Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Signature:

Jennifer Zheng March 31, 2022

Regularization of ill-posed inverse problems under mixed precision arithmetics

By

Jennifer Zheng

James Nagy Ph.D.
Advisor

Mathematics

James Nagy Ph.D.
Advisor

Clifford Carrubba, Ph.D.
Committee Member

Lars Ruthotto, Ph.D.
Committee Member

2022

Regularization of ill-posed inverse problems under mixed precision arithmetics

By

Jennifer Zheng

James Nagy Ph.D.
Advisor

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences of
Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Mathematics

2022

Abstract

Regularization of ill-posed inverse problems under mixed precision arithmetics
By Jennifer Zheng

In numerical computations, using low precision floating point arithmetic enables computer programs for scientific applications to have faster loops, less communication, and lower energy consumption. As low precision arithmetic leads to limited accuracy for certain data, modern computer architectures built on graphics processing units can be implemented using mixed precision for scientific computations. The basic idea is to use low precision arithmetic to accelerate speed on certain calculations, mixed with a limited amount of high precision calculations, while maintaining sufficiently appropriate accuracy of the final result.

Recent work studies the use of mixed precision arithmetic for algorithms to solve certain basic, well-conditioned linear systems. In this thesis, we will extend these ideas to the more complicated class of inverse problems, where it is necessary to employ a technique known as regularization to compute an approximate solution of a severely ill-conditioned problem.

We will explore different regularization methods, such as truncated singular value decomposition, Tikhonov regularization, and methods of choosing their respective regularization parameters, under mixed precision arithmetic. We will also analyze the performance of iterative methods with different preconditioners to regularize under mixed precision arithmetic.

Regularization of ill-posed inverse problems under mixed precision arithmetics

By

Jennifer Zheng

James Nagy Ph.D.
Advisor

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Mathematics

2022

Acknowledgments

I would like to thank my advisor, Dr. James Nagy, for his support along the way. This thesis would not have been possible without his guidance. Throughout the years of working with Dr. Nagy, he has taught me the necessary skills of becoming an independent researcher. He led me into the world of computational mathematics with his experience, patience, enthusiasm, and knowledge.

I would also like to thank my honors committee and the Department of Mathematics at Emory. Both Dr. Clifford Carrubba and Dr. Lars Ruthotto provided me with lots of help and guidance throughout my undergraduate life at Emory and brought me to where I am today.

Lastly, I want to thank my parents, my friends, and all my other peers at Emory. They have supported me at each step of my journey.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In numerical computations, using low precision floating point arithmetic enables computer programs for scientific applications to have faster loops, less communication, and lower energy consumption [12]. As low precision arithmetic leads to limited accuracy for certain data, modern computer architectures built on graphics processing units can be implemented using mixed precision for scientific computations. The basic idea is to use low precision arithmetic to accelerate speed on certain calculations, mixed with a limited amount of high precision calculations, while maintaining sufficiently appropriate accuracy of the final result.

Recent work by Higham and colleagues [3] studies the use of mixed precision arithmetic for algorithms to solve certain basic, well-conditioned linear systems. In this thesis, we extend these ideas to the more complicated class of inverse problems, where it is necessary to employ a technique known as regularization to compute an approximate solution of a severely ill-conditioned problem.

We consider the problem

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{b} \tag{1.1}$$

developed from

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.2}$$

in the context of image or signal reconstruction. $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a matrix representation of the point spread function (PSF) that blurs the image, $\boldsymbol{x} \in \mathbb{R}^n$ is the vector representation of the true image, $\boldsymbol{e} \in \mathbb{R}^n$ is the vector representation of the unknown noise added to the true image, and $\boldsymbol{b} \in \mathbb{R}^n$ is a vector representation of the blurred image that we observe. Our goal is to solve for $\boldsymbol{x}$ given $\boldsymbol{A}$ and $\boldsymbol{b}$ with the consideration that $\boldsymbol{A}$ is ill-conditioned.

There are many different regularization methods, such as truncated singular value decomposition (TSVD) and Tikhonov regularization [10]. Each regularization method requires choosing a problem and data dependent regularization parameter, which adjusts the quality of the solution. There are well known methods, such as generalized cross validation (GCV), to help estimate good parameters.

Inverse problems arise in many important fields, including medical imaging, astronomy, geophysics, microscopy, and more. While work has been done to develop algorithms and software implementations for techniques such as TSVD, Tikhonov, and GCV, little to no work has been done on how these methods perform when using mixed precision arithmetic.

Other than the previous regularization methods mentioned, we also work on analyzing the performance of iterative methods under mixed precision arithmetic.

1.1 Notation

In this thesis, all matrices are notated with bold upper-case letters such as $\boldsymbol{A}$, all vectors are notated with bold lower-case letters such as $\boldsymbol{x}$, and all scalars are notated with nonbold lower-case letters such as c. The lower-case u generally indicates half of the machine epsilon of the different precisions. The norm $\|\cdot\|$ in this thesis refers to the 2-norm $\|\cdot\|_2$.

Chapter 2

Background

In this chapter, we introduce the mathematical and computational background necessary for the later sections. Some algorithms previously developed that serve as the foundation to section 3 are also introduced.

2.1 Floating Point Arithmetics

Before introducing algorithms under mixed precision arithmetic, we first define the different precisions considered in this thesis.

Our definitions of precision come from IEEE 754 [1], an industry standard for representing floating-point numbers in computers. The most recent revision happened in 2019. We define floating point 16-bit as half precision, floating point 32-bit as single precision, and a 64-bit format as double precision. The respective bits for storing significant figures of the fraction and exponent of a number are in the chart below. In addition, one bit is used to store the sign of the number, and u in the chart indicates the unit round-off to three significant figures.

The different data types result in different rounding errors and can thus be applied at different parts of the algorithms to maintain sufficiently appropriate amount of accuracy.

Type	f bits	Exp bits	Range	$u = 2^{-f}$
bfloat16	8	8	$10^{\pm 38}$	3.91×10^{-3}
fp16	11	5	$10^{\pm 5}$	4.88×10^{-4}
fp32	24	8	$10^{\pm 38}$	5.96×10^{-8}
fp64	53	11	$10^{\pm 308}$	1.11×10^{-16}

Table 2.1: Parameters for four IEEE arithmetic precisions [11].

2.2 Inverse Problems

As mentioned in Chapter 1, we consider a problem of the form

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{b},$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, $\boldsymbol{b} \in \mathbb{R}^n$, $\boldsymbol{x} \in \mathbb{R}^n$, and the unknown noise $\boldsymbol{e} \in \mathbb{R}^n$ is added to the measured data.

When solving a forward problem, we are given $\boldsymbol{A}$ and $\boldsymbol{x}$, and compute an approximation of the vector $\boldsymbol{b}$. And when solving an inverse problem, we are given $\boldsymbol{A}$ and $\boldsymbol{b}$, and compute an approximation of the vector $\boldsymbol{x}$.

A challenge we face when solving the inverse problems is that the inverse problems are often ill-posed, which means that $\boldsymbol{A}$ is an ill-conditioned matrix with a large condition-number $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$. In this case, small changes in $\boldsymbol{b}$ can lead to large changes in our solution $\boldsymbol{x}$.

The inverse problem can be analyzed and approximate solutions computed with the singular value decomposition (SVD) of the matrix $\boldsymbol{A}$ which is given by

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{n} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^\top \tag{2.1}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices with columns $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, and $\boldsymbol{\Sigma}$ is a diagonal matrix with diagonal entries σ_i being the singular values of $\boldsymbol{A}$.

Using the SVD in equation (2.1), the naive inverse solution can be written as

$$\begin{aligned}\hat{\boldsymbol{x}} &= \boldsymbol{A}^{-1}\boldsymbol{b} \\ &= \boldsymbol{A}^{-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}) \\ &= \boldsymbol{x} + \boldsymbol{A}^{-1}\boldsymbol{e} \\ &= \boldsymbol{x} + \sum_{i=1}^{n} \frac{\boldsymbol{u}_i^\top \boldsymbol{e}}{\sigma_i}\boldsymbol{v}_i. \end{aligned} \tag{2.2}$$

The naive inverse solution shows that if the error vector $\boldsymbol{e}$ is zero, then $\hat{\boldsymbol{x}} = \boldsymbol{x}$, the exact desired solution. However, if the error is not equal to zero, then it can be seen from equation (2.2) that division by small singular values will magnify the errors in vector $\boldsymbol{b}$. The smaller the singular values, the larger the magnification. For inverse problems of concern in this project, the singular values decay gradually to zero, such that errors will be highly magnified making the naive inverse solution, $\hat{\boldsymbol{x}}$, a very poor approximation of the true vector $\boldsymbol{x}$. Regularization is thus needed to damp the effects caused by division of small singular values.

2.3 Regularization Methods

In this section, we introduce two regularization methods commonly used to reduce the effects of error magnification caused by division of small singular values.

We express the regularized inverse solution as

$$\boldsymbol{x}_F = \sum_{i=1}^{n} \phi_i \frac{\boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i \tag{2.3}$$

where ϕ_i is the filter factor. The values of ϕ_i depends on the regularization method used.

2.3.1 Truncated SVD

The most traditional regularization method, the Truncated Singular Value Decomposition (TSVD), truncates the small singular values to avoid the magnification of errors. We choose a truncation index k such that all $\sigma_i = 0$ for all $i > k$. The filter factors are thus

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

Thus the regularized system has solution

$$\boldsymbol{x}_F = \sum_{i=1}^{n} \phi_i \frac{\boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i = \sum_{i=1}^{k} \frac{\boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i. \tag{2.4}$$

The solution can also be written as $\boldsymbol{x}_F = \boldsymbol{V}\boldsymbol{\Sigma}_F^\dagger \boldsymbol{U}^\top \boldsymbol{b}$ where

$$\boldsymbol{\Sigma}_F^\dagger = \operatorname{diag}\left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \cdots, \frac{1}{\sigma_k}, 0, \cdots, 0\right).$$

2.3.2 Tikhonov Regularization

A potential problem with the TSVD is that the filter factors have too sharp of a cutoff, thus we seek another regularization method to obtain a smoother transition of the filter factors. The Tikhonov regularization has filter factors

$$\phi_i = \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2}$$

where α is a small value that regularizes the singular values, called the regularization parameter. The Tikhonov solution can be written in the form

$$\boldsymbol{x} = \sum_{i=1}^{n} \phi_i \frac{\boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i = \sum_{i=1}^{n} \frac{\sigma_i \boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i^2 + \alpha^2} \boldsymbol{v}_i \tag{2.5}$$

If the σ_i is large, then $\phi_i \approx 1$; for tiny σ_i, $\phi_i \approx 0$. This approach effectively works as a filter. Components of the solution corresponding to large singular values pass through the filter, but components corresponding to small singular values are blocked by the filter. In this case, $\boldsymbol{x}_F = \boldsymbol{V}\boldsymbol{\Sigma}_F^{\dagger}\boldsymbol{U}^{\top}\boldsymbol{b}$ where

$$\boldsymbol{\Sigma}_F^{\dagger} = \operatorname{diag}\left(\frac{\sigma_1}{\sigma_1^2+\alpha^2}, \frac{\sigma_2}{\sigma_2^2+\alpha^2}, \cdots, \frac{\sigma_n}{\sigma_n^2+\alpha^2}\right).$$

Equivalently, we can solve the least squares problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}||^2 + \alpha^2||\boldsymbol{x}||^2.$$

We can see this equivalence by noticing that

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}||^2 + \alpha^2||\boldsymbol{x}||^2 = \min_{\boldsymbol{x}\in\mathbb{R}^n} \left\| \begin{bmatrix} \boldsymbol{A} \\ \alpha\boldsymbol{I} \end{bmatrix} \boldsymbol{x} - \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} \right\|^2. \tag{2.6}$$

To solve a least squares problem $\boldsymbol{Ax} = \boldsymbol{b}$, we solve for its normal equation $\boldsymbol{A}^\top \boldsymbol{Ax} = \boldsymbol{Ab}$. Substituting $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$ in the normal equation of equation (2.6), we get

$$\begin{bmatrix} \boldsymbol{A}^\top & \alpha \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} \\ \alpha \boldsymbol{I} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{A}^\top & \alpha \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}$$

$$(\boldsymbol{A}^\top \boldsymbol{A} + \alpha^2 \boldsymbol{I})\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b}$$

$$(\boldsymbol{V\Sigma U}^\top \boldsymbol{U\Sigma V}^\top + \alpha^2 \boldsymbol{I})\boldsymbol{x} = \boldsymbol{V\Sigma U}^\top \boldsymbol{b}$$

$$(\boldsymbol{V\Sigma}^2 \boldsymbol{V}^\top + \alpha^2 \boldsymbol{I})\boldsymbol{x} = \boldsymbol{V\Sigma U}^\top \boldsymbol{b}$$

$$\sum_{i=1}^{n} \boldsymbol{v}_i (\sigma^2 + \alpha^2) \boldsymbol{v}_i^\top \boldsymbol{x} = \sum_{i=1}^{n} \boldsymbol{v}_i \sigma_i \boldsymbol{u}_i^\top \boldsymbol{b}$$

$$\boldsymbol{x} = \sum_{i=1}^{n} \frac{\boldsymbol{v}_i^\top \boldsymbol{v}_i \sigma_i \boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i^2 + \alpha^2} \boldsymbol{v}_i$$

$$= \sum_{i=1}^{n} \frac{\sigma_i \boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i^2 + \alpha^2} \boldsymbol{v}_i$$

which is equivalent to equation (2.5).

2.4 Iterative Methods

When solving for a linear system $\boldsymbol{Ax} = \boldsymbol{b}$ as mentioned in equation (1.2), methods that use matrix factorizations such as $\boldsymbol{PA} = \boldsymbol{LU}$ (i.e. the Gaussian elimination), $\boldsymbol{A} = \boldsymbol{QR}$, or $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$ might require too much time or space. These methods that compute the exact answers after a finite number of steps are called *direct methods*. In contrast, *iterative methods* do not compute the exact answers after a finite number of steps but compute successively better approximations at each iteration [4].

2.4.1 Iterative Refinement

Iterative Refinement in multiple precisions was a method proposed by Carson and Higham to accelerate the solution of linear systems [3]. It solves the problem $\boldsymbol{Ax} = \boldsymbol{b}$

with the following approach under precisions $u_r \leq u \leq u_s \leq u_f$:

Algorithm 1: Iterative Refinement in four precisions

Solve $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{b}$ in precision u_f and store $\boldsymbol{x}_0$ at precision u_s.
for $i = 0, 1, \ldots$ **do**
  Compute $\boldsymbol{r}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$ at precision u_r and round $\boldsymbol{r}_i$ to precision u_s.
  Solve $\boldsymbol{A}\boldsymbol{d}_i = \boldsymbol{r}_i$ at precision u_s and store $\boldsymbol{d}_i$ at precision u.
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{d}_i$ at precision u.
end for

The idea is to start with an initial guess $\boldsymbol{x}_0$. Under each iteration, we compute the residual $\boldsymbol{r}_i$ of the current solution, compute a correcting term $\boldsymbol{d}_i$ using the residual as the right-hand side of the linear system we are solving, and add the correcting term to the current solution to create a new solution $\boldsymbol{x}_{i+1}$. These steps are repeated until the stopping criteria are reached.

Note that u indicates the unit roundoff, thus smaller values of u suggests higher precision. Variables with sub-indices in iteratvie methods indicate the value of the variable at the i-th iteration.

Since the initial guess is likely incorrect and heavily distorted by the error terms, the accuracy of this step is not the most important, it can thus be computed at a low precision to save computation expenses. On the other hand, the computation of the residual is the most important step of the algorithm that improves the accuracy of the solution, thus requires the term to be computed under a high precision. The solving of the correcting term is again using the ill-conditioned linear system thus could be computed under a relatively low precision.

In comparison to the traditional fixed precision iterative refinement solver, the mixed precision algorithm acquires faster computations within reasonable errors. However, in cases where $\boldsymbol{A}$ is ill-conditioned and the problem is ill-posed, the noise is maximized in step 1 and 3, thus regularization is needed, which we will cover in section 3.3

2.4.2 Iterated Tikhonov Regularization

As Tikhonov Regularization is a popular approach in solving ill-posed inverse problems, iterated Tikhonov regularization can yield numerical solutions with even higher accuracy [2].

As we solve for the problem $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{b}$ using the iterated Tikhonov method, the steps are similar to those of Iterative Refinement (under uniform precision), except that the step of computing the correcting term $\boldsymbol{d}_i$ will be solving the same least squares problem as the one used in the Tikhonov regularization.

Algorithm 2: Iterative Tikhonov method

Solve $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{b}$.
for $i = 0, 1, \ldots$ **do**
  Compute $\boldsymbol{r}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$.
  Solve $\min_{\boldsymbol{d} \in \mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{d} - \boldsymbol{r}_i||^2 + \alpha_i^2 ||\boldsymbol{d}||^2$.
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{d}_i$.
end for

The choice of α_i can be stationary or non-stationary depending on the problem. If stationary, $\alpha_1 = \alpha_2 = \cdots = \alpha_n$. If non-stationary α_i is dependent on i. We will discuss more details about the parameters in section 3.3.1.

2.4.3 Krylov Subspace Methods

Krylov subspaces are subspaces of the form

$$\mathcal{K}_m(\boldsymbol{M}, \boldsymbol{v}) := \text{span}\{\boldsymbol{v}, \boldsymbol{M}\boldsymbol{v}, \boldsymbol{M}^2\boldsymbol{v}, \ldots, \boldsymbol{M}^{m-1}\boldsymbol{v}\} \tag{2.7}$$

with $\boldsymbol{M}$ being an $n \times n$ matrix and $\boldsymbol{v}$ being a vector of size n [15].

Algorithms introduced in this section all utilize the Krylov subspaces, with different choices for $\boldsymbol{M}$ and $\boldsymbol{v}$, such as $\boldsymbol{M} = \boldsymbol{A}$ or $\boldsymbol{A}^\top \boldsymbol{A}$, and $\boldsymbol{v} = \boldsymbol{b}$.

Conjugate Gradient

The Conjugate Gradient (CG) method is unique among Krylov subspace methods for it only requires the memory to store four vectors at a time. This algorithm is based on the Lanczos algorithm that reduces a matrix to symmetric tridiagonal form. It is similar to the steepest descent method but searches in the gradient search directions that are A-conjugate [4]. The algorithm is listed below:

Algorithm 3: Conjugate Gradient method

Start with an initial guess $\boldsymbol{x}_0$
Compute $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$
$\boldsymbol{p}_0 = \boldsymbol{r}_0$
for $k = 0, 1, \ldots$ **do**
  $\alpha_k = (\boldsymbol{r}_k^T \boldsymbol{r}_k)/(\boldsymbol{p}_k^T \boldsymbol{A}\boldsymbol{p}_k)$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_i + \alpha_k \boldsymbol{p}_k$
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_i - \alpha_k \boldsymbol{A}\boldsymbol{p}_k$
  $\beta_k = (\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1})/(\boldsymbol{r}_k^T \boldsymbol{r}_k)$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$
end for

A restriction of CG is that it can only be implemented on symmetric positive definite (SPD) matrices. The algorithm has convergence rate

$$\|\boldsymbol{e}_k\|_{\boldsymbol{A}} \leq \left(\frac{\sqrt{\kappa(\boldsymbol{A})} - 1}{\sqrt{\kappa(\boldsymbol{A})} + 1} \right)^k \|\boldsymbol{e}_0\|_{\boldsymbol{A}}$$

where $\kappa(\boldsymbol{A})$ refers to the condition number of $\boldsymbol{A}$.

This result implies that convergence is faster for well-conditioned matrices, which means that $\kappa(\boldsymbol{A}) \approx 1$. The algorithm converges fast if the eigenvalues of $\boldsymbol{A}$ are tightly clustered around a nonzero value.

LSQR

For least squares problems, the CG method can be applied to the normal equations,

$$\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{A}^\top \boldsymbol{b}.$$

Two well-known implementations of the approach are CGLS and LSQR.

GMRES

The Generalized Minimum Residual Method (GMRES) is a projection method based on the Krylov subspaces. It utilizes the Arnoldi process to construct the basis and minimizes the norm of the residual at each iteration. It also utilizes the decomposition of an upper Hessenberg matrix. The algorithm generated is listed below [15].

Algorithm 4: GMRES

Compute $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\beta := \|\boldsymbol{r}_0\|$, and $\boldsymbol{v}_1 := \boldsymbol{r}_0/\beta$
Define matrix $\boldsymbol{H}_m := 0$ of size $(m+1) \times m$ where h_{ij} indicates element
for $j = 0 : m$ **do**
  Compute $\boldsymbol{w}_j := \boldsymbol{A}\boldsymbol{v}_j$
  for $i = 1 : j$ **do**
    $h_{ij} := (\boldsymbol{w}_j, \boldsymbol{v}_j)$
    $\boldsymbol{w}_j = \boldsymbol{w}_j - h_{ij}\boldsymbol{v}_i$
  end for
  $h_{j+1,j} = \|\boldsymbol{w}_j\|$
  if $h_{j+1,j} = 0$ **then**
    $m = j$
    Break loop
  end if
  $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$
end for
Compute $\boldsymbol{y}_m = \min_{\boldsymbol{y}} ||\beta \boldsymbol{e}_1 - \boldsymbol{H}_m \boldsymbol{y}||_2$
$\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m$

GMRES can only be applied to square matrices, but symmetry is not required.

Preconditioning

The idea of preconditioning is to apply the Krylov subspace methods to another system $\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ such that $\hat{\boldsymbol{A}}$ has more clustered eigenvalues than $\boldsymbol{A}$ that would reduce the number of iterations.

Common selections of preconditioners are generated by incomplete LU factorization (ilu) and incomplete Cholesky factorization (ichol). Take ichol for example, find an upper triangular matrix $\boldsymbol{R}$ such that $\boldsymbol{A} \approx \boldsymbol{R}^\top \boldsymbol{R}$. The new linear system can then be written as

$$\boldsymbol{R}^{-\top} \boldsymbol{A} \boldsymbol{R}^{-1} \boldsymbol{R}\boldsymbol{x} = \boldsymbol{R}^{-\top} \boldsymbol{b} \tag{2.8}$$

where $\hat{\boldsymbol{A}} = \boldsymbol{R}^{-\top} \boldsymbol{A} \boldsymbol{R}^{-1}$, $\hat{\boldsymbol{x}} = \boldsymbol{R}\boldsymbol{x}$, and $\hat{\boldsymbol{b}} = \boldsymbol{R}^{-\top}\boldsymbol{b}$.

When preconditioning is applied to CG, the algorithm can be called PCG.

2.4.4 GMRES-based iterative refinement

GMRES-based iterative refinement (GMRES-IR) is an algorithm proposed by Carson and Higham [11] that improves the iterative refinement using GMRES.

Algorithm 5: GMRES-based iterative refinement

Compute an LU factorization $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ in precision u_f
Solve $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{b}$ in precision u_f and store $\boldsymbol{x}_0$ at precision u_s.
for $i = 0, 1, \ldots$ **do**
  Compute $\boldsymbol{r}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$ at precision u_r and round $\boldsymbol{r}_i$ to precision u_s.
  Solve $\boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{d}_i = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{r}_i$ by GMRES at precision u_s and store $\boldsymbol{d}_i$ at precision u.
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{d}_i$ at precision u.
end for

In the algorithm proposed, the selection of the preconditioner is constructed by LU factorization, but we will discuss different selections in later sections.

Chapter 3

Methods

In this section, we introduce the methods used in this thesis to develop the numerical results. Some are implementations of previous methods under mixed precision arithmetic, and some are newly developed methods.

3.1 Tikhonov Regularization under Mixed Precision Arithmetic

As low precision only guarantees sufficient accuracy for certain data, we can improve the Tikhonov regularization method through performing certain computations of the SVD under low precision.

Traditional Tikhonov regularization has one parameter α. In this work, we propose an algorithm that implements Tikhonov regularization under mixed precision with three regularization parameters, k, α_1, and α_2. As recalled from equation (2.5) in section 2.3.2,

$$\boldsymbol{x}_F = \sum_{i=1}^{n} \frac{\sigma_i \boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i^2 + \alpha^2} \boldsymbol{v}_i.$$

In the new method, terms 1 through k are computed under double precision with filter factor $\phi_i = \frac{\sigma_i^2}{\sigma_i^2+\alpha_1^2}$ and terms $k+1$ through n are computed under half precision

with filter factor $\phi_i = \frac{\sigma_i^2}{\sigma_i^2+\alpha_2^2}$. Now that we take the indices into consideration, $\boldsymbol{x}_F = \boldsymbol{V}\boldsymbol{\Sigma}_F^\dagger \boldsymbol{U}^\top \boldsymbol{b}$ where

$$\boldsymbol{\Sigma}_F^\dagger = \operatorname{diag}\left(\frac{\sigma_1}{\sigma_1^2+\alpha_1^2}, \frac{\sigma_2}{\sigma_2^2+\alpha_1^2}, \cdots, \frac{\sigma_k}{\sigma_k^2+\alpha_1^2}, \frac{\sigma_{k+1}}{\sigma_{k+1}^2+\alpha_2^2}, \cdots, \frac{\sigma_n}{\sigma_n^2+\alpha_2^2}\right).$$

The new solution can be written as

$$\boldsymbol{x}_F = \sum_{i=1}^{k} \frac{\sigma_i \boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i^2+\alpha_1^2}\boldsymbol{v}_i + \sum_{i=k+1}^{n} \frac{\sigma_i \boldsymbol{u}_i^\top \boldsymbol{b}}{\sigma_i^2+\alpha_2^2}\boldsymbol{v}_i \tag{3.1}$$

Since α_1 and α_2 are continuous variables, we use the `fmincon` function from MATLAB to search for the optimal pair of filtering parameters for each k while enforcing the constraint that α_1 and α_2 are between the smallest and the largest singular values. The optimal set of parameters is then found by minimizing the relative error among each truncation index k and their corresponding sets of α's.

3.2 Choosing Regularization Parameters

As we introduced a few regularization methods in section 2.3 and section 2.4, most methods require an additional parameter such as k in TSVD and α in Tikhonov regularization. Selecting the optimal parameter improves the regularized solution. To determine the regularization parameters without the actual solution, there are two common practices, the Discrepancy Principle and the Generalized Cross Validation (GCV). While the Discrepancy Principle requires the knowledge and assumption about the error term $\|\boldsymbol{e}\|_2$, the GCV is an $\|\boldsymbol{e}\|_2$-free method [8].

3.2.1 Generalized Cross Validation

The GCV method is based on the assumption that the model should be able to predict a missing data point if removed. The GCV function of equation (3.2) thus calculates

the average error between the dropped data point, and the goal of the method is to minimize this error.

$$G = \frac{n\|\boldsymbol{A}\boldsymbol{x}_F - \boldsymbol{b}\|_2^2}{(\text{trace}(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}_F^\dagger))^2} \tag{3.2}$$

By the property of

$$\text{trace}(\boldsymbol{I} - \boldsymbol{\Sigma}\boldsymbol{\Sigma}_F^\dagger) = \sum_{i=1}^{n} (\frac{\alpha^2 \hat{b}_i}{\sigma_i^2 + \alpha^2})^2$$

where $\hat{\boldsymbol{b}} = \boldsymbol{U}^\top \boldsymbol{b}$ and

$$\|\boldsymbol{A}\boldsymbol{x}_F - \boldsymbol{b}\|_2^2 = \left\|(\boldsymbol{\Sigma}\boldsymbol{\Sigma}_F^\dagger - \boldsymbol{I})\boldsymbol{U}^\top \boldsymbol{b}\right\|_2^2,$$

we retrieve the GCV function below in equation (3.3) for three regularization parameters for the implementation of mixed precision Tikhonov regularization.

$$\begin{aligned} G(\alpha_1, \alpha_2, k) &= \frac{n\|\boldsymbol{A}\boldsymbol{x}_F - \boldsymbol{b}\|_2^2}{(\text{trace}(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}_F^\dagger))^2} \\ &= \frac{n\left\|(\boldsymbol{\Sigma}\boldsymbol{\Sigma}_F^\dagger - \boldsymbol{I})\boldsymbol{U}^\top \boldsymbol{b}\right\|_2^2}{(\text{trace}(\boldsymbol{I} - \boldsymbol{\Sigma}\boldsymbol{\Sigma}_F^\dagger))^2} \\ &= \frac{k\sum_{i=1}^{k}(\frac{\alpha_1^2 \hat{b}_i}{\sigma_i^2 + \alpha_1^2})^2 + (n-k)\sum_{i=k+1}^{n}(\frac{\alpha_2^2 \hat{b}_i}{\sigma_i^2 + \alpha_2^2})^2}{(\sum_{i=1}^{k}(\frac{\alpha_1^2}{\sigma_i^2 + \alpha_1^2}) + \sum_{i=k+1}^{n}(\frac{\alpha_2^2}{\sigma_i^2 + \alpha_2^2}))^2} \end{aligned} \tag{3.3}$$

To solve the optimization problem, we use the MATLAB function `fmincon` to restrict the parameter range between the smallest and the largest singular value.

3.2.2 Discrepancy Principle

The Discrepancy Principle is a $\|\boldsymbol{e}\|$-based method that restricts the residual of the solution to be close to the error norm such that $\|\boldsymbol{A}\boldsymbol{x}_F - \boldsymbol{b}\|_2^2 - \|\boldsymbol{e}\|_2^2 = 0$.

As we want to find a regularization parameter that satisfies the previous equation,

our goal is to solve for

$$D(\lambda) = \|\boldsymbol{A}\boldsymbol{x}_F - \boldsymbol{b}\|_2^2 - \|\boldsymbol{e}\|_2^2$$

where λ refers to the set of regularization parameters.

Considering $\boldsymbol{\Sigma}_F^\dagger = \text{diag}\left(\frac{\sigma_1}{\sigma_1^2+\alpha_1^2}, \frac{\sigma_2}{\sigma_2^2+\alpha_1^2}, \cdots, \frac{\sigma_k}{\sigma_k^2+\alpha_1^2}, \frac{\sigma_{k+1}}{\sigma_{k+1}^2+\alpha_2^2}, \cdots, \frac{\sigma_n}{\sigma_n^2+\alpha_2^2}\right)$ from section 3.1 and using similar properties as the deduction of the GCV function, we derive a discrepancy function for mixed precision Tikhonov regularization as shown in equation (3.4).

$$\begin{aligned} D(\alpha_1, \alpha_2, k) &= \|\boldsymbol{A}\boldsymbol{x}_F - \boldsymbol{b}\|_2^2 - \|\boldsymbol{e}\|_2^2 \\ &= \left\|(\boldsymbol{\Sigma}\boldsymbol{\Sigma}_F^\dagger - \boldsymbol{I})\boldsymbol{U}^\top \boldsymbol{b}\right\|_2^2 - \epsilon^2 \\ &= \sum_{i=1}^{k} (\frac{\alpha_1^2 \hat{b}_i}{\sigma_i^2 + \alpha_1^2})^2 + \sum_{i=k+1}^{n} (\frac{\alpha_2^2 \hat{b}_i}{\sigma_i^2 + \alpha_2^2})^2 - \epsilon^2 \end{aligned} \tag{3.4}$$

where $\epsilon = \|\boldsymbol{e}\|$.

To solve the equation, we use the MATLAB function `fsolve` to find the closest set of multiple parameters of the function to zero.

A disadvantage of the Discrepancy method is that it requires the knowledge of the noise level. In case of absence of such knowledge, we refer back to the GCV as mentioned in section 3.2.1.

3.3 Iterative Refinement with Tikhonov Regularization

To improve the efficiency of the Iterated Tikhonov Regularization method [2], we implemented the algorithm under mixed precision arithmetic as well. As Iterative Refinements [3] involve frequent operations of changing precisions, the codes are implemented in Julia for its easier conversion of Floating-point types.

To regularize the Iterative Refinement method as referred to in section 2.4.1, we select the Tikhonov regularization method, which involves solving a least-square problem of

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}||^2+\alpha||\boldsymbol{x}||^2$$

as a replacement of step 1 and 4 in the previously stated algorithm. We reduce the number of precisions to two where $u_h \leq u_l$, and the updated mixed precision iterated Tikhonov algorithm is shown as below. Again note, since u indicates the unit-roundoff of the precision, smaller values of u_h suggests that u_h is the higher precision.

Algorithm 6: Iterative Refinement with Tikhonov Regularization

1: Solve $||\boldsymbol{A}\boldsymbol{x}_0-\boldsymbol{b}||^2+\alpha||\boldsymbol{x}_0||^2$ and store $\boldsymbol{x}_0$ in precision u_l.
2: **for** $i=0,1,\ldots$ **do**
3: Compute $\boldsymbol{r}_i=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_i$ at precision u_h and round $\boldsymbol{r}_i$ to precision u_l.
4: Solve $\min_{\boldsymbol{d}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{d}_i-\boldsymbol{r}_i||^2+\alpha_i^2||\boldsymbol{d}_i||^2$ at precision u_l and store $\boldsymbol{d}_i$ at precision u_h.
5: $\boldsymbol{x}_{i+1}=\boldsymbol{x}_i+\boldsymbol{d}_i$ at precision u_h.
6: **end for**

3.3.1 Non-stationary parameter

Note that, as the algorithm iterates, the filtering parameter converges to zero and fails to serve the purpose of regularization. Thus, we choose a list of non-stationary regularization parameters that is dependent on the iteration number, such that we adjust the regularization parameter as the residual decreases for each iteration.

A common selection of the non-stationary regularization parameters is the geometric sequence, where we start with a positive α_0, and $\alpha_i=\alpha_0 q^i$ where q is a positive real number between 0 and 1, and i is the number of iterations ranging from 1 to n.

This choice of regularization parameter is studied by Hanke [5] and proved to reach the stopping criteria of the discrepancy principle within $O(|\delta|)$ iterations with δ being the error level.

3.4 Preconditioning

As mentioned in section 2.4.3, preconditioning is often used with Krylov subspace iterative methods to reduce the number of iterations needed towards convergence. While common selections of preconditioners like the incomplete LU factorization and the incomplete Cholesky factorization are effective, we will be using a preconditioner that also regularizes the solver.

The idea is to use TSVD to regularize the preconditioner [14]. For $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, we construct a preconditioner $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{\Sigma}_T\boldsymbol{V}^\top$ where $\boldsymbol{\Sigma}_T = \text{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$. A problem with this approach is that the preconditioner $\boldsymbol{P}_T$ is singular, such that the computed solution $\boldsymbol{x}_T = \boldsymbol{V}\boldsymbol{\Sigma}_T^{-1}\boldsymbol{U}^\top\boldsymbol{b}$ cannot be guaranteed to be nonzero, which will break the conjugate gradient method.

An alternative approach is to replace the zero in $\boldsymbol{\Sigma}_T$ with ones, such that $\boldsymbol{\Sigma}_\tau = \text{diag}(\sigma_1, \ldots, \sigma_k, 1, \ldots, 1)$, and $\boldsymbol{P}_\tau = \boldsymbol{U}\boldsymbol{\Sigma}_\tau\boldsymbol{V}^\top$. This approach was proposed by Hanke, Nagy, and Plemmons [6].

When using the alternative approach, the preconditioned system has the form

$$\begin{aligned}
\boldsymbol{P}_\tau^{-1}\boldsymbol{A} &= (\boldsymbol{U}\boldsymbol{\Sigma}_\tau\boldsymbol{V}^\top)^{-1}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \\
&= \boldsymbol{V}\boldsymbol{\Sigma}_\tau^{-1}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \\
&= \boldsymbol{V}\boldsymbol{\Sigma}_\tau^{-1}\boldsymbol{\Sigma}\boldsymbol{V}^\top \\
&= \boldsymbol{V}\boldsymbol{\Delta}\boldsymbol{V}^\top
\end{aligned} \tag{3.5}$$

where

$$\boldsymbol{\Delta} = \boldsymbol{\Sigma}_\tau^{-1}\boldsymbol{\Sigma} = \text{diag}(1, \ldots, 1, \sigma_{k+1}, \ldots, \sigma_n).$$

In this case, the singular values form two clusters. The larger singular values that correspond to the signal subspace are clustered around one and well separated from the small singular values that correspond to the noise subspace.

Since the convergence of the conjugate gradient method is dependent on the clustered singular values, the regularized solution only needs one iteration in this case.

Building upon the TSVD approach, as the LSQR method is CG applied to the normal equations $\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b}$, applying the method to the least-square representation of Tikhonov regularization method $\min_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}||^2+\alpha^2||\boldsymbol{x}||^2$ can further reduce the disruption of noise.

The normal equations of the minimization problem that we just mentioned is

$$(\boldsymbol{A}^\top \boldsymbol{A} + \alpha^2 \boldsymbol{I})\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b} \tag{3.6}$$

Since $\boldsymbol{A}^\top \boldsymbol{A}$ is symmetric positive semidefinite and $\boldsymbol{A}^\top \boldsymbol{A} + \alpha^2 \boldsymbol{I}$ is symmetric positive definite, we can apply the Cholesky factorization such that

$$\boldsymbol{A}^\top \boldsymbol{A} + \alpha^2 \boldsymbol{I} \approx \boldsymbol{R}^\top \boldsymbol{R} \tag{3.7}$$

where $\boldsymbol{R}$ is an upper-triangular matrix and our preconditioner.

When combining TSVD and Tikhonov, another preconditioner we can use is

$$\boldsymbol{V}\boldsymbol{\Delta}\boldsymbol{V}^\top = \boldsymbol{R}_\tau^\top \boldsymbol{R}_\tau \tag{3.8}$$

We can also use $\boldsymbol{\Sigma}_T$ as mentioned earlier in this section, such that we will form $\boldsymbol{R}_T$ using the same method as equation (3.8) except replacing $\boldsymbol{\Delta}$ with $\boldsymbol{\Sigma}_T^{-1}\boldsymbol{\Sigma}_T$.

Chapter 4

Numerical Experiments

4.1 Data

Our numerical experiments involve two main datasets.

We took a sample inverse problem from the MATLAB Regularization Tools toolbox where $\boldsymbol{A}$ represents a first kind Volterra integral equation with [0,1] as integration interval [9]. This sample problem will be referred to as `heat` and is very ill-conditioned. Figure 4.1 shows the original and the observed signal of the problem.

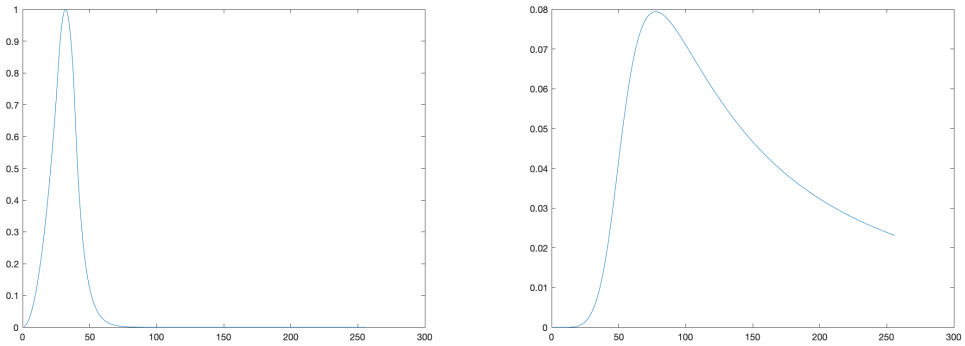

Figure 4.1: The left figure shows the exact solution, and the right figure shows the right-hand side of `heat`.

Another test problem taken from the toolbox is a discretization of a first kind Fredholm integral equation whose kernel K is the Green's function for the second

derivative [9]. This sample problem is mildly ill-posed and will be referred to as `deriv2`. Figure 4.2 shows the original and the observed signal of the problem.

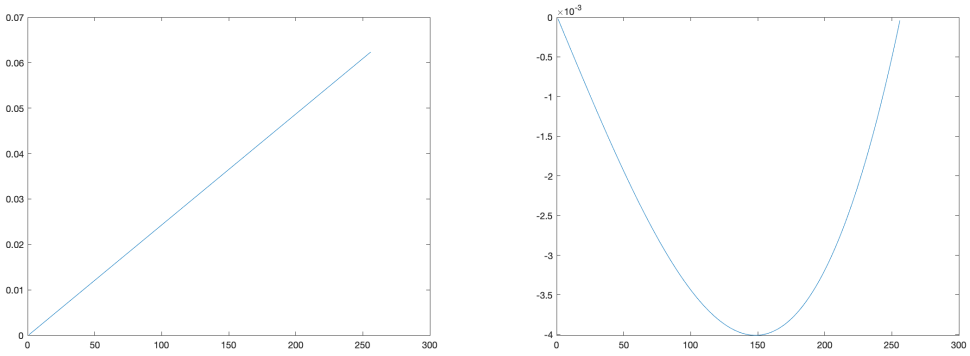

Figure 4.2: The left figure shows the exact solution, and the right figure shows the right-hand side of `deriv2`.

The other sample problem is to improve the resolution of gamma-ray spectra. We simulate an x-ray spectrum as introduced by Trussell [16]. The true solution is a single signal of length 64 points consisting of four data peaks. The point spread function (PSF) is constructed using a Gaussian blur point spread function. This sample problem will be referred to as `spectra`. Figure 4.3 shows the original spectra and the signal that went through the Gaussian blur PSF.

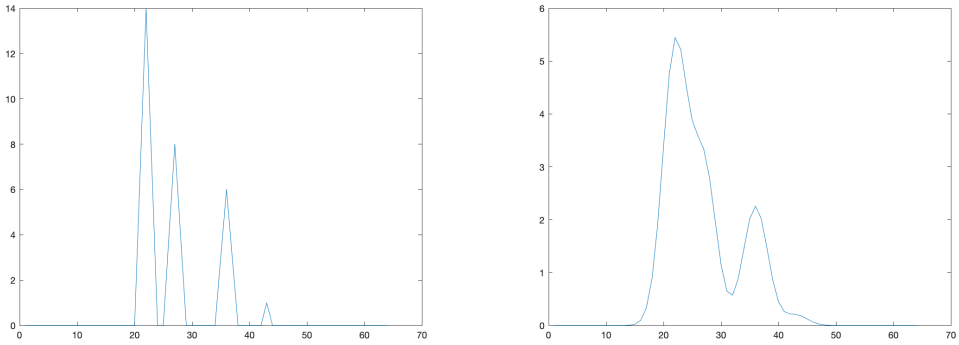

Figure 4.3: The left figure shows the exact signal of `spectra`, and the right figure shows the signal going through the Gaussian blur PSF.

As the test problems are generated with MATLAB tools, they are saved as a `.mat` file for usage in Julia.

4.2 Implementation of Mixed Precision Arithmetic

To perform mixed precision arithmetic under the double precision environment in MATLAB, we implemented the `chop` function that simulates low precision arithmetic. The package was developed by Higham and Pranesh [13] where every operation is rounded using one of the four IEEE arithmetic rounding modes: round to nearest with ties broken by rounding to an even least significant bit, round towards plus infinity or minus infinity, and round towards zero. A problem with this approach is that MATLAB only supports datatypes `single` and `double` in its default environment, and data with manually chopped precisions do not carry the rounding to the later operations, which means that half precision can only be performed if we chop every single operation.

Thus, we switched to Julia for the later numerical experiments, where we used the default function `convert(::TypeMyType, x) = MyType(x)` of the language to convert datatypes to achieve mixed precision arithmetic.

4.3 Stopping Criteria

As mentioned in section 2.4, the methods that involve iterations require stopping criteria. For the numerical experiments introduced in this chapter, two types of stopping criteria will be enforced - the maximum number of iterations and the tolerance.

The maximum number of iterations stopping criteria is typically referred as an integer parameter of `MaxIts` in our algorithms, where the parameter is set based on the computing resource and environment. A common selection of MaxIts value in this chapter is 100. The iteration terminates when the number of iteration reaches `MaxIts`.

The tolerance stopping criteria is typically referred as a constant parameter of `tol` and is often set by the discrepancy principle where $tol = \tau\delta$. δ is the bound of the

unknown noise such that $\|\boldsymbol{e}\| \leq \delta$, and $\tau > 1$ is a user-supplied constant independent of δ [2]. Our selection of τ is mostly 1.01. The iteration terminates when the norm of the residual is smaller than `tol`, which can be expressed as $\|\boldsymbol{r}_{k+1}\| \leq \tau\delta$. This can also be written as $\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|} < \tau\nu$ where ν is the noise level such that $\nu = \frac{\delta}{\|\boldsymbol{b}\|}$. A common selection of ν when generating data in this thesis is $\nu = 0.05$.

4.4 Tikhonov Regularization under mixed precision

In this section, we will look at the behavior of Tikhonov Regularization under mixed precision with different sets of regularization parameters. Data will be generated from the sample problem `heat`, and computations will be done under MATLAB.

We iterate through every possible truncation index k which ranges from 1 to n and find the optimal pair of α's using the MATLAB function `fminsearch` by minimizing the relative error produced by the objective function. We then find the index k in correspondence to the minimized relative error to find the optimal set of regularization parameters. However, this method can only be used when we know the true solution.

Under mixed precision, the optimal sets of α's are nearly equal for different k-values. We thus label each set of the parameters by the k-value and plot each set with its corresponding error to visualize the optimization.

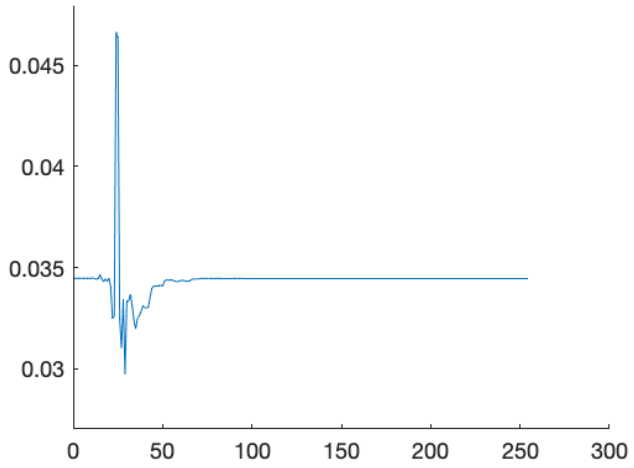

Figure 4.4: Relative error (y-axis) at different truncation indices (x-axis) for computing `heat` using Tikhonov regularization under mixed precision arithmetic.

In Figure 4.4, the x-axis is the k-index, the truncation index of terms where the computation switches from the standard computing precision to a lower precision, which in this case is from double precision to half precision, and the y-axis is the normalized error. As indicated in the plot, while most selections of k produce similar error as does the uniform precision algorithm, which is approximately 0.03445, certain truncation indices produce significantly improved accuracy. At $k = 29$, the error between the calculated data and the original data is only 0.0297. The pattern of certain k-indices in mixed precision having significantly improved accuracy in comparison to uniform double precision and other k-indices in mixed precision holds for this test case with other randomly generated noise vectors and other similar test cases.

In order to validate the output, we perform a few numerical experiments.

In Figure 4.5, we graph the singular values σ_i in red, the noise-free Fourier coefficients $\boldsymbol{U}^\top \boldsymbol{b}_c$ in blue, and the noise-corrupted Fourier coefficients $\boldsymbol{U}^\top \boldsymbol{b}$ in green where $\boldsymbol{b}_c = \boldsymbol{A}\boldsymbol{x}$ is the noise-free data, and $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$ is the noise-corrupted data. It is obvious that the singular values decay towards zero such that regularization is needed. The discrete Picard condition states that if the Fourier coefficients decay

to zero faster than the generalized singular values, then the regularized solutions are guaranteed to have approximately the same properties as the exact solution [7]. The graph follows the Picard condition in which the noise-corrupted terms oscillate and deviate from the noise-free computed output starting from the optimal truncation index at $k = 30$ that we observed.

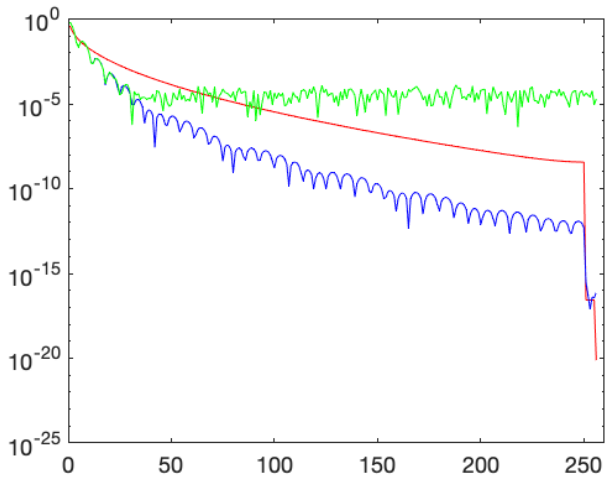

Figure 4.5: The graph illustrates the Picard condition of `heat` with truncation index k = 30. The red line shows the singular values, the green line shows the noise-corrupted Fourier coefficients, and the blue line shows the noise-free Fourier coefficients.

4.4.1 Methods selecting regularization parameters

We use the same problem to test the optimality of the parameters found using GCV and the discrepancy principle. The numerical experiments conducted under this section are computed using MATLAB.

From the previous numerical experiments conducted using the real solution, we learn that the optimal group of regularization parameters are $\alpha_1 = 0.0006045$, $\alpha_2 = 0.185944$, and $k = 29$. The following experiments are conducted with the initial search set close to or far away from the optimal result.

For GCV, when the initial search is set close to the optimal parameters, the resulting error is 0.248, which is much greater than the average error shown in Figure 4.4,

and the error of a non-optimal initial search is 0.275, even worse. When we modify the function to be in terms of only α_1 and α_2, we are not able to find an optimal pair of parameters since the initial search is likely a local minimum. The function would thus return the initial search. The GCV method does not pass the numerical experiments.

For the discrepancy principle method, we set the initial search to be $\alpha_1 = 0.06$, $\alpha_2 = 0.1$, and modify the k values to observe changes. When k is set to 29, the optimal k, the resulting error is 0.0358, slightly higher than the uniform precision error but still an acceptable optimal result. When k is set to 25, the error improves to be 0.0314. This method produces optimal parameters, we thus test it on different noise levels. In one of the test cases, we set the initial search to be at the local maximum where $k = 25$ as shown in Figure 4.6. The optimal parameters given by this method produces an error of 0.0358, which not only averts a local maximum error, but also significantly improves the accuracy in comparison to the error of 0.0380 computed with only one precision.

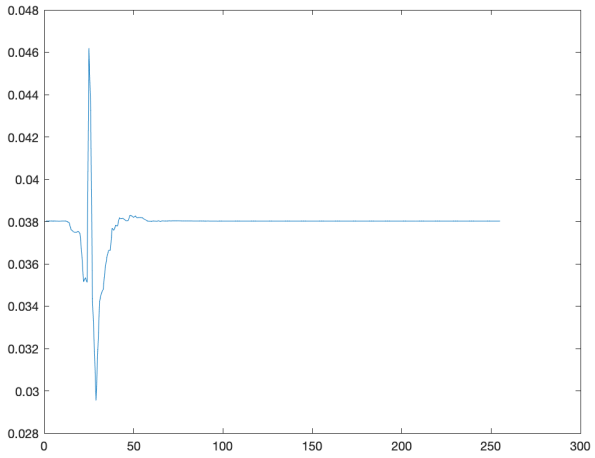

Figure 4.6: Relative error (y-axis) at different truncation indices (x-axis) for computing `heat` tested using the discrepancy principle.

The discrepancy principle method is tested to be optimal and robust for this

specific test problem, we thus test the method on another test problem, `deriv2`. While the method produces a relatively optimal result for most test problems, its effect significantly deteriorates in a very ill-posed problem with the singular values quickly declining as shown in Figure 4.7.

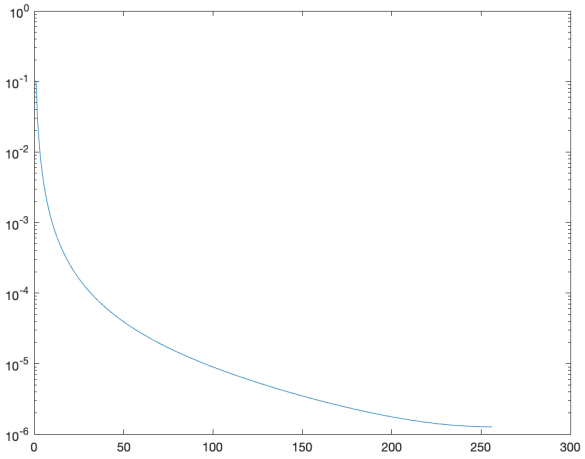

Figure 4.7: The semilog of the singular values of `deriv2`.

As shown in Figure 4.8, implementing mixed precision does not produce smaller error in a severely ill-posed problem as no significant local minimums are found.

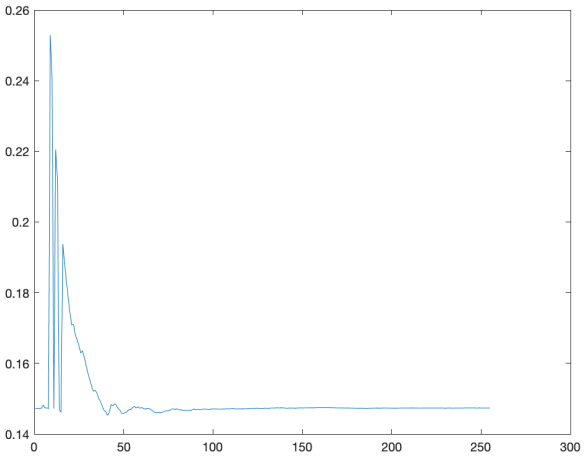

Figure 4.8: Relative error (y-axis) at different truncation indices (x-axis) for computing `deriv2` tested using the discrepancy principle.

4.5 Iterative Refinement for Tikhonov

We use the same sample inverse problem, `heat`, as the one used in Section 4.4. In this section, we will test the accuracy of the Iterative Refinement for Tikhonov as mentioned in section 3.3 across different combinations of precisions. The computations will be done in Julia.

The initial α we use as the regularization parameter is $\sqrt{\alpha_0}$ where α_0 is the optimal α found in Figure 4.4. Since we are taking the square root of α_0, it is no longer optimal.

For the first set of numerical experiments, we set the maximum iteration number to be 200, and label the legend in the format of $u_h + u_l$. As shown in Figure 4.9, there is no significant difference in the different combination of the precisions. As we zoom into the plot, we observe that the lower precisions generate a slightly larger error. However, the relative error between (double, single) and (double, half) is 0.0076, the relative error between (double,single) and (single, half) is 0.0077, and the relative error between (double, half) and (single, half) is only 0.00014. We can thus conclude that lower precision will not cause significant error in the result other than accelerating the computation.

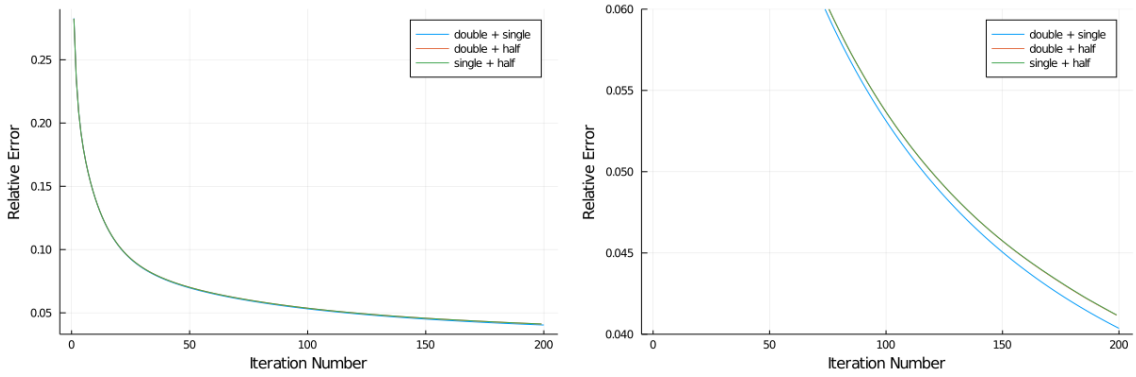

Figure 4.9: The left figure shows the relative errors at iterations 1 to 200 for `heat` computed using Iterative Refinement for Tikhonov. The right figure is a zoomed in version of the left figure to illustrate the difference between the computation under different mixed precision combinations.

When we set the regularization parameter as $\alpha = \alpha_0$, we observe that the error plot

no longer converges and increases instead as the number of iteration increases. We thus increased maximum iteration number to 5000 instead of 200 for $u_h = \text{Float64}$ and $u_l = \text{Float32}$. We observe from Figure 4.10 that the algorithm is semi-convergent. The regularization parameter gets smaller as the algorithm iterates, and after a certain threshold where the error is minimized, where $error = 0.3168$, the regularization parameter becomes too small for it to be functional. It is thus important to end the refinement process after a certain number of iterations.

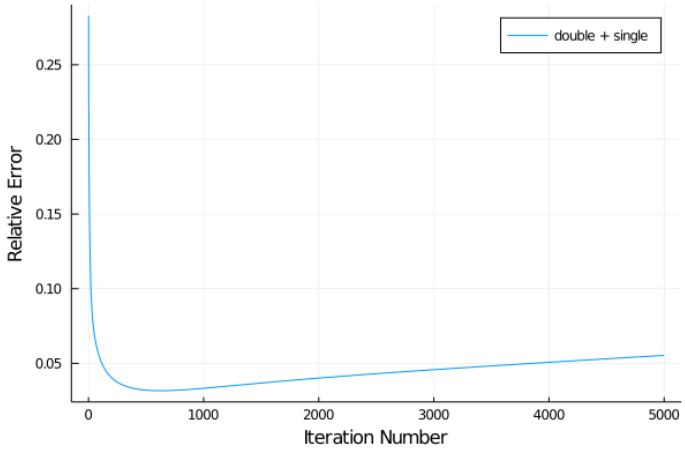

Figure 4.10: The figure illustrates the semi-convergence behavior of stationary regularization parameters by showing the relative error at iterations 1 to 5000 of `heat` computed using Iterative Refinement under Tikhonov. The computation is done with the higher precision set to double precision and the lower precision set to single precision.

To study the effectiveness of non-stationary parameters as mentioned in section 3.3.1, we compare a set of results with stationary and non-stationary parameters when other parameters are held the same.

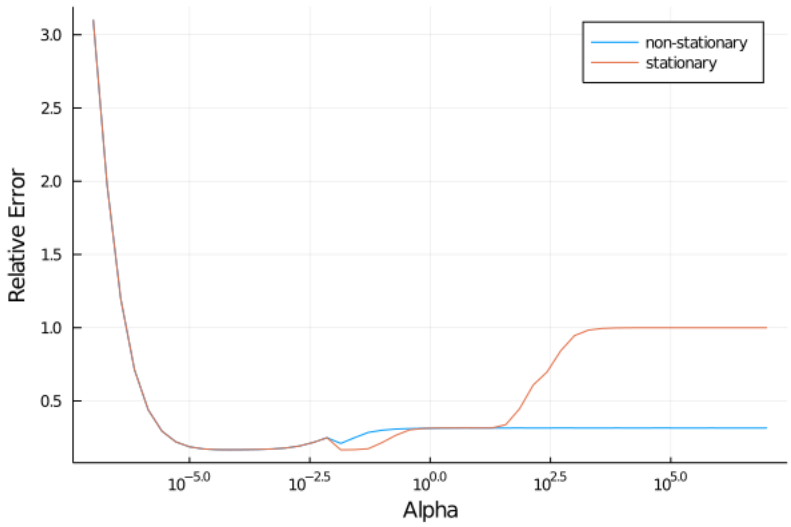

Figure 4.11: The figure compares the relative error of `heat` computed with stationary and non-stationary parameters with maximum number of iterations being 1000 using Iterative Refinement under Tikhonov.

In this experiment, we are using single and half precisions. We can observe directly from Figure 4.11 that the non-stationary method has smaller relative errors for the large regularization parameters where more iterations are needed. The non-stationary parameters thus serve the purpose of preventing the filtering parameter from getting too close to 1.

4.6 Preconditioned Conjugate Gradient method

In this section, we conduct numerical experiments to compare CG with and without the preconditioners constructed from section 3.4. Data will be generated from the sample problem `heat`, and computations are done under Julia.

Figure 4.12 illustrates the behavior of the algorithm with `MaxIts` being the only stopping criteria, and Figure 4.13 adds a tolerance using the discrepancy principle. While the algorithm converges within tolerance, it behaves differently for different precision environments and does not necessarily converge.

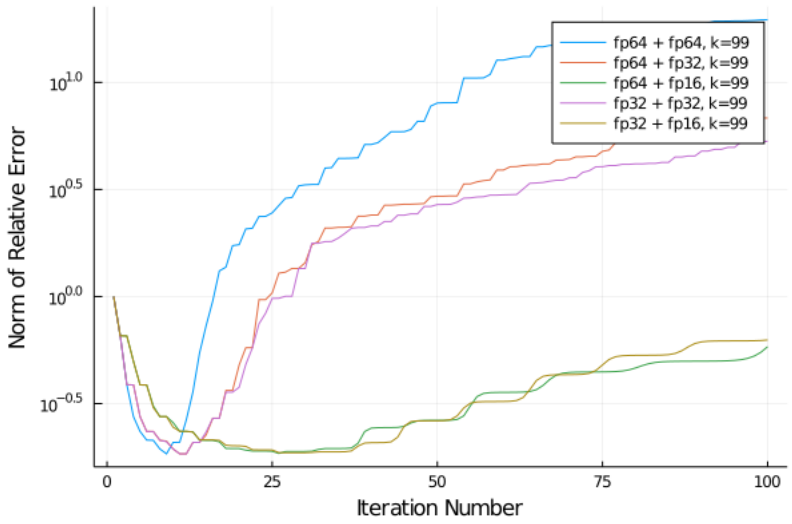

Figure 4.12: The figure shows the relative error of `heat` computed with PCG using $\boldsymbol{R}_\tau$ with the number of iterations being the only stopping criteria.

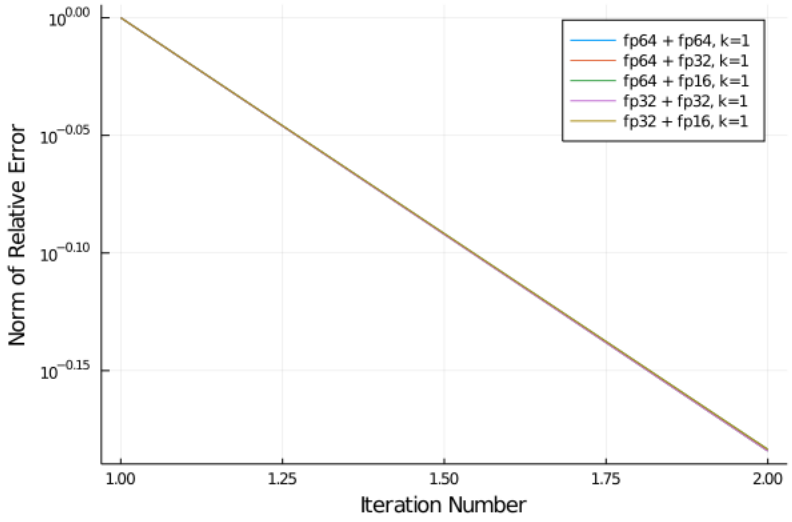

Figure 4.13: The figure shows the relative error of `heat` computed with PCG using $\boldsymbol{R}_\tau$ with a tolerance set as the stopping criteria. The method converges in two iterations.

4.7 Preconditioned GMRES-IR

In this section, we will conduct numerical experiments to compare GMRES-IR with and without the preconditioners constructed from section 3.4. Data will be generated from the sample problem `spectra`, and computations are done in Julia.

Figure 4.14 illustrates the accuracy of the solution computed with GMRES-IR using the truncated preconditioner $\boldsymbol{R}_T$, and Figure 4.15 illustrates the norm of the relative error using the same algorithm with the untruncated preconditioner $\boldsymbol{R}$ respectively.

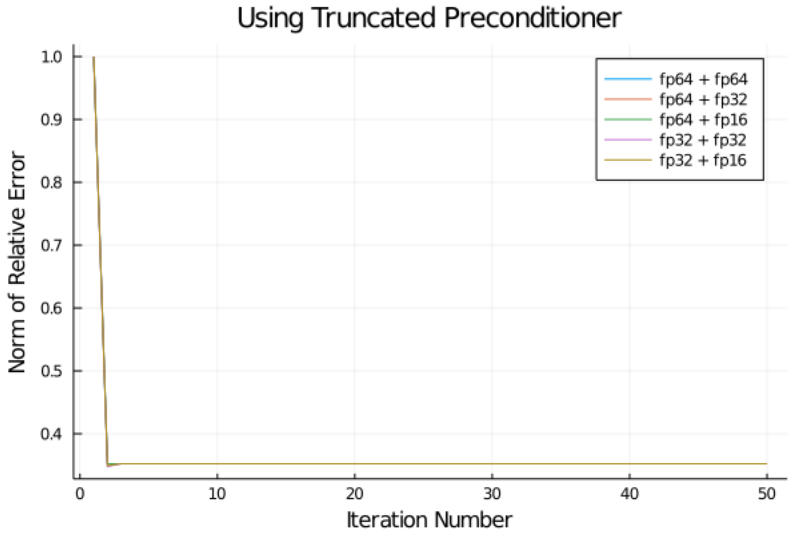

Figure 4.14: The figure shows the relative error of `spectra` computed with GMRES-IR using $\boldsymbol{R}_T$.

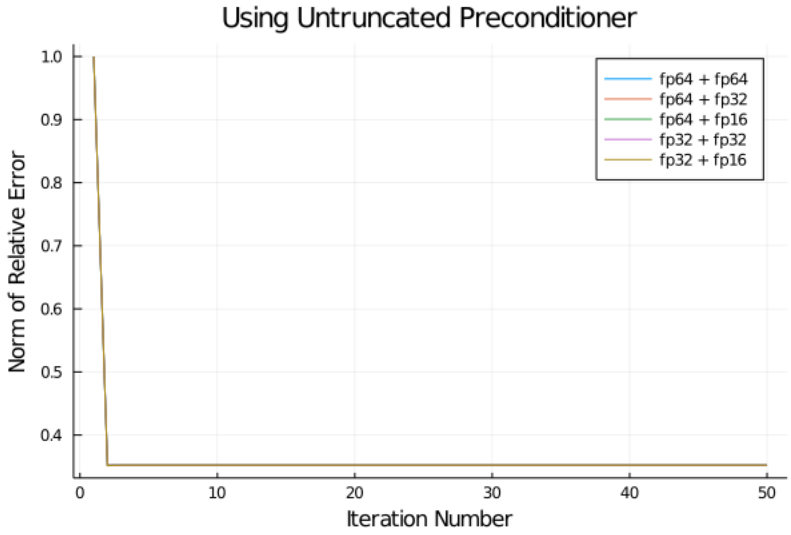

Figure 4.15: The figure shows the relative error of `spectra` computed with GMRES-IR using $\boldsymbol{R}$.

From the figures, we can observe that both converge within very few iterations.

No significant difference is observed but both preconditioners serve to regularize the solution.

The rounding of the different precisions leave small differences in the relative error. The solution we computed using the truncated preconditioner has relative error larger than the one computed using the untruncated preconditioner only by a very small margin as shown in Figure 4.16

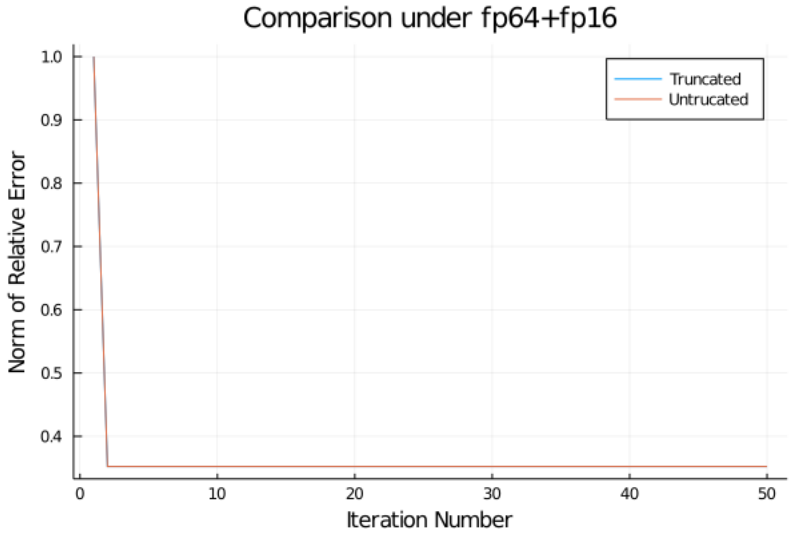

Figure 4.16: The figure compares the behavior of GMRES-IR computed using $\boldsymbol{R}$ and $\boldsymbol{R}_T$ which almost overlaps.

Chapter 5

Concluding Remarks

From the numerical experiments, we can conclude that mixed precision arithmetic can be more accurate while taking less storage to compute for different regularization methods, like the Tikhonov method, in solving ill-posed inverse problems. While improving the accuracy of our algorithm using the Iterated Tikhonov method and non-stationary regularization parameters, it is also important to set our stopping criteria using the discrepancy principle.

In some other iterative methods, the computations under mixed precision arithmetic reduce computational cost while maintaining sufficiently appropriate accuracy of the final computed solution. We have also constructed preconditioners that improve the accuracy and reduce the computational cost by reducing the number of iterations.

The algorithms can be used in improving the robustness in image restoration, blind deconvolution, remote sensing, and other fields. As a common concern about the Tikhonov regularization method is that it often over-smooths the edges, a possible solution of improvement is to combine Tikhonov with the Total variation regularization method into a Hybrid regularization method. As Hybrid regularization uses adaptive weighted parameters, the implementation of mixed precision arithmetic be-

comes more challenging. In future studies, We will broaden the application of mixed precision into other regularization methods like the Hybrid regularization.

Bibliography

[1] IEEE standard for binary floating-point arithmetic. *ANSI/IEEE Std 754-1985*, pages 1–20, 1985.

[2] A. Buccini, M. Donatelli, and L. Reichel. Iterated tikhonov regularization with a general penalty term. *Numerical Linear Algebra with Applications*, 24(4):e2089, 2017.

[3] E. Carson and N. J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM Journal on Scientific Computing*, 40(2):A817–A847, 2018.

[4] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, January 1997.

[5] M. Hanke and C. W. Groetsch. Nonstationary iterated tikhonov regularization. *J. Optim. Theory Appl.*, 98(1):37–53, July 1998.

[6] M. Hanke, J. G. Nagy, and R. Plemmons. *Preconditioned iterative regularization for ill—posed problems*, pages 141–164. De Gruyter, 2011.

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

[8] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. Society for Industrial and Applied Mathematics, 1998.

[9] P. C. Hansen. *Regularization Tools Version 4.1.* Technical University of Denmark, 2007.

[10] P. C. Hansen, J. G. Nagy, and Dianne P. O'Leary. *Deblurring Images.* Society for Industrial and Applied Mathematics, 2006.

[11] N. J. Higham. Error analysis for standard and GMRES-based iterative refinement in two and three-precisions. MIMS EPrint 2019.19, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, November 2019.

[12] N. J. Higham. Exploiting low precision arithmetic in the solution of linear systems. ICIAM 2019 Congress, 2019.

[13] N. J. Higham and S. Pranesh. Simulating low precision floating-point arithmetic. *SIAM Journal on Scientific Computing*, 41(5):C585–C602, 2019.

[14] J. G. Nagy, K. Palmer, and L. Perrone. Iterative methods for image deblurring: A matlab object-oriented approach. *Numerical Algorithms*, 36(1):73–93, 2004.

[15] Y. Saad. *Krylov Subspace Methods, Part I*, pages 151–216. Society for Industrial and Applied Mathematics, 2003.

[16] H. Trussell. Convergence criteria for iterative restoration methods. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31(1):129–136, 1983.