solution. For Landweber's method the solution in terms of the SVD components of the coefficient-matrix A will be on the form

$$x^{(k)} = V\Phi\Sigma^{-1}U^{\mathsf{T}}b,$$

where $\Phi = \text{diag}\Big(1 - \big(1 - \omega\sigma_i^2\big)^k\Big)$. Here V contains the right singular vectors of the matrix A, so the vectors in the columns of V will act as basis vectors for the solution. A similar expression can be obtained for Cimmino by using SVD to decompose A: $A = U\Sigma V^{\mathsf{T}}$. I investigate the first three column vectors of the matrix V, since they act as basis vectors for the Cimmino solution. The vectors are shown in figure 8. We see that the first basis vectors are zero at the boundary points. Since the regularized solution for Cimmino is

dominated by the first basis vectors, this may lead to problems in the reconstruction, as the solution is not zero at both boundary points. So in this case the basis vectors do not match with the structure of the solution. Compared to Cimmino, PCimmino produced way better results because the priorconditioner matrix L can adjust for this factor.

For this test problem PKaczmarz was very sensible to higher noise-levels. At a 0.1% noise-level PKaczmarz was able to produce better results than Kaczmarz especially for the first derivative matrix L, but for higher noise-levels PKaczmarz performed worse.

In conclusion; for this specific test problem especially PCimmino was able to produce much better results than Cimmino. In this case the priorconditioner matrix L is suited for the problem using Cimmino since the priorconditioner can adjust for the problems we get by the structure of the coefficient matrix. In comparison Kaczmarz and particulary PKaczmarz performed worse for higher noise levels than Cimmino/PCimmino.

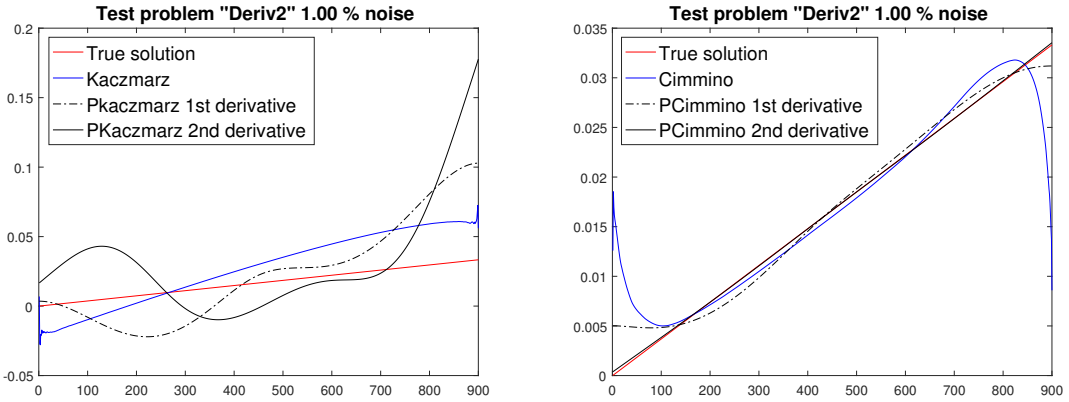

Figure 7: Performance of PKaczmarz and PCimmino on the test problem "Deriv2". PCimmino produces way better solutions than PKaczmarz.

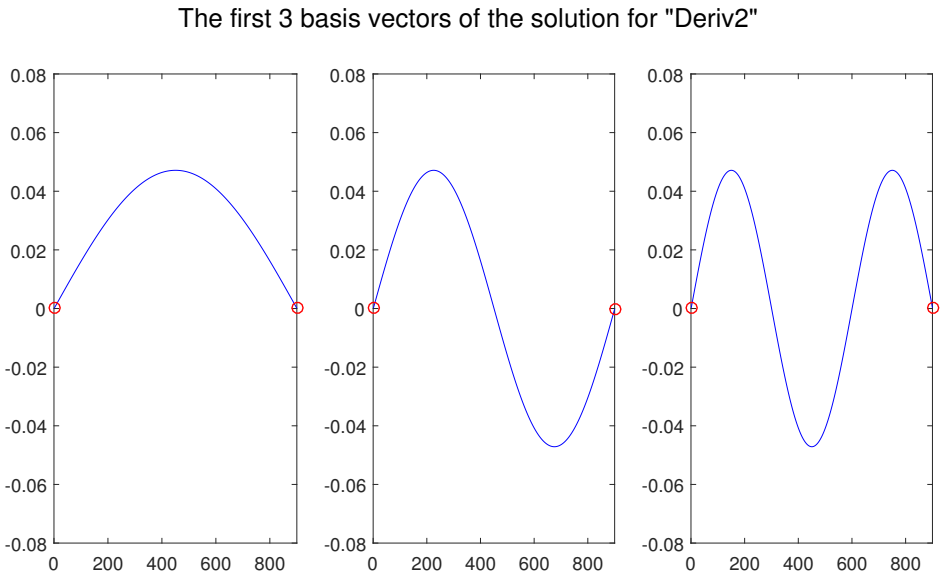

Figure 8: First basisvectors of the solution for the test problem "Deriv2". We see that the vectors are zero at the boundary points.

7.1.1.2 "Gravity"

The test problem "Gravity" is the discretization of a 1D model in gravity surveying where the solution is very smooth. The reconstructions for PKaczmarz and PCimmino at a 1% noise-level are shown in figure 9. In this case all methods produced good solutions, but the choice of the L-matrix is quite important for the quality of the reconstruction when using the priorconditioned methods. For both PKaczmarz and PCimmino the second derivative matrix performed way better than the first derivative. This may be caused by the smoothness of the problem, since we recognized a similar trend for the smooth problem "Deriv2". And the second derivative matrix may cause more smoothing of the solution than the first derivative matrix does. Note that in this case the noise level did not have that much influence on the quality of the reconstruction using PKaczmarz.

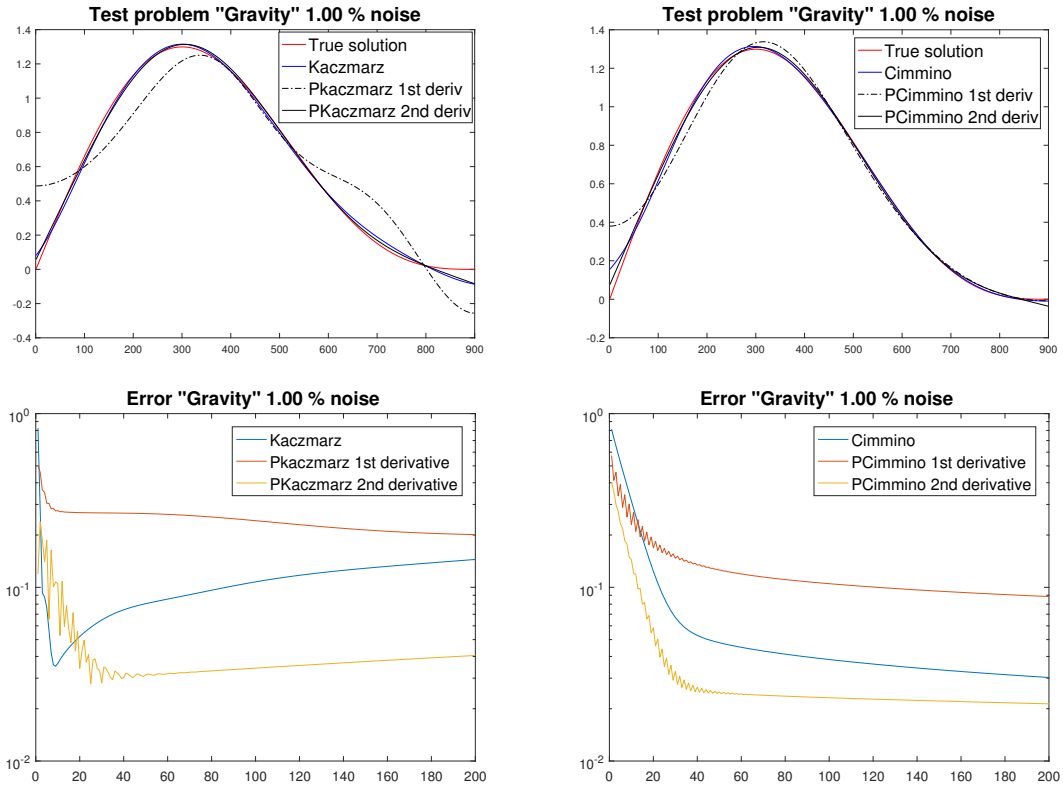

Figure 9: Performance of PKaczmarz and PCimmino on the test problem "Gravity". As both PKaczmarz, PCimmino and their standard versions produced good solutions the error level indicates which method performs best. For the error plots the x-axis shows the number of iterations.

7.1.1.3 "Phillips"

We derived the methods PKaczmarz and PCimmino in the hope that we could get better solutions for smooth problems. Because not all problems are either smooth or piecewise constant this test problem should give an indicate of how PKaczmarz and PCimmino perform on a composition of both types of problems.

The test problem "Phillips" is a discretization of the 'famous' first-kind Fredholm integral equation deviced by D. L. Phillips. The solution is a combination of two constant parts and a very smooth part, where the changes from constant to smooth and reverse order forms two sharp corners. The reconstructions for the four different methods are shown in figure 10. We see that all methods had difficulties to capture the corners in the solution, but definitely Cimmino performed best. The first derivative matrix can improve the solution of Kaczmarz, but PKaczmarz still wasn't as good as Cimmino. Investigating the basis vectors for the solution may give an explanation for the good performance of Cimmino. Here the first three basis vectors are illustrated in figure 11. These vectors fit well with the exact solution: The boundary points are close to zero and especially for the third basis vector the structure matches the solution. So the first basis vectors, that dominates the solution, creates best conditions for Cimmino to give a good reconstruction. As there are piecewise constant parts in the exact solution the priorconditioner matrices

cannot improve the good result of Cimmino, since they cause too much smoothing.

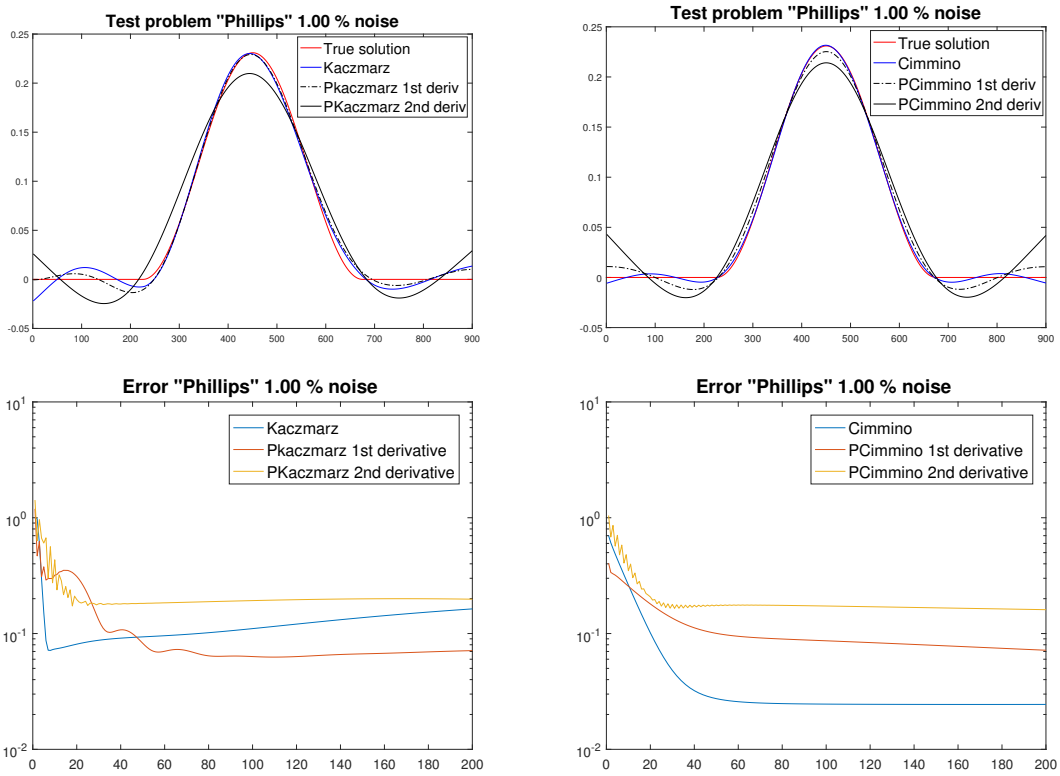

Figure 10: Performance of PKaczmarz and PCimmino on the test problem "Phillips". PKaczmarz produces a slightly better solution than Kaczmarz, while Cimmino performs best. For the error plots the x-axis shows the number of iterations.

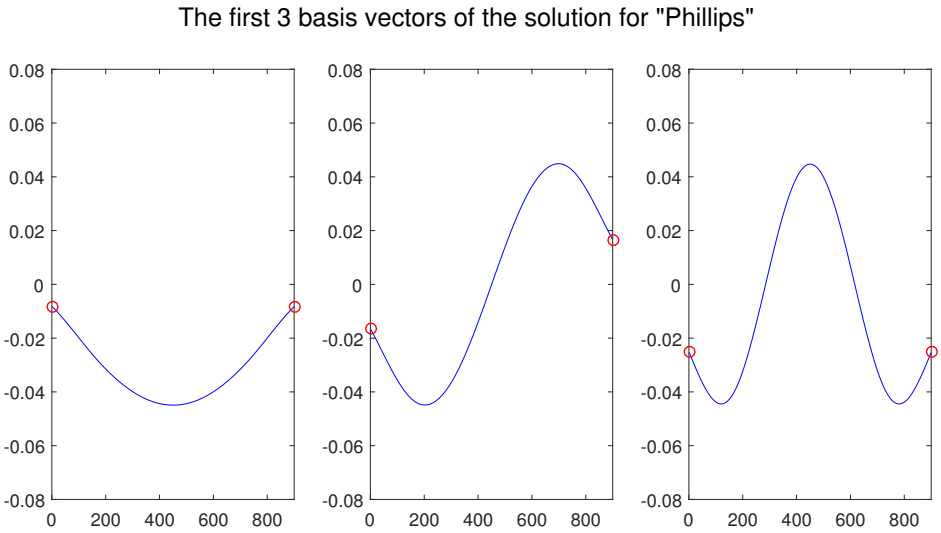

Figure 11: The first three basis vectors of the solution for the test problem "Phillips".

7.1.2 Two dimensions

In the following I define my coefficient matrix A by using the function `paralleltomo.m` from the "AIR Tools" package that uses `round`$(\sqrt{2} * N)$ parallel rays, N discretization intervals and 60 angles from 3° to 180° to simulate a parallel beam test problem and creates the matrix A. As there are `round`$(\sqrt{2} * N)$ parallel rays from 60 different angles, $60 \cdot$ `round`$(\sqrt{2} * N)$ data points are stored in data vector b. The number of discretization intervals is set to $N = 60$ in most of the test problems in this section and thus the solution x is an image consisting of $N \times N$ pixels. Note that x is reshaped to an vector of dimension $N^2 \times 1$ for the calculations. The coefficient matrix A containing information about the set up of the parallel beam geometry is of size $60 \cdot$ `round`$(\sqrt{2} * N) \times N^2$.

7.1.2.1 Phantomgallery: "Smooth"

The phantom "Smooth" is obtained from the `phantomgallery.m` in the "AIR Tools" package using the fourth smoothing function. The solution contains one large object that is very smooth. To identify which priorconditioner matrix works best for this problem we investigate PKaczmarz's and PCimmino's performance for the first and second derivative L-matrix using a low noise-level (0.1%). The colormap "colorcube" is used to highlight the noise in the reconstruction and the result is illustrated in figure 12. We see that Pkaczmarz performed slightly better using the second derivative, while PCimmino only produced a good result for the first derivative. Though PKaczmarz performed good, noise appeared in the corners of the image. The reason for this is the set up of the coefficient matrix: We use 60 different projection angles and 85 parallel rays, but not many rays capture the pixels in the corners. This could also be an explanation why PCimmino gives such a strange reconstruction for the second derivative matrix. Note that the priorconditioners behaved very differently for the two methods. While the second derivative matrix worked fine for PKaczmarz, it didn't work at all for PCimmino.

Now the first derivative matrix for PCimmino and the second derivative matrix for PKaczmarz were used to compare the priorconditioned methods with their standard versions. I tested the methods for three different noise-levels, 0.01%, 0.1% and 1%. The results for Cimmino and PCimmino are illustrated in figure 13. Comparing PCimminos reconstructions with those of Cimmino it is remarkable that PCimmino gives better results for all noise levels. Especially for higher noise levels the difference to Cimmino is significant. But for all noise levels the striking differences between the reconstructions obtained by PCimmino and the exact solution lie in the edges and in the corners of the image. It seems like the priorconditioner matrix creates a smooth solution that make some of the darker orange edges disappear and for higher amount of noise more edges disappear. Apparently at 1% noise green noise pixels appear in the upper right and lower left corner due to the few rays that capture these pixels. Even though the error for the different noise levels indicates that PCimmino yields quite good reconstructions even for higher noise levels.

Figure 14 shows the reconstructions for Kaczmarz and PKaczmarz, here like for Cimmino and PCimmino the priorconditioned method worked better than the standard version for all noise levels. But for higher noise levels the reconstruction got very bad for PKaczmarz and even worse for Kaczmarz. Investigating the reconstructions, PKaczmarz like PCimmino seemed to have problems in the edges and the corners of the image. For

Kaczmarz we can still notice some of the dark orange edges in the reconstruction, while they partially disappear using the priorconditioner matrix. Nevertheless PKaczmarz seemed to perform quite good for lower noise levels on this test problem, but gave a distorted reconstruction for a higher amount of noise like 1%.

In comparison PCimmino worked much better than PKaczmarz for higher noise levels. However PKaczmarz worked slightly better for lower noise levels. Here PKaczmarz performed better, since PCimmino for this test problem required many iterations to obtain a good result. Nevertheless both methods had problems in reconstructing the sections that lie in the corners and edges in the image. This may be a consequence of using priorconditioner matrices: If the sections in the corners and edges of the results are very noisy in the reconstructions obtained by the standard methods, these sections are smoothed by the priorconditioned methods. Due to few rays that capture the corners, the priorconditioned methods may have difficulties in differing between noise and exact solution.

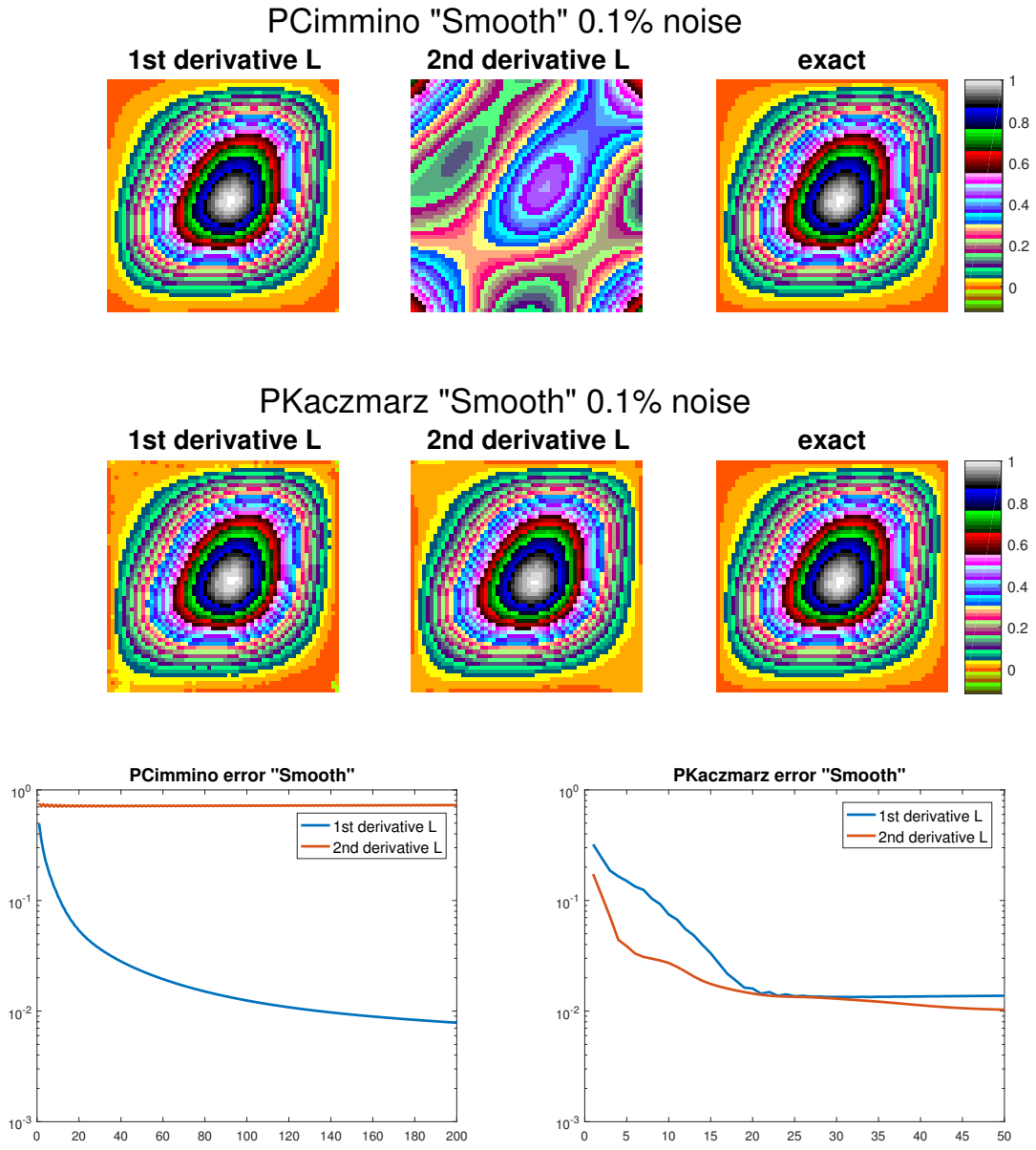

Figure 12: Performance of PCimmino and PKaczmarz on the test problem "Smooth" for different L-matrices. The second derivative matrix works slightly better for PKaczmarz, while it works not at all for PCimmino. For the error plots the x-axis shows the number of iterations.

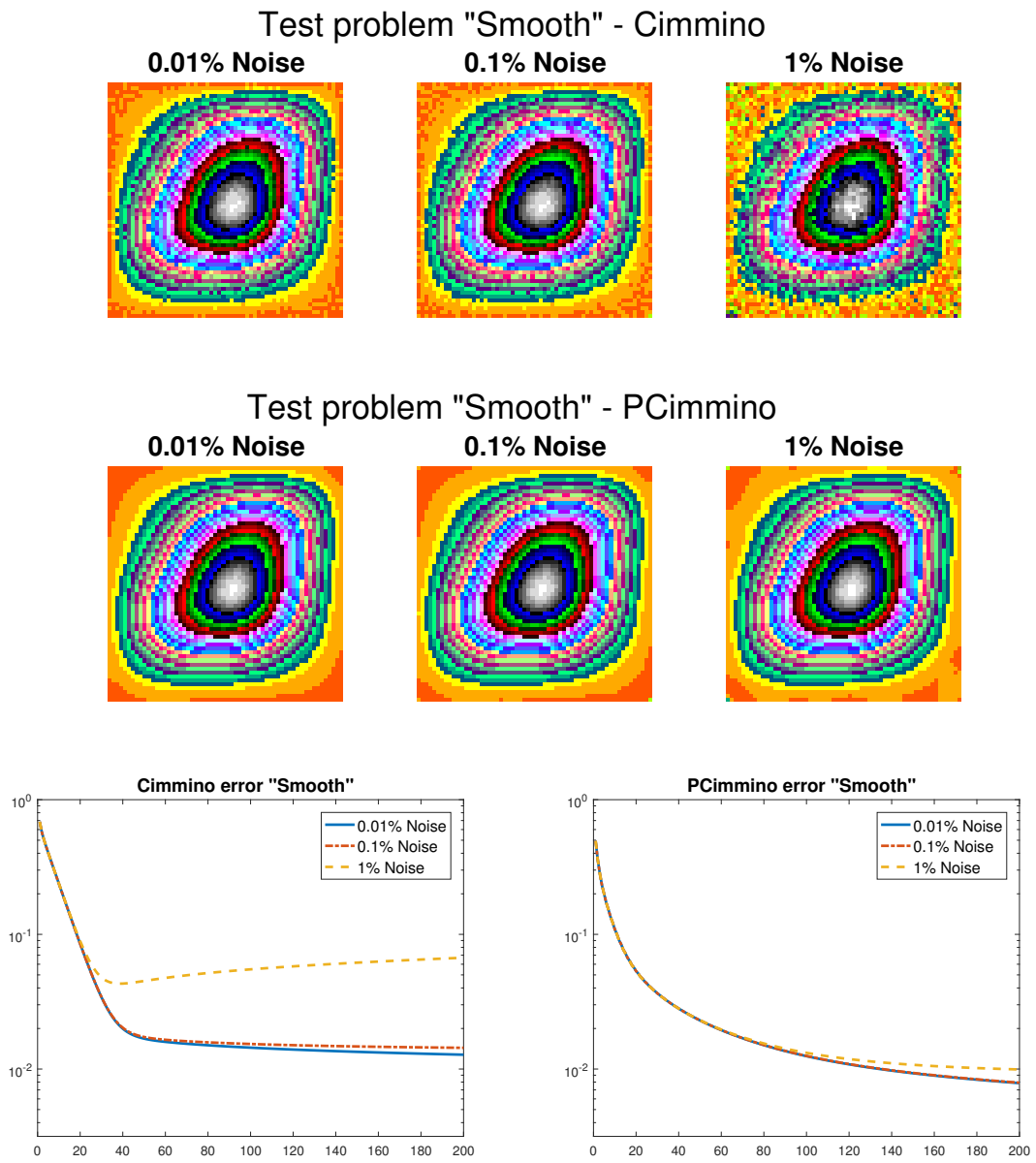

Figure 13: Performance of PCimmino compared with Cimmino on the test problem "Smooth" for different noise levels. In all cases PCimmino produces the best result, but especially for high noise levels there is a significant improvement in the reconstruction using PCimmino. For the error plots the x-axis shows the number of iterations.

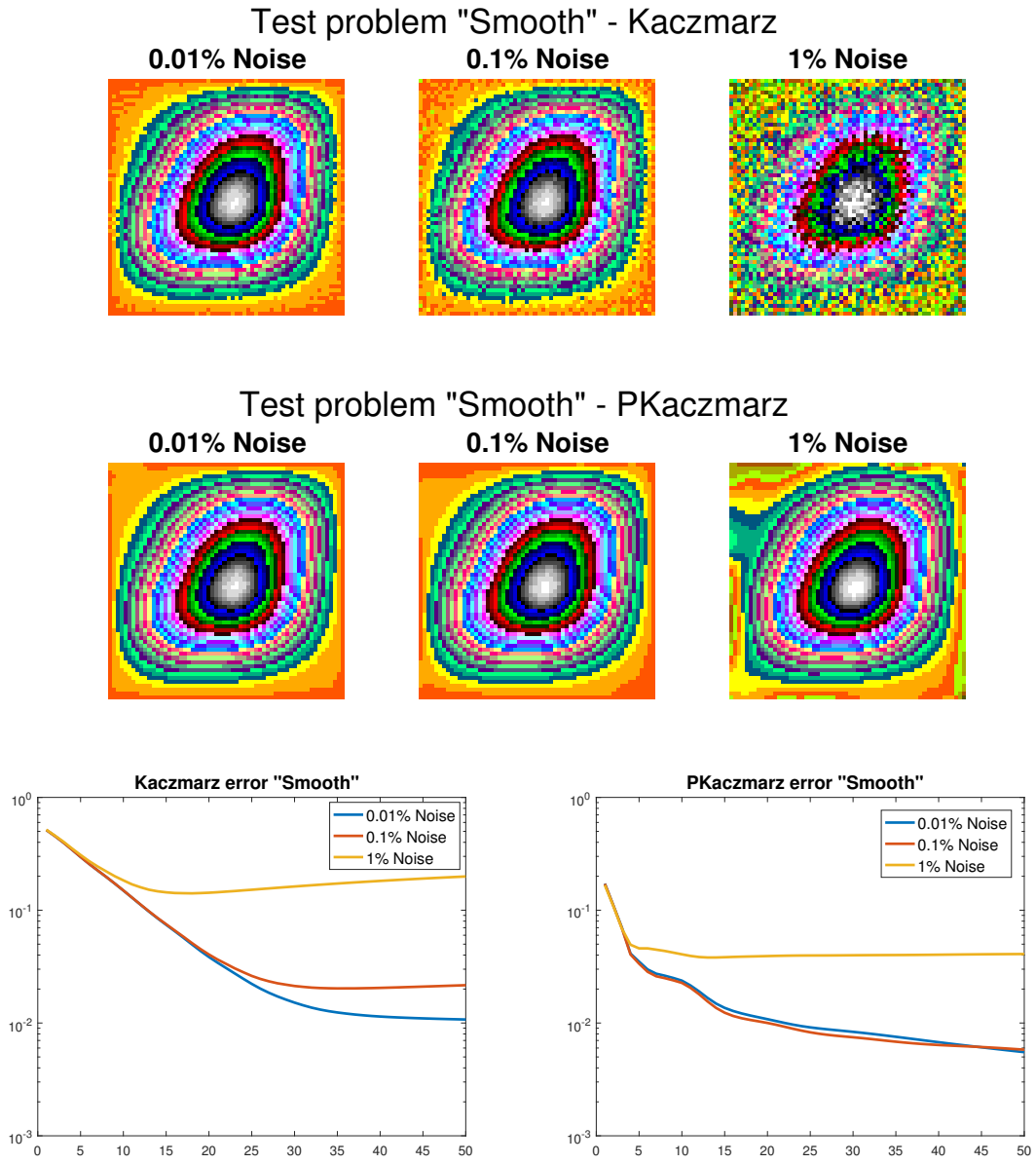

Figure 14: Performance of PKaczmarz compared with Kaczmarz on the test problem "Smooth" for different noise levels. Pkaczmarz gives the best reconstructions, but is very sensitive to higher noise levels. For the error plots the x-axis shows the number of iterations.

7.1.2.2 Phantomgallery: "Threephases"

The phantom "Threephases" from the Phantomgallery is a random image containing the pixel values 0, 0.5 and 1 arranged in domains. For this test problem five domains are used and the seed for the random number generator is set to 2. Because the phantom is piecewise constant I used the gaussian filter `imgaussfilt.m` in Matlab with standard deviation 2 to create a smooth test problem containing smaller objects than the test problem "Smooth". I tested PKaczmarz and PCimmino on this problem for the two different L-matrices and noise level 0.1%. Here the colormap "hsv" seemed appropriate to illustrate the behavior of the methods. The results are shown in figure 15. We see that PKaczmarz gave a really good reconstruction using the first derivative matrix, which was way better than the result obtained by Kaczmarz. The second derivative matrix didn't work well for PKaczmarz and even worse for PCimmino. Also the first derivative matrix worked badly for PCimmino, here Cimmino performed better. As the reconstructions indicate, there is a relation between the order of the derivatives and the intensity of the smoothness in the image: For both PKaczmarz and PCimmino the second derivative matrix gives smoother solutions than the first derivative. But since the methods work differently, the L-matrices have different impacts on the reconstruction. For PKaczmarz the results were less smooth than for PCimmino matching the exact solution better for this test problem. Also none of the reconstructions obtained by the priorconditioned methods looked similar. While the second derivative matrix for PKaczmarz caused red "smoothing stripes" in the background, for PCimmino this derivative matrix induced a butterfly-like formation of the objects and the background. Even though compared to the result for the previous test problem PCimmino worked a lot better for this problem using the second derivative matrix. In this case PCimmino captured the largest objects and the background, but the solution is way to smooth. Note that this problem was not as smooth as the test problem "Smooth", since the objects are smaller and the difference in the pixel values of some neighbour pixels are larger.

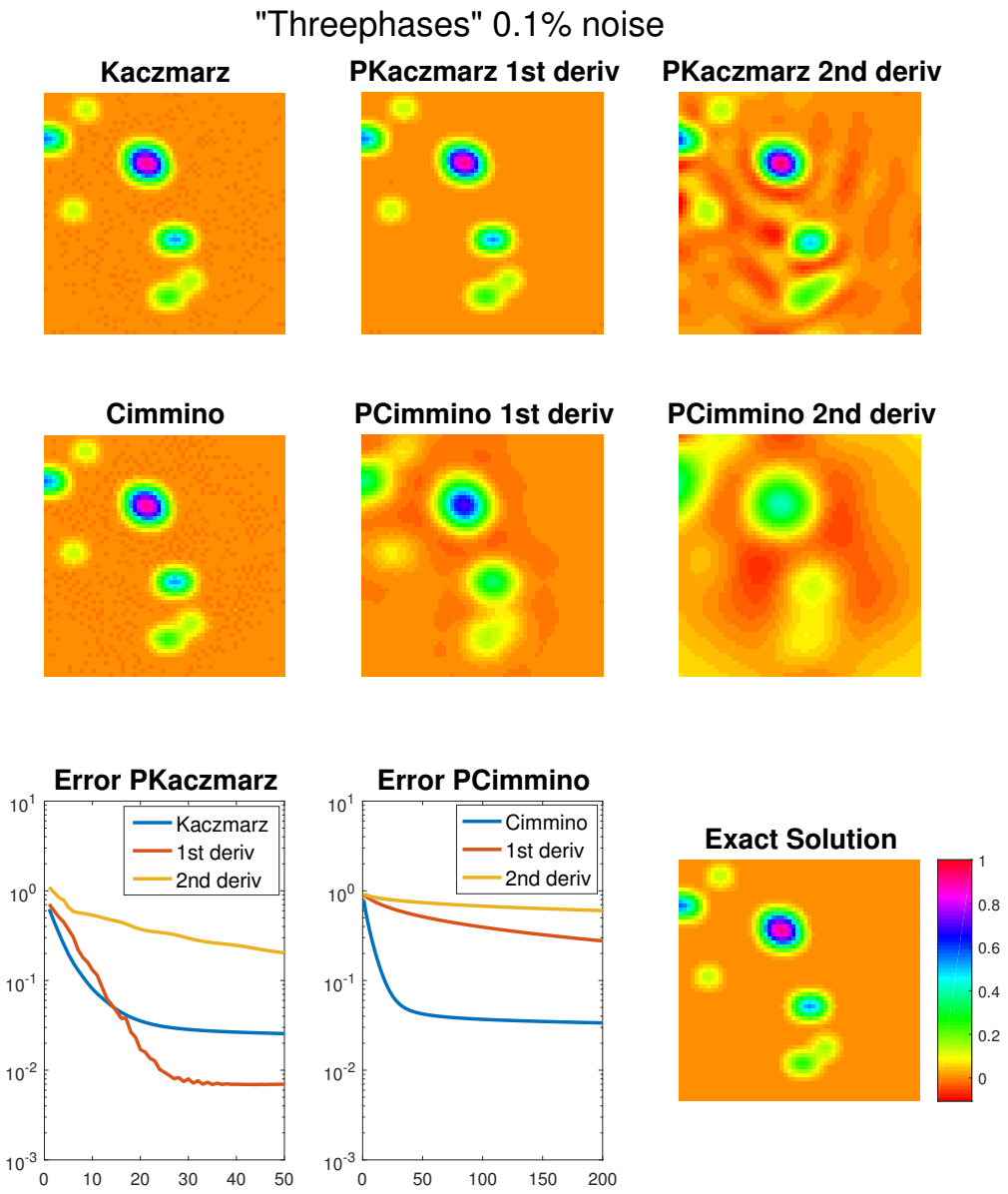

Figure 15: Performance of PKaczmarz and PCimmino on the test problem "Threephases" for the different L-matrices compared to their standard versions. PKaczmarz performed best of all using the first derivative matrix, while the reconstructions produced by PCimmino are way too smooth and worse than the result obtained by Cimmino. For the error plots the x-axis shows the number of iterations.

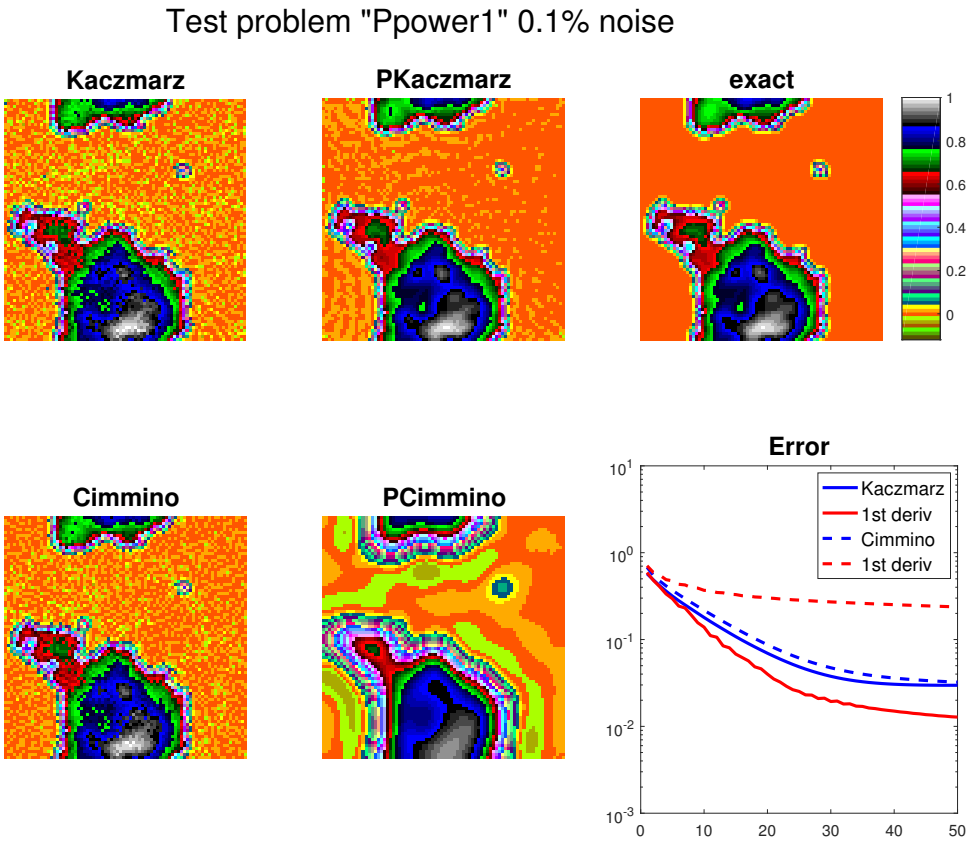

Figure 16: Performance of PKaczmarz and PCimmino on the test problem "Ppower1". PKaczmarz performed best, while PCimmino produced a way too smooth solution. For the error plot the x-axis shows the number of iterations.

7.1.2.3 Phantomgallery: "Ppower"

I created several test problems using the phantom "Ppower" from the phantomgallery that creates a random image with patterns of nonzero pixels. Here I used different smoothness intensities and different seeds for the random number generator. For these test problems I used more discretization intervals: $N = 80$.
The previous test problems indicate that the second derivative only works for very smooth problems, but the following test problems won't be as smooth as "Smooth" or "Threephases" and therefore I will use the first derivative matrix to obtain results from PKaczmarz and PCimmino.

7.1.2.3.1 Ppower1

For the test problem "Ppower1" the phantom "Ppower" is obtained from the `phantomgallery.m` using 0.3 as the ratio for nonzero pixels, 2 as smoothness intensity and 2 as seed for the random number generator. Afterwards I perform smoothing on the phantom using `imgaussfilt.m` with standard deviation 1 to obtain my test problem. Here the shape of the objects in this solution is non-uniform compared to the previous test problems. Using 0.1% noise I get the reconstructions shown in figure 16. PKaczmarz performed best followed by Kaczmarz, Cimmino and PCimmino. The

reconstruction obtained by PKaczmarz was good, but contained more noise than the result for the test problem "Threephases". A reason for this may be the non-uniform shape of the objects and a slightly lower smoothness intensity compared to "Threephases". In constrast, PCimmino created a solution that was way too smooth.

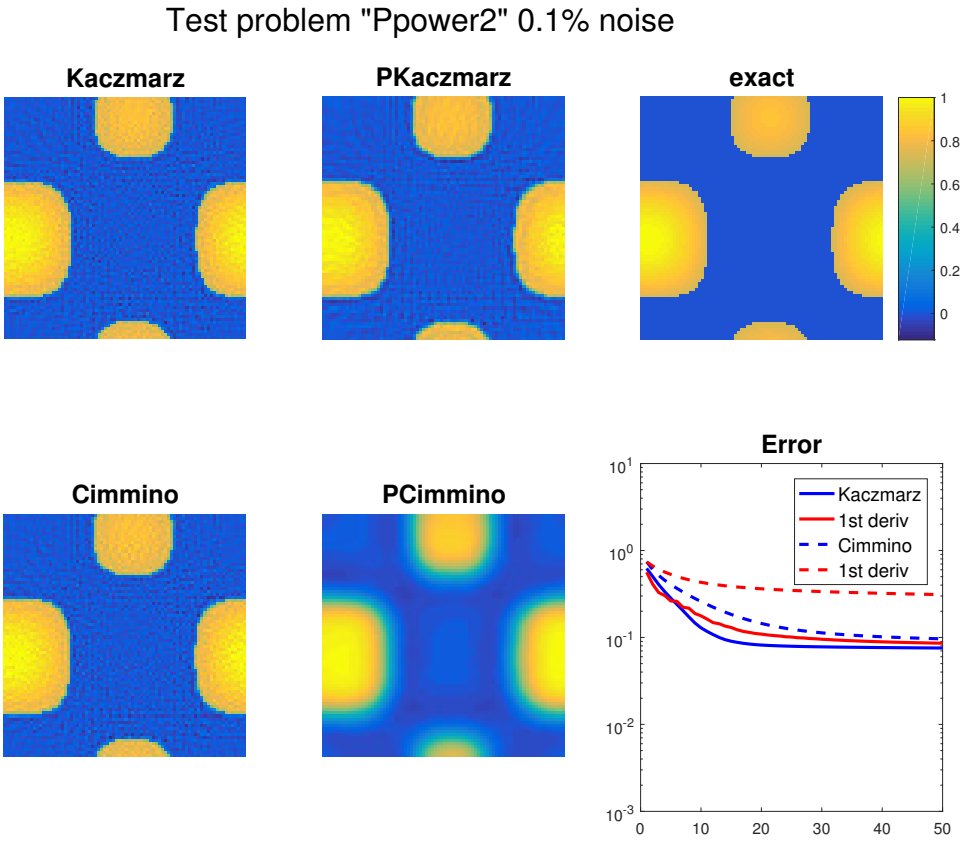

Figure 17: Performance of PKaczmarz and PCimmino on the test problem "Ppower2". Kaczmarz performed best, but PKaczmarz produced a reasonable solution too. For the error plot the x-axis shows the number of iterations.

7.1.2.3.2 Ppower2

As smooth problems in general may contain some parts that are piecewise constant the test problem "Ppower2" should illustrate how the methods perform on piecewise constant problems. The phantom is obtained using 0.3 as the ratio for nonzero pixels, 20 as smoothness intensity and 15 as seed for the random number generator. Here The result is illustrated in figure 17. We see that Kaczmarz gave the best reconstruction, while PKaczmarz didn't perform too bad either. PKaczmarz seemed to smooth the errors in Kaczmarz's solution, but couldn't make them disappear. This is caused by the piecewise constant parts in the solution, PKaczmarz may not be able to recognize the difference between the noisy and the piecewise constant parts and reconstructs all of it. On the other hand PCimmino again gave a result that was way too smooth.

7.1.2.3.3 Ppower3

Finally the test problem "Ppower3" was obtained using 0.3 as the ratio for nonzero pixels, 20 as smoothness intensity and 16 as seed for the random number generatior and then separating this phantom into two parts. The left half contains very smooth objects, where the smoothness is obtained by `imgaussfilt.m` in Matlab using standard deviation 2 and the right half contains two objects, whose edges are piecewise constant. Using a noise level of 0.1%, the reconstructions are shown in figure 18. The error indicates that Kaczmarz, PKaczmarz and Cimmino obtained almost the same quality in their results, while PCimmino again performed bad due to too much smoothing. The reconstruction obtained by PKaczmarz was slightly better than the others and quite interesting: The right half was filled with the same amount of noise as Kaczmarz, while there appeared less noise in the left half, especially between the two smooth objects. So this test problem actually illustrates how much influence piecewise constant parts have on the reconstruction for PKaczmarz compared to the smooth parts. What is common for Kaczmarz, PKaczmarz and Cimmino is a noisy stripe that appeared in the upper right corner of the reconstructions. This may be associated with the small piecewise constant object in the same corner. As this object is very small it may cause even more noise in it's neighbourhood.

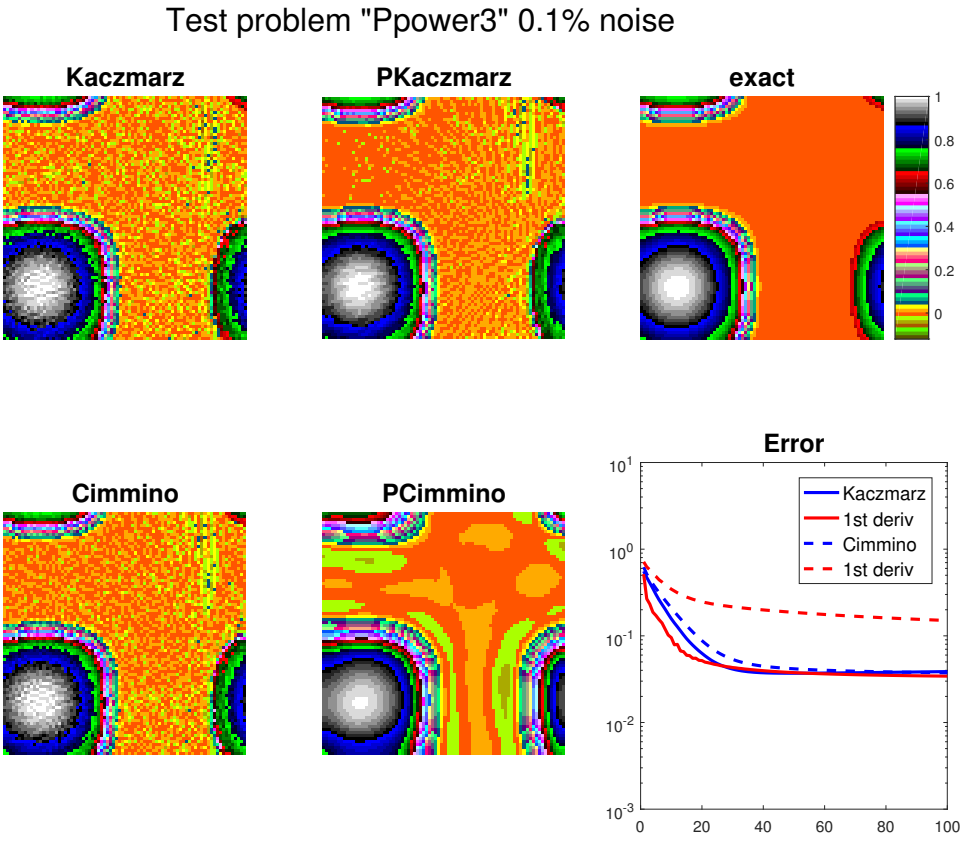

Figure 18: Performance of PKaczmarz and PCimmino on the test problem "Ppower3". PKaczmarz yielded the best reconstruction, while PCimmino had problems with the piecewise constant parts in the solution. For the error plot the x-axis shows the number of iterations.

7.2 Test problems from the real world

To get an impression of how the methods perform on real world problems we got in contact with Mirko Salewski from the Department of Physics at DTU. He is doing research in the field "Plasma Physics and Fusion Energy" and handed us some tomography problems from his field.

7.2.1 "Analytic test case": Bi-Maxwellian distribution

For the test problem "Analytic test case" the coefficient matrix A and the solution x were found analytically and the solution is a bi-Maxwellian distribution [10]. Here A is of size 282×3600 and the solution is an image consisting of 60×60 pixels. The datavector b is synthetic and calculated by the matrix A and the solution x; here we added 0.1% noise and used level 1 for the background Bremstrahlung level. The solution to this test problem is even smoother than the problem "Smooth" used in the previous section, since the contour lines for different intensities in the image are shaped like ellipses. In contrast, for the problem "Smooth" the contour lines look like trapezoids with smooth corners. The reconstructions obtained by PKaczmarz and PCimmino are shown in figure 19. PKaczmarz performed best for the second derivative closely followed by PCimmino for the same derivative matrix. For both Kaczmarz and Cimmino the reconstructions were slightly deformed by the noise. This trend reappeared more smooth in the results for PKaczmarz and PCimmino using the first derivative. For the second derivative both methods could achieve smooth reconstructions that didn't have these deformations, were more smooth and looking way more similar to the exact solution. Even though there is a great difference in how the two methods achieve the good reconstructions. For PKaczmarz the deformations appeared as wavy structures of the contour lines in the result for the first derivative. These wavy structures were smoothed sectionwise in the solution for the second derivative; otherwise the reconstructions for the first and second derivative looked quite similar. For PCimmino using the first derivative the outer contour lines in the reconstruction were not shaped like ellipses and therefore the reconstruction was not good. This trend disappeared completely for the second derivative and a really good solution was obtained. Since there was almost no relation between the reconstructions for the first and second derivative, PCimmino especially for higher derivatives seem to assume that the solution is absolutely smooth.
As this problem is extremely smooth, the second derivative matrix worked best as a priorconditioner for both PKaczmarz and PCimmino.

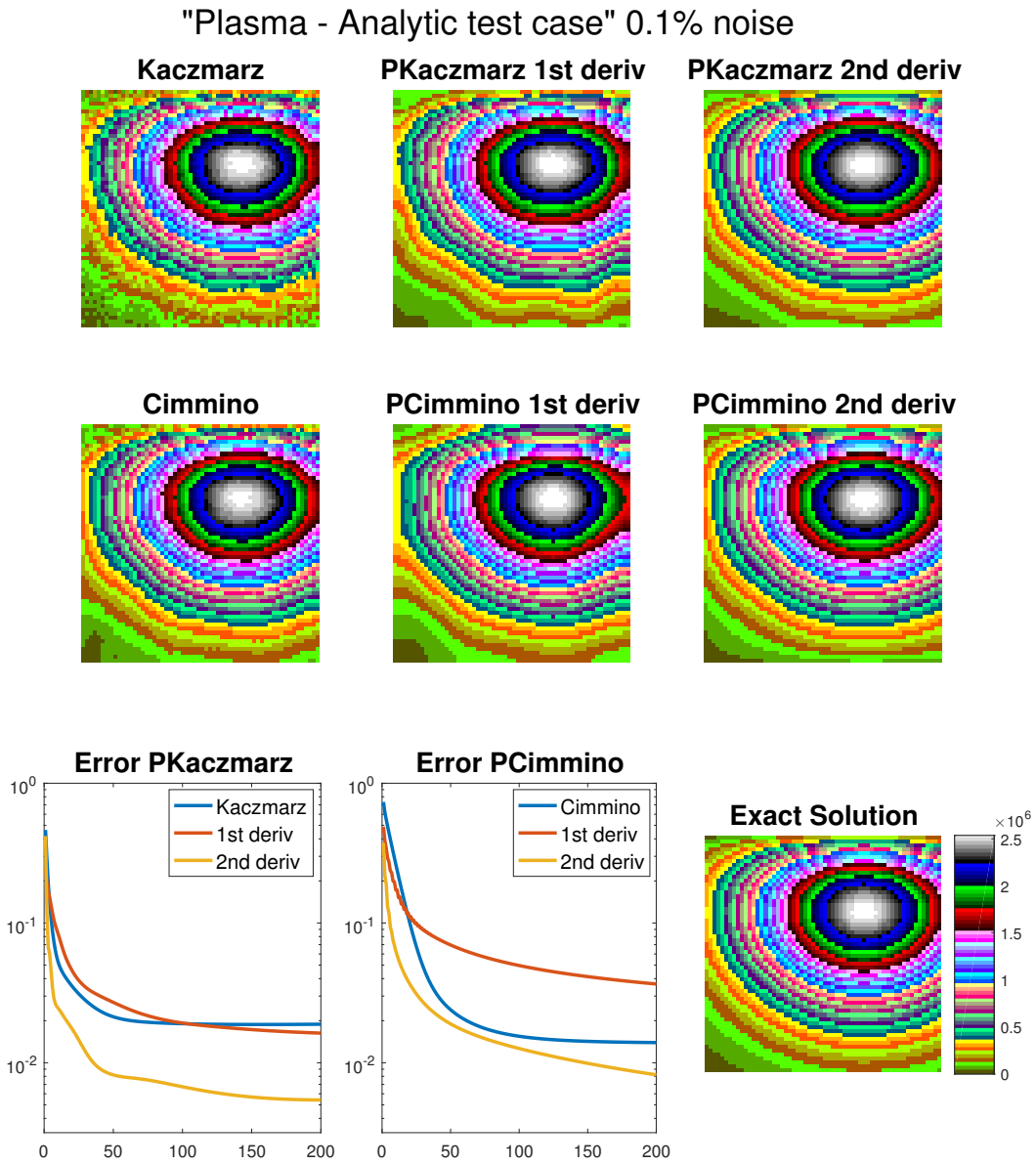

Figure 19: Performance of PKaczmarz and PCimmino on the test problem "Plasma - Analytic test case". Both PKaczmarz and PCimmino obtain really good results for the second derivative matrix. For the error plots the x-axis shows the number of iterations.

7.2.2 "Trimmed": Fast-ion distribution in a fusion plasma

The test problem "Trimmed" was obtained by real data. Here the solution is the fast-ion distribution function in a fusion plasma. This is a velocity-space tomography problem, where the tomographies are calculated using measurements obtained by the fast-ion D_α (FIDA) spectroscopy set-up [9]. The measurements were obtained from 5 different points of views and this information about the geometry was stored in the coefficient matrix A. Here A is of size 863×900. The distribution function x is an image consisting of 30×30 pixels and the datavector is of size 863×1. As the fast-ion distribution is difficult to measure, the error in the data is at least 10%. The performance of PKaczmarz and PCimmino for the test problem "Trimmed" using real data is illustrated in figure 21 in the Appendix. The results are so bad, it is difficult to recognize any similarities between the reconstructions and the exact solution. High error in the datavector and bad resolution in the image may have caused these bad results for the methods. Therefore I tried synthetic data instead, where I used A and the expected fast-ion distribution function x to compute b in the same way as for the problem "Analytic test case". Using 0.1% noise the result for this test problem is shown in figure 20. PKaczmarz and PCimmino performed okay but not as good as for the "Analytic test case". All reconstructions were way to smooth, but PKaczmarz gave the best result for the second derivative. Especially PCimmino performed bad, since both reconstructions didn't show the two small objects seen in the solution. In this case the noise level cannot be a cause for the bad reconstructions, but factors like low resolution and small objects in the solution may influence the quality of the reconstructions.
Further investigations for higher noise levels showed that we couldn't get good reconstructions, but even for noise levels of about 25% the results didn't get as bad as for the real data. So the amount of noise and the low resolution not necessarily explain the difference between the results for real and synthetic data. Note that the better results for the synthetic data may be caused by the fact that the same model is used to generate the test data and compute the reconstruction. Otherwise the set-up for the measurements may also have influence on the quality of the reconstruction: Too few views to obtain measurements yielding areas that are not captured. I don't know much about the physical aspects or the set-up of FIDA used to obtain the measurements, however I want to point out that the set-up may be an important factor for this methods to obtain a good solution, since the information about the set-up is stored in the matrix A.

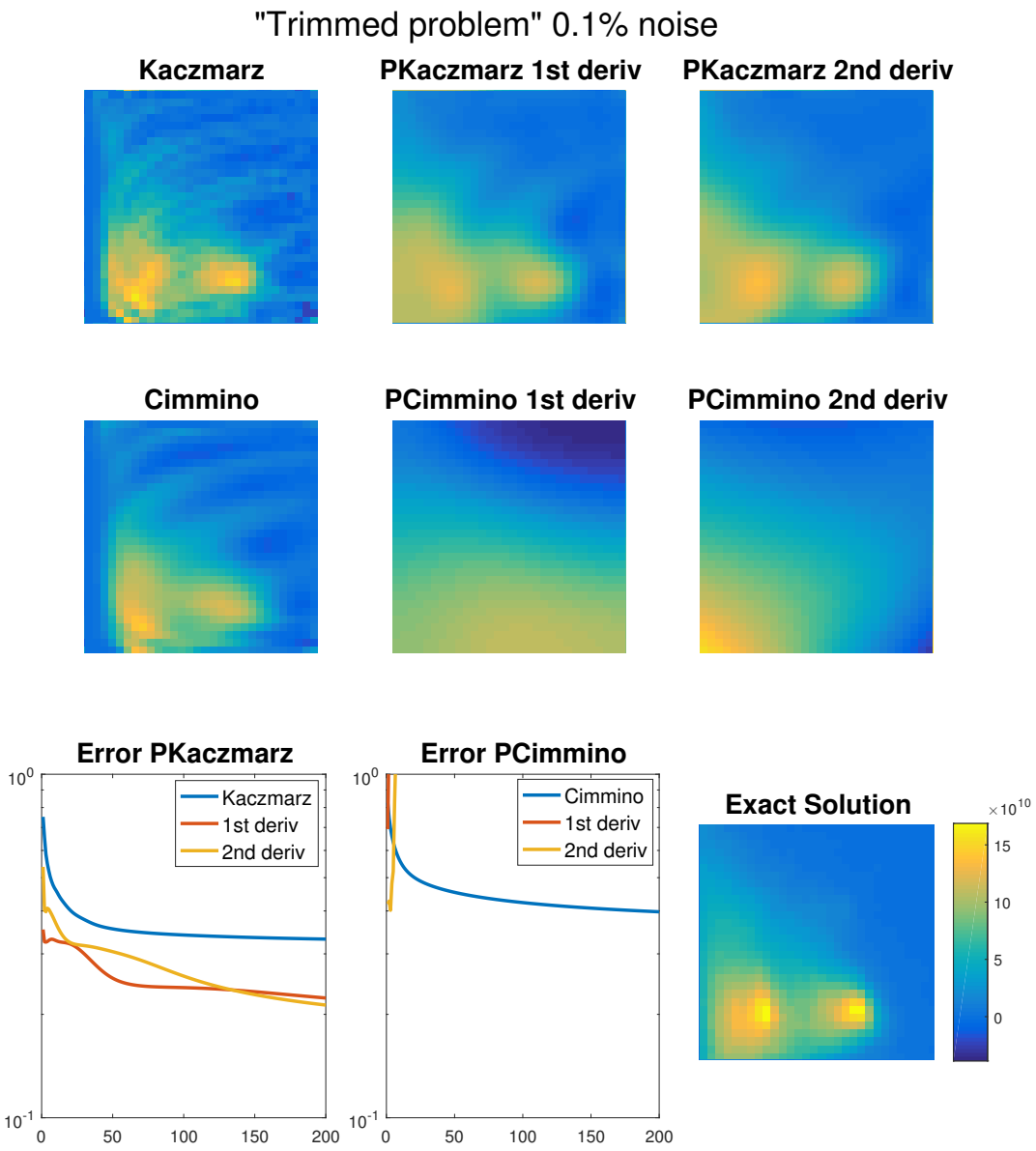

Figure 20: Performance of PKaczmarz and PCimmino on the test problem "Trimmed". PKaczmarz performs best for the second derivative, but all reconstructions are way too smooth. For the error plot the x-axis shows the number of iterations.

7.3 Summary of results and Analysis

Overall, the test problems showed that both PKaczmarz and PCimmino were able to give good reconstructions for specific problems. Here factors like smoothness and composition of the test problem, choice of the L-matrix, amount of noise in the data and structure of the coefficient matrix seemed to be important to the quality of the reconstruction.

The test problems "Threephases" and the three "Ppower" problems represented test cases, where the solution was not perfectly smooth in terms of size, shape and composition of the objects in the solution. These problems illustrated how sensitive PCimmino is to less smooth solutions: Small, non-uniform and piecewise constant objects caused significantly worse performance of Cimmino. Here PKaczmarz is not that sensitive: The method produced good results for small objects, slightly worse results for smooth objects with non-uniform shape and worst results for solutions containing piecewise constant parts. Even though the performance on problems of the latter kind was still almost as good as for Kaczmarz. Nevertheless we can conclude: The smoothness and composition of the solution is a factor that has influence on the quality of the reconstruction.

As the test problems "Gravity", "Threephases" and "Analytic test case" illustrates, there is a relation between the order of derivative for the L-matrix and the smoothness of the reconstruction: In both cases the second derivative matrix caused a smoother solution than the first derivative matrix. Here the solution to the "Analytic test case" and "Gravity" is exceptionally smooth, so the second derivative matrix is suited for this kind of problem. However for "Threephases" the objects are small and not that smooth; therefore the first derivative matrix was a better choice in this case.

The test problems "Smooth" and "Trimmed" using real data illustrated how sensitive PKaczmarz is to noise. For a high amount of noise the reconstructions for Kaczmarz on the test problem "Smooth" were that noisy that PKaczmarz gave a distorted reconstruction. At a noise level of about 10% PKaczmarz was not able to find a solution to the test problem "Trimmed". PCimmino could not find a reasonable solution either, but this may be caused by the fact that the objects are small and the resolution is low.

The one-dimensional problems "Deriv2" and "Phillips" illustrated that the structure of A is crucial for the performance of the methods: For "Deriv2" the basis vectors of the solution didn't match with the exact solution, so Cimmino couldn't produce a good result. Here the priorconditioner matrix could adjust for this factor, so PCimmino could improve Cimmino's reconstruction. In comparison for the partly piecewise constant problem "Phillips" the basis vectors did match well with the problem, so Cimmino gave a good result. As the L-matrices give smoother solutions, PCimmino performed worse than Cimmino.

Especially the test problems "Threephases" and "Analytic test case" showed how different the priorconditioned methods behave for the same priorconditioners. For PKaczmarz there was a close relation between the solution for the different priorconditioners and the solution of Kaczmarz. Here PKaczmarz performed a sectionwise smoothing on the solution obtained by Kaczmarz. For PCimmino the difference between the solution of PCimmino for the two L-matrices and the solution of Cimmino was more striking. Here it seemed like PCimmino assumed the solution is very smooth and obtained a solution that not necessarily was related to the solution produced by Cimmino. For really smooth problems this behavior may be an advantage, but for less smooth problems like "Threephases" this leads to bad reconstructions.

Note that for most of the test problems low noise levels were used and that most of the data were obtained involving inverse crime. Thus, the results may not be representative. Nevertheless the test problems were enough to determine whether or not it would be worth it continue research on the priorconditioned algebraic iterative methods. And the investigations gave first impressions of some of PKaczmarz's and PCimmino's strengths and weaknesses.

It is striking that PKaczmarz is able to produce good results for all different kind of smooth test problems using a low amount of noise, since the priorconditioner matrices was derived for PCimmino only. Therefore it may be reasonable to do further research in this field and derive the priorconditioner matrices suitable for PKaczmarz. The solution for PKaczmarz and PCimmino may also be improved by making the methods work with non-negativity or box constraints when prior information about the function values in the solution is available.

The tomography problem "Trimmed" from the Department of Physics at DTU showed the relevance of these methods for research reasons. Even though the approach for this test problem was rather optimistic, since a high noise level and low resolution made it very difficult to obtain good reconstructions. Nevertheless PKaczmarz produced reasonable results for the synthetic data on the same problem. This is a step in the right direction. A next approach may be to test PKaczmarz and PCimmino on more simple problems obtained from real data to get a more realistic impression how the methods perform when inverse crime is avoided.

PCimmino fulfilled our expectations, as this method created smooth solutions and in case of very smooth test problems the performance of Cimmino was improved. A disadvantage is PCimmino's performance on less smooth problems, here the method is rather ineffective. On the other hand we had no great expectations for PKaczmarz since the priorconditioners were derived for PCimmino. Here our expectations were exceeded, since PKaczmarz for low noise levels could handle different kinds of smooth problems.

8 Conclusion

Using knowledge from priorconditioned CGLS we successfully derived and implemented priorconditioned versions of Cimmino and Kaczmarz. Based on generalized Tikhonov regularization we were able to establish priorconditioner matrices for PCimmino appearing as discrete approximations of the first and second derivative operator. We used the same approach for PKaczmarz, but didn't know if these priorconditioner matrices were a good choice for PKaczmarz.

In this project efficient implementation of PKaczmarz and PCimmino was of secondary importance: The expression involving computations with the pseudoinverse and the transpose of the pseudoinverse of the priorconditioner matrix was defined explicitly due to this expression appeared in every update of the iteration vector. As we tested the methods on smaller test problems this saved computation time. A general approach would be to calculate the mentioned expression in every iteration to save memory space and make it possible to use the methods on large problems. Otherwise a combination between the two approaches matching with the size of the given problem may be a possibility for future implementations. Also the programming language Matlab isn't suitable for iterative methods, especially row-action methods, since they involve many loops and Matlab is optimized for matrix and vector operations. Here another programming language supporting loops could improve the efficiency of the implementation.

We expected that PKaczmarz and PCimmino might improve the performance of Kaczmarz and Cimmino on tomography problems where the solution was smooth. Here our expectations for both methods were only met for test problems where the solution was very smooth. PCimmino produced way too smooth reconstructions for problems where the solution contained small, non-uniform or piecewise constant objects. For these problems PKaczmarz performed way better. Here our expectations were exceeded since the priorconditioner matrices were derived for PCimmino, but PKaczmarz could obtain better solutions using this priorconditioners. Therefore it may be reasonable to do further research in this field and derive the priorconditioner matrices suitable for PKaczmarz.

The analysis for different test problems showed that the performance of the priorconditioned methods depends on the specific test problem: Factors like smoothness and composition of the test problem, amount of noise in the data and structure of the coefficient matrix were most important to the quality of the reconstruction. Especially PCimmino performed bad when the solution was less smooth. However PKaczmarz was not that sensitive to the smoothness of the solution; here problems occurs when piecewise constants parts appear in the solution. On the other hand PCimmino is way more robust to noise in the data compared to PKaczmarz. For the specific test problem the choice of the priorconditioner matrix is important: If the solution is very smooth, especially for PKaczmarz a higher order derivative priorconditioner matrix is needed, while PCimmino already creates a very smooth solution using the first derivative matrix.

Test problems from real data from the Department of Physics at DTU showed the relevance of these methods for research reasons. Even though PKaczmarz and PCimmino were not able to produce reasonable results for the real data due to a high amount

of noise and low resolution. Nevertheless PKaczmarz produced reasonable results for synthetic data on the same problem. As all test problems in this project work were obtained by synthetic data, a next approach may be to test PKaczmarz and PCimmino on more simple problems obtained from real data to get a more realistic impression how the methods perform. Also the methods may be improved by introducing non-negativity or box constraints when prior information about the function values in the solution is available.

References

[1] *From the course "01005 Mathematics 1" - eNote 8: Linear Mappings.* URL: `http://01005.mat.dtu.dk/materialer/enoter/afsnit/NUID21-tn8/`, Autumn 2016.

[2] Joost Batenburg, Per Christian Hansen, and William Lionheart. *Scientific Computing for Computed Tomography.* URL: `http://www2.compute.dtu.dk/~pcha/HDtomo/SCforCT.html`, Manuscript in preparation.

[3] Daniela Calvetti and Erkki Somersalo. *Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing.* Springer, United States of America, 2007.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, 1996.

[5] P. C. Hansen and M. Saxild-Hansen. *AIR Tools - A MATLAB Package of Algebraic Iterative Reconstruction Methods*, Journal of Computational and Applied Mathematics, 236 (2012), pp.2167-2178. URL: `http://www.sciencedirect.com/science/article/pii/S0377042711005188`.

[6] Per Christian Hansen. *Discrete Inverse Problems: Insight and Algorithms.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2010.

[7] Alan J. Laub. *Matrix Analysis for Scientists and Engineers.* SIAM, Philadelphia, 2005.

[8] Randall J. LeVeque. *Finite difference methods for ordinary and partial differential equations : steady-state and time-dependent problems.* SIAM, Philadelphia, 2007.

[9] M. Salewski, B. Geiger, A. S. Jacobsen, P.C. Hansen, W. W. Heidbrink, S. B. Korsholm, F. Leipold, J. Madsen, C. Moseev, S. K. Nielsen, M. Nocente, T. Odstrcil, J. Rasmussen, L. Stagner, M. Stejner, M. Weiland, and the ASDEX Upgrade Team. *High-definition velocity-space tomography of fast-ion dynamics*, Nuclear Fusion, 56 (2016), 106024 (15pp). URL: `doi:10.1088/0029-5515/56/10/106024`.

[10] M. Salewski, B. Geiger, D. Moseev, W. W. Heidbrink, A. S. Jacobsen, S. B. Korsholm, F. Leipold, J. Madsen, S. K. Nielsen, J. Rasmussen, M. Stejner, M. Weiland, and the ASDEX Upgrade Team. *On velocity-space sensitivity of fast-ion D-alpha spectroscopy*, Plasma Physics and Controlled Fusion, 56 (2014), 105005 (15pp). URL: `doi:10.1088/0741-3335/56/10/105005`.

[11] Jonathan Richard Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain.* 1994.

Appendix

Investigation of the Kronecker product $L_2 \otimes I$

I investigate the vector $\text{vec}(IXL_2^\mathsf{T})$:

$$\text{vec}(IXL_2^\mathsf{T}) = \text{vec}\left(\begin{bmatrix} x_1 & x_{N+1} & \dots & x_{(N-1)N+1} \\ x_2 & x_{N+2} & \dots & x_{(N-1)N+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_N & x_{2N} & \dots & x_{N^2} \end{bmatrix} \begin{bmatrix} 1 & & & & \\ -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \\ & & & & 1 \end{bmatrix}\right)$$

$$= \begin{bmatrix} x_1 - 2x_{N+1} + x_{2N+1} \\ x_2 - 2x_{N+2} + x_{2N+2} \\ \vdots \\ x_N - 2x_{2N} + x_{3N} \\ x_{N+1} - 2x_{2N+1} + x_{3N+1} \\ \vdots \\ \vdots \\ x_{(N-2)N} - 2x_{(N-1)N} + x_{N^2} \end{bmatrix}.$$

Separating this vector into N column vectors of identical magnitude, we see that each of these vectors correspond to the second derivative of three elements in each row of X. And investigating the Kronecker product $L_2 \otimes I$ gives me

$$L_2 \otimes I = \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} & \begin{bmatrix} -2 & & & \\ & -2 & & \\ & & \ddots & \\ & & & -2 \end{bmatrix} & \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} & & \\ & \ddots & & \ddots & & \ddots \\ & & \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} & \begin{bmatrix} -2 & & & \\ & -2 & & \\ & & \ddots & \\ & & & -2 \end{bmatrix} & \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \end{bmatrix}.$$

Multiplying this matrix $L_2 \otimes I$ with the vector x gives me the same expression as $\text{vec}(IXL_2^\mathsf{T})$. Therefore we must have that $\text{vec}(IXL_2^\mathsf{T}) = (L_2 \otimes I)x$.

Figures

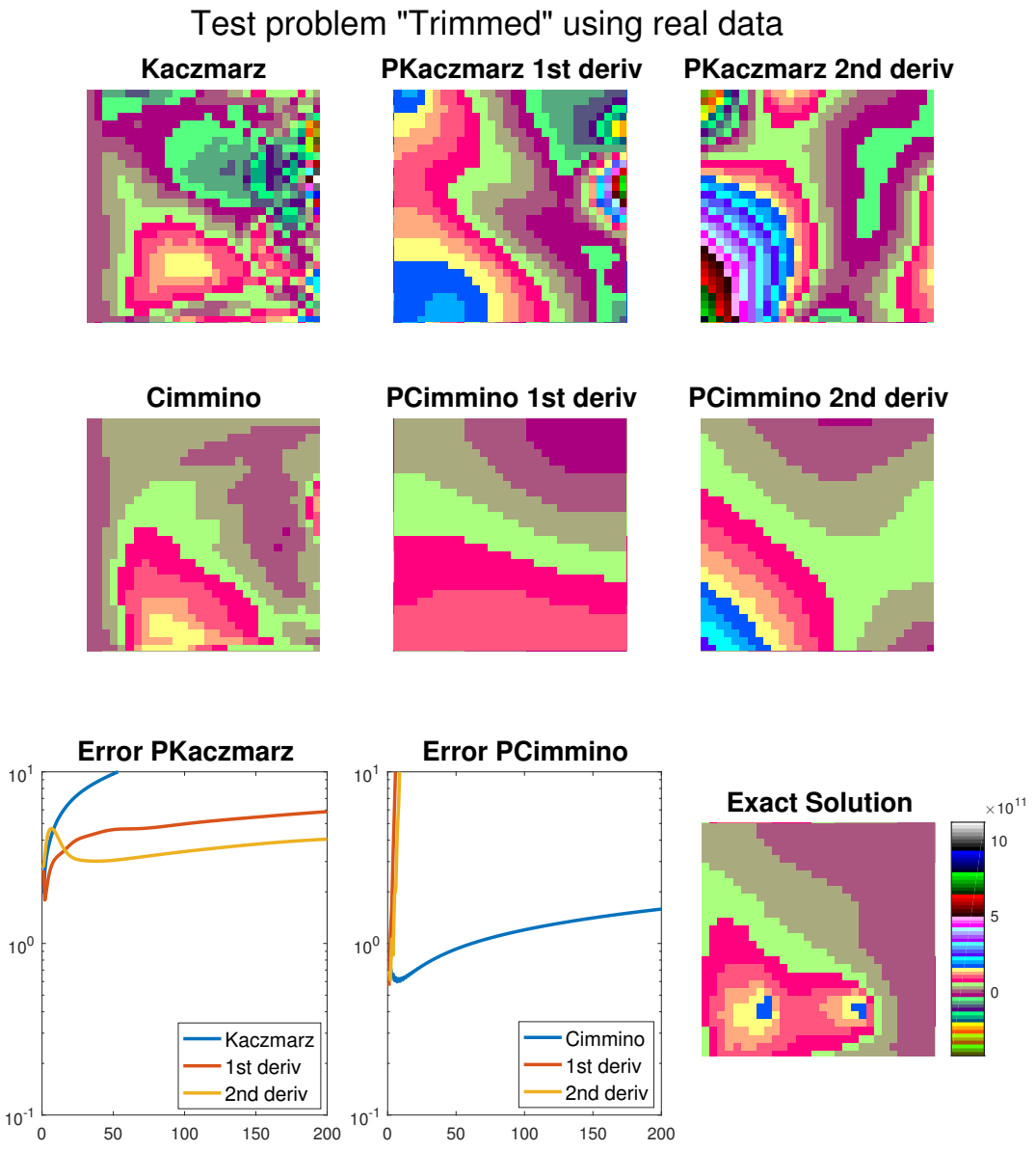

Figure 21: Performance of PKaczmarz and PCimmino on the test problem "Trimmed" using real data. All methods perform really bad and it is difficult to recognize similarities between the reconstructions and the exact solution. For the error plot the x-axis shows the number of iterations.