

ABSTRACT

XUE, TIANCHENG. Reduced Order Method (ROM) and Anderson Acceleration for Iterative Schemes on Least Square Problems and Optimization, with Applications to Neural Networks and Partial Differential Equations (PDEs) . (Under the direction of Dr. Kazufumi Ito).

In this thesis, we propose an acceleration framework for a class of iterative methods using the Reduced Order Method (ROM). Assuming that the underlying iterative scheme generates a rich basis for the solution space, we construct the next iterate by minimizing the equation error over the linear manifold spanned by this basis. The resulting optimal linear combination yields a more accurate approximation of the solution and significantly enhances convergence. The optimality condition is expressed as a fixed-point equation in a Banach space, and the proposed ROM-accelerated scheme minimizes the sum of residuals by updating the solution via optimal linear combinations of prior iterates. In essence, the method can be seen as a history-based acceleration technique, akin to a delayed or memory-enhanced iterative scheme. This approach effectively remedies semi-ill-posed problems, enabling convergence where standard methods may fail, and also acts as a stabilizing and regularizing mechanism for the original iteration.

Motivated by applications in numerical optimization, variational methods for PDEs, and large-scale AI-driven optimization, we demonstrate the effectiveness of our approach through theoretical analysis and numerical experiments.

Our analysis covers several settings, including variable-step formulations for fixed-point iterations and Newton-like gradient methods. In particular, our approach aligns closely with Anderson acceleration. For linear systems, we interpret Anderson's method the same as ROM. We focus on variable step-size gradient and quasi-Newton methods, as well as fixed-point schemes enhanced by ROM. These combinations exhibit rapid convergence, especially when the condition number of the matrix A is moderate. The ROM-accelerated variable iteration proves to be nearly optimal, further stabilizing and regularizing convergence. Additionally, we extend our method to a randomized, overlapping Kaczmarz-type scheme for large-scale ill-posed linear systems and analyze the

convergence behavior of ROM applied to operator equations. We also investigate Anderson acceleration and its relation to ROM in the context of nonlinear least-squares problems of the form $|F(x)|^2$, formulated in function spaces. Our approach synthesizes with Anderson acceleration, and we interpret Anderson's method as an approximation of nonlinear ROM. In this setting, the solution is approximated as a linear combination of sequential iterates, and the ROM-based Gauss-Newton method is employed to minimize the equation error over the solution manifold. We further explore applications in constrained convex optimization and discuss the role of ROM in improving conditioning.

We conduct numerical experiments in science and engineering that involve mathematical models. We develop new scenario based tools and refine tools based on applications: a control problem and an indefinite PDE. We develop the Anderson map as a continuation to find fixed points for a positive scalar parameterized Matrix Riccati equation. The use of Anderson type map significantly increases the parameter for a fixed point existence of the quadratic equation. We also demonstrate schemes of variable step update with Anderson map as a matrix free method. We access the matrix by evaluating matrix-vector products. We pay attention to specialized saddle point problems, specifically linear steady state Stokes equation like problems. Our plot based on relative residual suggests Anderson-type acceleration drives convergence efficiently. On a parallel side, we also consider the Neural Network digit classification optimization problems. We demonstrate the applicability of Anderson method for Deep Neural Network (DNN) and Convolution Neural Network (CNN). Deep neural network with a fully connected layer connecting each neuron applies a linear transformation to the input vector through a weights matrix, shifted with a bias matrix. Convolution Neural network enriches data architecture with data compression and filtered decomposition. We use backpropagation to optimize necessary parameters to maximize prediction results. We develop Anderson-type acceleration method for the stochastic descent method with batches and improve the network permanence very much.

Reduced Order Method (ROM) and Anderson Acceleration for Iterative Schemes on
Least Square Problems and Optimization, with Applications to Neural Networks and
Partial Differential Equations (PDEs)

by
Tiancheng Xue

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research

Raleigh, North Carolina
2025

APPROVED BY:

Dr. Zhilin Li

Dr. Hien Tran

Dr. Hangjie Ji

Dr. Kazufumi Ito
Chair of Advisory Committee

DEDICATION

To my family and friends

BIOGRAPHY

Tiancheng Xue was born and raised in Shanghai, China. During the process of his growth, he found himself wanted to study more in appreciating the bright side of Mathematics. After finishing high school, he decided to come to study in the USA to continue with his education beyond K-12 education in Shanghai. He has completed a Bachelor of Math at Rutgers University, New Brunswick in 2018. Then he became interested in mathematical modeling in business applications and geared to earn a Masters degree of Operations Research and Supply Chain Management in 2020. During his time in Master program, he found his interests in inventing and improving algorithms related to mathematical optimization in a general context, which led him to the journey of Ph.D. program of Operations Research at NC State University, while he also earned an en-route Masters in Mathematics. He also included a minor of Math, in addition to his Ph.D. degree. He completes his research dissertation under the direction of Dr Kazufumi Ito.

ACKNOWLEDGEMENTS

First off, I would like to thank the Graduate School at NC State University for giving me an offer to study in the Graduate Program of Operations Research (OR), the first interdisciplinary program established at NC State University, where students have the freedom to choose to work with faculties from multiple departments than only one department. I would like to specifically thank our program director, Dr Michael Kay and Dr Maria Mayorga, our program assistant Mrs Linda Smith, and OR Graduate Student Association (GSA) in making sure that all OR Ph.D. students can graduate with a Ph.D. degree smoothly and on time. Thank you for your time in organizing seminars, coffee breaks, and end of year celebration event. I will miss those OR activities after my graduation. Thank you for letting me know potential research and career opportunities. I would not have proceeded onto this step without the support from our program.

Afterwards, I would like to thank my research supervisor, Dr Kazufumi Ito, for his constant help, guidance, and discussion about how to do and think about research in a variety of angles with his vast range of knowledge, and finally confirm that our research results stand out. First, his diversity of interests in all aspects of subjects, specifically mathematics and science, stand out to set up an achievable research problem for a Ph.D. dissertation research. Second, his expressions in expressing one concept repeatedly with multiple perspectives give directions of possible ways to think about a problem and makes arguments from paper more complete. Third, he thinks highly in terms of level of abstraction to find the core of complicated concepts with simplest words to confirm the highlight of research progress and direct potential future work. To conclude, the 3 step method of “do think confirm” reshapes my style of staying on track with any projects that I have to complete.

I would also like to thank him effort of treating research meeting as a class to push me to make progress during normal semesters while he stays in Raleigh. Even with times when his status quo reached lowest, he still was able to manage to meet with me and gave me your ideas to try. Every meeting with him gives me more momentum to making progress and completing my study. The journey is full of trial-and-error: one never knows what will work until having it tried and confirmed. He focuses on giving general ideas

rather than specific details that always stays theme consistent. During his time in Hong Kong for 70th birthday celebration, he still kept track with me and made sure that I were able to graduate on-time. His persistence and attitude towards doing research really builds a model for all people, regardless of age.

In addition to research guidance, I would also like to thank Dr Ito in giving me a comfortable environment to study, grow, and socialize with as a future researcher. I specifically thank him for introducing me to the Nonlinear analysis student seminar groups to have a community that students can discuss research progress freely and know each other, with no conflict of interests sharing. I would like to thank Adam Pickarski, Diego Cornejo, Javier Madariaga, Matthew Broussard, Guang Yang, and among others for sharing your own research topics and having patience in listening and providing valuable feedback about my research. I also appreciate him introducing me his past collaborators and students that I can reach out to, if I need help, from all over the world.

Choosing a research supervisor is not an easy task, therefore I specifically thank the discussion with various faculty members in guiding me to choose Professor Ito as a research supervisor, specially Dr Kazufumi Ito, Dr Zhilin Li, Dr Ryan Murray, Dr Hangjie Ji, and Dr Maria Mayorga. I want to thank Dr Ito in initiating conversation of research interests when I was serving as his teaching assistant for an undergraduate course. I appreciate Drs Li and Murray for their time in sharing about opinions of working with OR affiliated Math professors. I want to give special thank to Dr Ji in constantly talking with me about the importance of research progress and in her course sequence of numerical analysis, with incorporation at the end with an introduction to deep learning and numerical optimization. I also would like to thank Dr Dávid Papp in giving a reading course and then a course in the field of numerical optimization, which is the main field of this thesis as to solve an optimization problem with numerical methods. I appreciate Dr Mayorga in having a heart to listen and offer advice for choosing and getting along with a research supervisor. I am very grateful for the courses that makes me choose to work with a professor in the Mathematics department than from other departments. For example, control theory sequence and an introduction to PDE from Dr Murray, as well as Differential geometry sequence from Dr Peter McGrath. Their attitudes in helping non-math major students (like me) drive me to collaborate with Mathematics again, complementary to my undergraduate Math education. Moreover, I thank my fellow OR

schoolmates who worked with math professors, especially Zongxin Tian and Guoqing Zhang, in having an OR centered student research community in the Math department. I also express my gratitude to other OR students working with faculties from other department for our meeting and discussion at the end of every semester for life, well being, career development, and experience and suggestions towards working with OR affiliated faculty members, especially Kuangying Li, Yen-Hsi Chou, Jiewen Sheng, Rui Shi, Rui Wang, Jiaying Wang, and Lesheng Wang.

I would like to thank my committee members: Dr Kazufumi Ito, Dr Hien Tran, Dr Hangjie Ji, and Dr Zhilin Li for your time for being my committee members to give me your valuable feedback for my research exams and increase the potential values for this particular research project and my future career development. I would like to thank Dr Rahman Khorammar in offering me his Ph.D. thesis template when he studied at NC State. My special thank also gives to my friend and neighbor Jianghao Shen in not making me afraid of transferring my written thesis into a thesis with NC State template, and to my friend and neighbor Yafan Zhang in giving me his keyboard for free to improve my experiences to type this thesis. I am also grateful for Math department's friendly faculty and staffs that allows me to print my thesis for making revision on paper, that arranges a room for me to make a defense. I show gratitude to the administration in Math department of building a supportive community from Dr Alina Chertock, Dr Hien Tran, Dr Lorena Bociu, Dr Seth Sullivant, Dr Bojko Bakalov, and Ms Le To. After the end of my thesis defense, I am happy to see Dr Ralph Smith and Dr Mansoor Haider in greeting with me and reminding me to turn off lights and return keys to the department. I give gratitude to Dr Alen Alexandrian in always keeping in touch with me on lounge area in giving me advice to proceed.

I would like to thank my family and friends for your support. First, I want to bring particular appreciation to my office mates Arouni Hashim, Diego Castedo Pena, Elijah Rutter, and Regan Richardson. Thank you for always staying around with me in office. I would also like to thank other fellow Math grad students as friends in SAS Hall to interact with, e.g. Ian Livengood, Devon Olds, Steven Maio, Rachel Morris, Evangelia Ftaka, Nigar Sultana, Ziyang Jiang, and Fangrong Zhan. I am grateful for local restaurants in Raleigh areas, especially FunDippot, for letting me enjoy food after my research meeting. I also express my warmest thank to those whom wished me best of luck in thesis defense:

Yanze Chen, Shengjia Wang, Yen-Hsi Chou, Zhuyang Lin, Junheng Zhou, Kang Fu, Zixuan Huang, Yuhan Hu, Kuangying Li, Pingyao Feng, Guang Yang, Xiaohan Lin, Zezhao Li, Bingyi Su, Manting Jin, and many others. I add thanks to friends who helps me to adapt life in USA, including Yingchu Ni, Yibin Dong, and many others. Special thanks at the end are given to my parents, Hongjun Xue, Huiqun Zhu, and grandparents, Yuemei Xue and Genxiong Guo, Xiuying Yang, and my late grandpa Chengwu Zhu.

Lastly, I would like to thank my previous teachers and other family members. I show my appreciation to my teachers from high school Junmei Zhang, Lihua Feng, Lei Xu, Lei Du, Liping Wei, for your support in encouraging me study in the USA for my undergraduate years. I thank teachers from my masters' program, especially Dr Kamlesh Marthur for giving me a first hand research experiences and Dr Daniel Solow for letting me fear not in coding. and other family members from Shanghai in letting me believe that I could be a potential Ph.D. some time in the future.

TABLE OF CONTENTS

List of Tables	xi
List of Figures	xii
Chapter 1 Introduction	1
1.1 Problem Formulation	1
1.1.1 Least Square Problems	2
1.1.2 Fixed Point Problem: $F(x) = 0$	4
1.2 Objective and Outline	4
1.2.1 Research Background	5
1.2.2 Reduced Order Method and Anderson Type Acceleration	6
1.2.3 Variable Step Update	7
1.2.4 Convex Optimization	8
1.2.5 Convergence Analysis	9
1.2.6 Applications in Scientific Computing	9
1.2.7 Applications in Neural Network	10
1.2.8 Conclusion and Future Work	11
Chapter 2 Research Background	12
2.1 Problem of Interest	12
2.2 Regularization	14
2.3 Iterative Basis Generation	15
2.3.1 Random Search	15
2.3.2 Krylov Subspace Method	16
2.3.3 Gauss Newton's Method	17
2.3.4 Quasi-Newton Method	17
2.3.5 Gradient Descent and its Variants	19
2.3.6 Nonlinear Kaczmarz Method	21
2.4 Direct Iterative Inversion in the Iterative Subspace (DIIS)	22
Chapter 3 Reduced Order Method	24
3.1 ROM and Anderson Type Acceleration Comparison	25
3.2 Anderson and ROM Acceleration as Evolutional Fixed point	28
3.3 Variants of ROM	30
3.3.1 Nested ROM	30
3.3.2 Sampled ROM	31
3.3.3 Residual Based ROM	32
3.3.4 ROM Extrapolation	32
3.3.5 ROM with Preconditioning	33

3.3.6	Back-projection Method (Dual) A^*b	34
3.4	Applications	35
Chapter 4	Variable Step Update	37
4.1	Variable Step Size Update	37
4.1.1	Cauchy’s Method	39
4.1.2	Variants to Cauchy Methods	41
4.1.3	Along with ROM	43
4.2	Multi-Step Subspace Update	44
4.3	Applications	46
4.3.1	Regularization	46
4.3.2	Saddle Point Problem	47
Chapter 5	Convex Optimization	49
5.1	Conjugate Gradient Method	49
5.2	Conjugate Residual Method	52
5.3	Proximal Gradient Algorithm	53
Chapter 6	Convergence	55
6.1	Linear Function	55
6.2	Nonlinear Function	56
6.3	Sequential Method	57
6.4	Strong Convergence under Coercivity	58
Chapter 7	Applications in Scientific Computing	60
7.1	Nonlinear System for Matrix Riccati Equation	60
7.2	Saddle Point Problems	65
7.2.1	Saddle Point Problem 1	66
7.2.2	Saddle Point Problem 2	69
Chapter 8	Applications in Neural Network	72
8.1	Algorithms of Interest	77
8.1.1	Mini-batch Gradient Descent Method	77
8.1.2	Stochastic Gradient Descent (SGD) Method	78
8.1.3	Backpropagation Algorithm	79
8.1.4	Anderson Acceleration	81
8.2	A Fully Connected Neural Network	82
8.2.1	Algorithm Description	82
8.2.2	Algorithm Implementation	83
8.2.3	Results	84
8.3	Convolution Neural Network (CNN)	85
8.3.1	Algorithm Description	87

8.3.2	Algorithm Implementation	88
8.3.3	Results	89
Chapter 9	Conclusion and Future Work	91
9.1	Conclusion	91
9.2	Possible Future Work	92
References	93

LIST OF TABLES

Table 7.1	Contraction continuation with standard Banach Iterate with 3 steps ($n = 10$).	63
Table 7.2	Contraction continuation with Nested 4 step Anderson acceleration ($n = 10$).	64
Table 7.3	Contraction continuation with Nested 7 step Anderson acceleration ($n = 10$).	64
Table 7.4	Contraction continuation with Nested 7 step Anderson acceleration ($n = 200$).	65
Table 8.1	DNN Self Test: Accuracy Comparison.	84
Table 8.2	CNN Self Test: Accuracy Comparison.	89
Table 8.3	CNN Test: Accuracy Comparison.	89

LIST OF FIGURES

Figure 7.1	Saddle Point Problem 1 (One direction update: $r = b - Ax$): comparison of standard case, accelerated case, and sampled case. . .	68
Figure 7.2	Saddle Point Problem 1 (Two directions' update: $r = b - Ax, Ar$): comparison of standard case, accelerated case, and sampled case. .	68
Figure 7.3	Saddle Point Problem 2 (One direction update $r = b - Ax$): comparison of standard case, accelerated case, and sampled case. . . .	71
Figure 7.4	Saddle Point Problem 2 (Two directions' update $r = b - Ax, \bar{A}r$): comparison of standard case, accelerated case, and sampled case. .	71

Notation

Number Line:

- \mathbf{R} : Real number;
- \mathbf{R}_+ : Positive real number;
- \mathbf{Z} : Integers;
- \mathbf{Z}_+ : Positive integers;
- $\mathbf{Z}_{\geq m}$: Integers with value greater than or equal to $m \in \mathbf{Z}$.

Vector Space:

- \mathbf{R}^n : Euclidean space of dimension n ;

Inner Product Space:

- (u, v) : Inner product of $u \in V$ and $v \in V$ in a vector space V , unless specified;
- $|\cdot|$: Norm of a given vector, with 2-norm.

Function Space:

- F : Equation of interest;
- f : Functional for minimization;
- α : Step size and scalar;
- $\{a_n\}$: Sequence of real numbers;
- Ψ : Regularization function for an ill-posed problem;

- ϕ : Activation function for a Neural Network.

Matrix Space:

- $\mathbf{R}^{m \times n}$: Real matrices with size $m \times n$;
- A^* : Conjugate transpose of matrix A ;
- A^\top : Transpose of matrix A .
- \odot : Hadamard product between two matrices of same size.
- \otimes : Kronecker product between two matrices of arbitrary size.

Chapter 1

Introduction

We introduce an acceleration scheme based on Reduced Order Methods (ROM). We accelerate a given iterative scheme, a procedure to generate a sequence of improving approximate solutions for a nonlinear least square problems of minimizing $|F(x)|^2$, where $F(x)$ is an equation error in a Banach space X , and for an optimization problem of minimizing $f(x)$, where f is a merit function. Assume that the iterative scheme generates a rich basis for the solution, we accelerate it by Reduced Order Method (ROM). We express the next iterate based on the reduced order method, i.e. minimizing equation error over the linear manifold spanned by the basis, and the problem is reduced to determine the optimal weights. The optimal weights give the optimal prediction of solution and accelerate the convergence. The ROM acceleration is very similar to the Anderson type acceleration with Direct Inversion in the Iterative Subspace (DIIS). We discuss DIIS in Section 2.4, and we compare ROM and Anderson type acceleration in 3.1.

1.1 Problem Formulation

We formulate two classes of problems: least squares problems and merit function based optimization problems. In terms of least squares problems, we formulate a regularized optimal transport problem and its special cases. For example, we provide a Neural Network formulation based optimal transport problem. We formulate fixed point problem as

a special case of least square problem to motivate a merit function based optimization problem.

1.1.1 Least Square Problems

We formulate a regularized least square problem. Let $(X, |\cdot|_X)$ be a Banach space, let $(Y, |\cdot|_Y)$ be a Banach space, and let $(Z, |\cdot|_Z)$ be a Banach space. Given a map $F : X \rightarrow Y$ as a residual function, $\Psi : X \rightarrow \mathbf{R}_+$ as a regularization function, and $\gamma \geq 0$ as its regularization constant, a regularized least square problem for F with regularization Ψ is given as:

$$\text{minimize} \quad |F(x)|_Y^2 + \gamma \Psi(x) \quad \text{over} \quad x \in X. \quad (1.1.1)$$

We solve by variable step update. Given $x_n \in X$, the variable step update at x_n in the direction $p_n \in X$ with step size $\alpha_n \in \mathbf{R}$ is the iterate with the form:

$$x_{n+1} = x_n + \alpha_n p_n, \quad (1.1.2)$$

where α_n is the step-size at iteration n . For example, for residual function $F : x \in X \rightarrow F(x) \in Y$, assuming $\gamma = 0$, we consider $p_n = F(x_n)$; α_n can be a fixed constant α . More details of variable step update will be involved in Chapter 4.

Notice that, formulation (1.1.1) with iterative scheme (1.1.2) has encompassed many applications. In order to illustrate this, we model an example from optimal transport, with Deep Neural Network and Conventional Network optimization from application side.

Example 1.1.1. Optimal Transport Problem Given a transport map $T_\theta : Y \rightarrow Z$, $(g_0, g_1) \in Y \times Z$, Θ as the set of internal variables for T_θ with $\theta \in \Theta$, and $\Psi : \Theta \rightarrow \mathbf{R}_+$ as the transport cost map. We minimize the transport cost. One can formulate a regularized residual problem for the transport problem as:

$$\text{minimize} \quad |T_\theta(g_0) - g_1|_Z^2 + \Psi(\theta), \quad (1.1.3)$$

over $\theta \in \Theta$, where T_θ is the transport map, and Ψ is the transport cost map. The map T_θ can represent the tomography map, e.g. CT, EIT, forward scattering, with θ

as corresponding medium. Other formulations of transport involve Monge-Ampere mass transport Benamou and Brenier (2000) , the value transport, and mass transport from PDE-based models.

1. (Neural Network Formulation) Let $L \in \mathbf{Z}_{\geq 2}$. For $l \in \{1, \dots, L\}$, given $b_l \in \mathbf{R}^{n_l}$ and variable step size $\tau_l \in \mathbf{R}$, one can transport $X_0 \in \mathbf{R}^{n_1}$ to $Y \in \mathbf{R}^{n_L}$ by the deep learning network:

$$X_l = \tau_l \phi_l(W_l X_{l-1} + b_l), X_0 = A, Y = H(X_L),$$

where $\theta_l = (W_l, b_l) \in (\mathbf{R}^{n_l \times n_{l-1}}, \mathbf{R}^{n_l}) := \Theta_l$ are unknown internal variables, $\phi_l : \mathbf{R}^{n_{l-1}} \rightarrow \mathbf{R}^{n_l}$ is an activation function for (W_l, b_l) that calculates the output of the node based on its individual inputs and their weights, element-wisely applied to a given vector in \mathbf{R}^{n_l} , and H is a classification map that connects input and output by discrete dynamics. For example, ϕ_l can be a RELU function, namely

$$\phi_l(x) = \max(x, 0).$$

For the last step, H can be a softmax function

$$H(z) = \frac{1}{\sum_{j=1}^{n_L} e^{z_j}} (e^{z_1}, \dots, e^{z_{n_L}}).$$

Let $\theta = (\theta_1, \dots, \theta_L) \in (\Theta_1 \times \dots \times \Theta_L) := \Theta$ be internal variables, $X = (X_1, \dots, X_N)$ be the predicted outcome, $T = (T_1, \dots, T_N)$ be the actual value for the prediction, and a loss function be $\mathcal{L} : (\theta, X, T) \rightarrow \mathbf{R}$. One formulates the transport problem as

$$\text{minimize} \quad |\mathcal{L}(\theta, X, T)|^2 + \Psi(\Theta). \tag{1.1.4}$$

For example, we can formulate the loss function as mean squared error with N as total sample amount

$$\mathcal{L}(\theta, X, T) = \frac{1}{2N} \sum_{i=1}^N (X_i(\theta_1, \dots, \theta_L) - T_i)^2.$$

2. (Network Transformation) Let $G = (V, E)$ be a directed acyclic graph, with V as the

vertex set and E as the edge set for the graph G . Let $i \neq j$ so that $(x_i, x_j) \in V \times V$, and the matrix $L = (L_{ij})$ be the transition matrix given by

$$L_{i,j} = \begin{cases} 1 & (i < j) \text{ and } (x_i, x_j) \in E \\ -1 & (i > j) \text{ and } (x_i, x_j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Now, with D a symmetric positive definite matrix with internal variable(s), set

$$A = L^\top DL.$$

For example, one can let $T_\theta = A$, $g_1 = 0$, and $\gamma = 0$ satisfy the form of (1.1.3).

1.1.2 Fixed Point Problem: $F(x) = 0$

We formulate merit function minimization as an extension of least square problem for fixed point of another map. Suppose we want to find a fixed point for $G : X \rightarrow X$. We can consider minimizing $|x - G(x)|^2$ over $x \in X$. Let $F(x) = x - G(x)$, then we have minimizing $|F(x)|^2$. With $f'(x) = F(x)$, we have defined an equivalent problem for finding a fixed point for G : $\min f(x)$. With $J \approx f''(x_n)$ by BFGS, J^{-1} computed by the Sherman-Morrison-Woodbury formula, and f' upgraded by Broyden method, we have:

$$x_{n+1} - x_n = \alpha J^{-1} f'(x_n).$$

We discuss Broyden update and BFGS update in Section 2.3.4.

In the next section, we bring highlights for each chapter of the thesis.

1.2 Objective and Outline

The thesis first aims to introduce various iterative methods to solve two certain classes of optimization problems — least square problems, and merit function based optimiza-

tion problems. Then, we aim to propose ROM with basis elements from a given iterative scheme to accelerate algorithms by considering variable step size, Anderson-type acceleration, and Direct Inversion in the Iterative Subspace (DIIS) with a reduced order basis generated from a given iterative scheme. We include numerical experiments to verify that our method works well, with examples motivated from numerical optimization, variational methods for Partial Differential Equations (PDEs), and large scale optimization problems from machine learning. In terms of convergence analysis, we will start from Euclidean space and then generalize them in function space context.

1.2.1 Research Background

In Chapter 2, we will introduce how Anderson type acceleration will be performed. We first make sure our problem a well-posed one. Then, we generate basis by generating solution iteratively. We look at random search, Krylov subspace method, Gauss-Newton, Damped Newton, gradient descent with its variants, and Kaczmarz method. In terms of acceleration algorithms, we pay special attention to Anderson acceleration, which brings us motives to study Reduced Order Method (ROM) in Chapter 3.

We update our result from ROM by Direct Inversion in the Iterative Subspace (DIIS) Pulay (1980) as a linear combination from previous iterations with minimized total residual. Consider $F : X \rightarrow X$ with the goal of minimizing $|F(x)|^2$ over $x \in X$. Given basis vectors $(x_1, \dots, x_m) \in X \times \dots \times X$, with $e_i = F(x_i)$ for $i \in \{1, \dots, m\}$, we have:

$$\begin{aligned} \text{minimize} \quad & \left| \sum_{i=1}^m c_i e_i \right|^2 \\ \text{subject to} \quad & \sum_i c_i = 1. \end{aligned} \tag{1.2.1}$$

With $B_{ij} = (e_i, e_j)$, one formulate the Lagrangian associated to (1.2.1) as:

$$L(c, \lambda) = c^\top Bc - 2\lambda \left(\sum_i c_i - 1 \right).$$

First we take partial derivatives of L with respect to c and λ . Then, we equate the result to 0. Finally, we move the minus sign to λ . Without loss of generality, we can treat $-\lambda$

as λ to get our DIIS system:

$$\begin{bmatrix} B_{11} & \cdots & B_{1m} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ B_{m1} & \cdots & B_{mm} & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \quad (1.2.2)$$

We update x_{m+1} with the linear combination of x_1, \dots, x_m from (1.2.1) with coefficients c_1, \dots, c_m from the solution of (1.2.2) by

$$x_{m+1} = \sum_{i=1}^m c_i x_i. \quad (1.2.3)$$

1.2.2 Reduced Order Method and Anderson Type Acceleration

In Chapter 3, we introduce Reduced Order Method (ROM), a method to express new solution generated by a sequence of monotonically decreasing solution basis. The method is similar to Anderson method Walker and Ni (2011): We compare ROM with Anderson type acceleration, both conceptually and numerically. We now give our definition of ROM.

Definition 1.2.1 (Reduced Order Method). *Let $F : X \rightarrow X$, and $\Psi : X \rightarrow \mathbf{R}_+$. Given a sequence $\{x_k\} \in X$, ROM is to solve the following optimization problem:*

$$\text{minimize} \quad |F(\sum_{k=1}^m \alpha_k x_k)|^2 + \Psi(\sum_{k=1}^m \alpha_k x_k), \quad (1.2.4)$$

over $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbf{R}^m$, and set

$$x_{m+1} = \sum_{k \leq m} \alpha_k^* x_k, \quad (1.2.5)$$

where α^* is the optimizer for (1.2.4).

Similar to ROM, Anderson-type acceleration, especially useful in accelerating fixed point iterates Walker and Ni (2011), generates solution from past iterate by considering total error minimization.

Definition 1.2.2 (Anderson Method for a Fixed Point $x = G(x)$). *Anderson method, for the fixed point $x = G(x)$, is to minimize the total sum of residuals, i.e. given $k \in \mathbf{Z}_{\geq 1}$, and let $G : X \rightarrow X$ be a map with fixed point, for each $k \in \{1, \dots, n\}$, let $x_k \in X$ be the solution iterate of k -th iteration, let the residual $F : X \rightarrow X$ defined by $F(x) = x - G(x)$, and let $r_k = F(x_k)$, then we solve*

$$\begin{aligned} & \text{minimize} && \left| \sum_{k \leq n} \alpha_k r_k \right|^2 \\ & \text{subject to} && \sum_{k \leq n} \alpha_k = 1, \end{aligned} \tag{1.2.6}$$

over $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbf{R}^n$, and set

$$x_{n+1} = \sum_{k \leq n} \alpha_k^* x_k, \tag{1.2.7}$$

where α^* is the optimizer for (1.2.6).

We compare ROM and Anderson:

- ROM is unconstrained optimization problem on α , while Anderson type acceleration is a constrained optimization problem on α .
- Constrained ROM with linear function is the same as Anderson type acceleration.
- The update step of ROM only relies on x , while Anderson relies on $G(x)$.

We also argue that they are related in Section 3.1. Anderson type method provides good approximation for ROM optimization. Anderson solution can be used as a ROM solution, if it provides enough descent.

1.2.3 Variable Step Update

We develop variable step update method in Chapter 4, even for the fixed point iterate. It accelerates the convergence with monotonically decreasing residual norm. ROM will be

more effective. For each iterate, we determine step size to make ROM work. For example, Cauchy step defined by (1.2.9) may make things work.

More specifically, we develop schemes for a residual error function $F : X \rightarrow X$. We minimize $|F(x)|^2$ over $x \in X$.

Definition 1.2.3 (Variable Residual Error Function Update). *Given $F : X \rightarrow X$ and $x_0 \in X$, the variable residual error function update is an iterate with the form:*

$$x_{n+1} = x_n + \alpha_n F(x_n), \tag{1.2.8}$$

where α_n is the variable step size.

We now introduce Cauchy step as a variable step size update.

Definition 1.2.4 (Cauchy Step). *Let $F : X \rightarrow X$. The Cauchy step α_n at $x_n \in X$ along direction $p_n \in X$ is defined as*

$$\alpha_n^* = \min |F(x_n + \alpha p_n)|, \tag{1.2.9}$$

over $\alpha \in \mathbf{R}$.

We study variable step update in the ROM context. We determine Cauchy one-step subspace update, and Cauchy multi-step subspace update, and their variants. We see the residual monotonically decreases.

1.2.4 Convex Optimization

In Chapter 5, we study convex optimization and ROM acceleration. We see monotone convergence with enough descent. We motivate our study by conjugate gradient method on quadratic programming without constraints, and then apply it on a convex optimization problem constrained over a convex set.

For the ROM step, given X as a nonempty convex set, we find α to minimize a convex

function $f : X \rightarrow \mathbf{R}$, for a linear combination of a given basis:

$$\text{minimize} \quad f\left(\sum_k \alpha_k x_k\right). \quad (1.2.10)$$

Or, given a nonempty convex subset \mathcal{C} of X , we consider:

$$\begin{aligned} &\text{minimize} \quad \sum_k \alpha_k f(x_k) \\ &\text{subject to} \quad \sum_k \alpha_k x_k \in \mathcal{C}, \end{aligned} \quad (1.2.11)$$

and let

$$x_{m+1} = \sum_{k \leq m} \alpha_k^* x_k.$$

As a consequence, we also see nonlinear conjugate residual method with ROM, and proximal gradient algorithm with ROM will give faster convergence.

1.2.5 Convergence Analysis

In Chapter 6, we develop convergence analysis to accelerate fixed point problems. We consider linear function and nonlinear function, and we generalize the analyses to function space and mini-batch descent.

Theorem ($\min |F(x)|^2$) The sequence $\{x_k\}$ with $\{r_k\} = \{F(x_k)\}$, assuming generated by ROM update (1.2.5) converges weakly to the pair (x^*, r^*) . Under the coercivity, if $|r^*| = 0$, then $x_k \rightarrow x^*$, the unique fixed point.

1.2.6 Applications in Scientific Computing

We conduct numerical experiments concerning models from science and engineering in Chapter 7. We consider solving the steady state linear Stokes equations. Applying ROM and its variants will make convergence happen a lot earlier than using the standard case update. Motivated from Navier Stokes equation, we determine the maximal coefficients for a Ricatti equation to be stable. The use of Anderson acceleration as a continuation

method increases b so that upgrading differences will be small.

We formulate a Riccati equation. Given Q a symmetric positive definite $n \times n$ matrix, and a small $b > 0$, we consider the equation

$$\dot{u} = -b u u + Q. \quad (1.2.12)$$

Given $u_0 = 0$ as a $n \times n$ matrix, we progressively maximize b and apply iterate

$$u_{n+1} = -b u_n u_n + Q,$$

until $|u_{n+1} - u_n|$ is unable to meet below tolerance level.

We consider a saddle point system given by a specified (D, E) , with D as a symmetric positive definite matrix:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(x, Dx) - (f, x) \\ \text{subject to} \quad & Ex = g. \end{aligned} \quad (1.2.13)$$

We consider a divergence free case, i.e. the case when $g = 0$. With simplification on necessary optimality conditions, we obtain $Ax = b$:

$$\underbrace{\begin{bmatrix} D & E^\top \\ E & \mathbf{0} \end{bmatrix}}_A \underbrace{\begin{bmatrix} u \\ p \end{bmatrix}}_x = \underbrace{\begin{bmatrix} f \\ 0 \end{bmatrix}}_b. \quad (1.2.14)$$

1.2.7 Applications in Neural Network

In Chapter 8, we consider Neural Network (NN) optimization problems. We develop Anderson-type acceleration method for the stochastic decent method and improves the network permanence very much. For a given database with a training and a testing set, we focus on incorporating ROM into optimization schemes (say stochastic gradient descent) to maximize prediction accuracy. The use of Anderson acceleration gives early stop of making prediction accuracy of 99% than the standard case. We consider it as a special case of Transport problem, with a given pair of input and output. We demonstrate the applicability of the method for Deep Neural Network (DNN) and Convolution Neural

Network (CNN). DNN is a map with input into a fully connected part to output prediction, with no convolution operations involved. CNN is a map with input by convolution with filters into a fully connected part to output prediction.

We now discuss key steps in NN optimization:

- Backward propagation Rumelhart et al. (1986) computes its parameter updates.
- Universal Approximation Theorem Hornik et al. (1989): one obtains significantly better results by training a neural network with given internal variables. that concerns the convergence for a sequence of neural networks to the actual solution.
- Vectorization for target values for least square objective: it converts categorical variables into a numerical format suitable for use in machine learning algorithms that require numerical input;
- Mini-batch gradient descent: we update the network by sequentially optimizing the internal variables, e.g. weight parameters and bias parameters. We can either keep the sample ordering the same, or shuffle the entire sample before training.

For simplicity, we consider using the original MNIST database LeCun and Cortes (2005) and its subsets for training and testing sets.

1.2.8 Conclusion and Future Work

We have developed algorithms and analysis on ROM. In addition, we have tested our algorithms through applications. As for future work, we discuss model selection problems. We intend to develop applications of general class of saddle point problems, and the neural network design for computer tomography and inverse medium problems.

Chapter 2

Research Background

We introduce necessary background materials regarding to our paper development. We first look at problem of interest and its characterization. Then, we introduce concerned algorithms, including iterative methods, optimization algorithms, and known accelerated algorithms. When describing algorithms, for simplicity purposes, we use numerical matrix, which can be generalized to operator settings.

2.1 Problem of Interest

We are interested in solving least square problem: let $(X, |\cdot|_X)$ and $(Y, |\cdot|_Y)$ be two given Banach spaces, a function $F : X \rightarrow Y$, and a nonempty closed $C \subset X$, we solve

$$\begin{aligned} & \text{minimize} && |F(x)|_Y \\ & \text{subject to} && x \in C, \end{aligned} \tag{2.1.1}$$

over $x \in X$.

Example 2.1.1. A Well-posed Problem and an Ill-posed Problem The examples below show a pair of a well-posed and an ill-posed problem.

1. Let $X = \mathbf{R}$, and let $I : X \rightarrow \mathbf{R}$ be defined by: $I(x) = x^2 - 2$. Solve:

$$\text{minimize } |I(x)|, \tag{2.1.2}$$

over $x \in X$. Then $\sqrt{2} = \operatorname{argmin} I(x)$, since $\sqrt{2} \in \mathbf{R}$. Thus, the problem (2.1.2) is a well-posed problem, when $X = \mathbf{R}$.

2. Let $X = \mathbf{Q}$. We want to solve

$$\text{minimize } |I(x)|, \tag{2.1.3}$$

over $x \in X$. Then, no solution exists for (2.1.2) since $\sqrt{2} \in \mathbf{R} \setminus \mathbf{Q}$. So, the problem (2.1.2) is ill-posed, when $X = \mathbf{Q}$. But one can pick $\epsilon > 0$ (say 0.002225) for regularization so that there exists $1.415 \in \mathbf{Q}$ so that $1.415^2 = 2 + \epsilon$ so that the problem become a well-posed one.

Example 2.1.2. Norm Approximation Boyd and Vandenberghe (2004) Let $A : X \rightarrow Y$, and $b \in Y$ be given. One needs to find $x \in X$ so that $|b - Ax| \approx 0$. Therefore, one has the following formulation:

$$\text{minimize } |Ax - b|, \tag{2.1.4}$$

over $x \in X$. Interpretations from applications: Suppose x^* is the desired result, then:

- Fitting: Find x^* so that Ax^* is close to b as possible.
- Estimation: x^* gives the most plausible x when $y = b$ for $y = A(x)$.

Throughout the thesis, we assume that our problem of interest a well-posed one, i.e. the given problem exists a unique stablized solution.

2.2 Regularization

We extend the discussion of (2.1.3) in Banach spaces. Let $(X, |\cdot|_X)$ and $(Y, |\cdot|_Y)$ be Banach spaces, and let $A : X \rightarrow Y$ be an operator, with regularization Ψ . One can formulate a Tikhonov functional as

$$J_\alpha(x) = |A(x) - b|^2 + \alpha\Psi(x). \quad (2.2.1)$$

Example 2.2.1. Optimization with Regularization Given an operator $A : X \rightarrow Y$, a vector $b \in Y$, and a scalar $\beta > 0$, we have 2 commonly used regularization method:

1. **Non-negativity regularization:**

$$\Psi(x) = |(-x)_+|^2.$$

2. **Sparsity/LASSO (L^1) regularization:**

$$\Psi(x) = |x|_1^2.$$

3. **Tikhonov (L^2) regularization:**

$$\Psi(x) = |Px|_2^2. \quad (2.2.2)$$

e.g. P can be a derivative map.

4. **Total Variation (TV) regularization:**

$$\Psi(x) = |x|_{\text{TV}(\Omega)}.$$

Ito and Jin (2014) gives a description about other possible candidates for the function Ψ : With Ω as an open and bounded domain of X , we may have $\Psi : X \rightarrow \mathbf{R}_+$ as $\Psi(u) = |u|_{\text{BV}(\Omega)}$, or $\Psi(u) = |u|_{\text{TV}(\Omega)}$.

2.3 Iterative Basis Generation

This section introduces iterative schemes to generate basis for ROM.

2.3.1 Random Search

We introduce Random Search, a derivative free optimization algorithm. Let $f : \mathbf{R}^n \rightarrow \mathbf{R}$ be defined everywhere.

Algorithm 1 Random Search

Require: f , x_0 , stopping criteria, r

Ensure: $z \in \mathbf{R}^n$ met stopping criteria

- 1: $x \leftarrow x_0$.
 - 2: $S \leftarrow B_{x_0}(r)$
 - 3: **while** Stopping Criteria not met **do**
 - 4: Sample new points $y \in S$.
 - 5: **if** $f(y) < f(x)$ **then**
 - 6: $x \leftarrow y$.
 - 7: **end if**
 - 8: **end while**
-

One can treat Algorithm 1 as (1.1.2). Specifically, we note that for each $y \in S$, there exists $p \in S$ and $\alpha \in [-r, r]$ such that $y = x_0 + \alpha p$.

Statistically speaking, random search is optimal, i.e. looking for gold on a beach. That is why we formulate, in ROM, since it generates a rich family of basis. For example, picking random m points in a given range will give basis $\{x_1, \dots, x_m\}$.

2.3.2 Krylov Subspace Method

First, we develop Newton-Krylov method Kelley (2003). Newton-Krylov methods are numerical methods for solving non-linear problems using Krylov subspace linear solvers. First, we generate Krylov subspace. Let d_0 be given, we generate the directions (basis) $\{d_k\}$ by t

$$D_t F(d_k) = d_{k+1},$$

where $D_t F$ is the difference approximation of $F'(x)$:

$$D_t F(d) = \frac{F(x + td) - F(x)}{t}, \quad t > 0.$$

Thus, we have

$$F(x + \sum_k \alpha_k d_k) - F(x) \sim F'(x)(\sum_k \alpha_k d_k) \sim \sum_k \alpha_k D_t F(d_k),$$

and we consider

$$\text{minimize} \quad \left| \sum_{k \leq n} \alpha_k d_{k+1} + F(x) \right|^2, \quad (2.3.1)$$

over $\alpha \in \mathbf{R}^n$. Notice (2.3.1) has an approximated form as in the form of ROM:

$$\text{minimize} \quad \left| F(x + \sum_k \alpha_k d_k) \right|^2, \quad (2.3.2)$$

over $\alpha \in \mathbf{R}^n$, which suggests we have developed ROM for Newton-Krylov.

Next, we introduce Arnoldi iteration Arnoldi (1951) as a method to minimize $|F(x)|^2$, with $F(x) = Ax - b$. We generate basis for the m -th Krylov subspace Nocedal and Wright (2006) as

$$K_m(A, r_0) = \text{Span}\{r_0, Ar_0, \dots, A^{m-1}r_0\}, \quad (2.3.3)$$

then orthogonalize the space (2.3.3) by Gram-Schmidt.

We introduce GMRES Saad and Schultz (1986), also known as the Generalized Minimum RESidue, as a Krylov subspace method. GMRES finds an approximated solution x_n for a linear system $Ax = b$ with $A \in \mathbf{R}^{m \times m}$ and $x_n \in x_0 + K_m(A, r_0)$, a Krylov m -th subspace with minimum residual.

2.3.3 Gauss Newton's Method

We first introduce **Newton's method**. Let $F : X \rightarrow Y$, and suppose dF and d^2F both exist and are invertible, we solve

$$\text{minimize } |F(x)|, \tag{2.3.4}$$

over $x \in X$, by iterating:

$$x_{n+1} = x_n - (F''(x_n))^{-1}F'(x_n). \tag{2.3.5}$$

Now, we introduce **Gauss-Newton** Nocedal and Wright (2006): Given $\bar{x} \in X$, we have:

$$\text{minimize } |F(\bar{x}) + J(\bar{x})^\top(x - \bar{x})|, \tag{2.3.6}$$

over $x \in X$. Now, given $\alpha_n \in \mathbf{R}$, damped Newton's method suggests:

$$x_{n+1} = x_n - \alpha_n(F'(x_n))^{-1}F(x_n). \tag{2.3.7}$$

Newton-Krylov iterate solves non-linear problems using Krylov subspace linear solvers. It may be possible to solve nonlinear equation without computing Jacobian inverse.

2.3.4 Quasi-Newton Method

First, we look at a simple Quasi-Newton method. The difference between a Quasi-Newton and a Newton method will be that a quasi-Newton method will involve approximated results for derivative calculation than exact result from Newton's method. In particular, given $f : \mathbf{R} \rightarrow \mathbf{R}$ be differentiable and dF non-singular, first consider a Newton iterate:

$$x_{k+1} = x_k - (f'(x_k))^{-1}f(x_k).$$

When one considers a finite difference scheme, the derivative term $f'(x_k)$, assuming existence and not equal to 0, will become $\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$. Replacing this into newton iteration

will result in

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k).$$

Next, we discuss the case when $f : \mathbf{R}^m \rightarrow \mathbf{R}^m$, with $m \in \mathbf{Z}_{\geq 1}$. Broyden's method Broyden (1965) generalizes a quasi-Newton method for finding roots in k variables. It is the most successful secant-method for solving systems of nonlinear equations. First, we see what a secant equation in the finite-difference approximation is for a function f with Jacobian matrix J :

$$J_n(x_n - x_{n-1}) \approx f(x_n) - f(x_{n-1}).$$

More compactly, one can write it as $J_n \Delta x_n \approx \Delta f_n$. Broyden, however, suggested to take the solution to the secant equation to update J_n from J_{n-1} :

$$J_n = J_{n-1} + \frac{\Delta f_n - J_{n-1} \Delta x_n}{|\Delta x_n|^2} \Delta x_n^\top.$$

With Sherman-Morrison-Woodbury Formula Sherman and Morrison (1949) Woodbury (1950), we have:

$$J_{n+1}^{-1} = J_n^{-1} + \frac{\Delta x_{n+1} - J_n^{-1} \Delta f_{n+1}}{\Delta x_{n+1}^\top J_n \Delta f_{n+1}} \Delta x_{n+1} J_n^{-1}.$$

It does not guarantee that every generated direction from iteration will be a descent direction, and store large amount of dataset. Having said that, when the initial iterate is near the solution, Broyden's method can perform very well. It is the simplest Quasi-Newton method.

Next, we discuss LBFGS (Broyden-Fletcher-Goldfarb-Shanno with limited memory) Broyden (1970) Fletcher (1970) Goldfarb (1970) Shanno (1970): It gives an approximation to Hessian matrix by determining the descent direction by preconditioning the gradient with curvature information. Given $H_0 = I$, Then, for all $n = 0, \dots, m$,

1. $p_n = -H_n \nabla f(x_n)$, where $\nabla f(x)$ is determined by Broyden update from above.
2. $\alpha_n = \operatorname{argmin} f(x_n + \alpha_n p_n)$, $s_n = \alpha_n p_n$, and update $x_{n+1} = x_n + s_n$.
3. $y_n = \nabla f(x_{n+1}) - \nabla f(x_n)$.

4. BFGS Update: With $\rho_n = \frac{1}{y_n^\top s_n}$, we have:

$$H_{n+1} = (I - \rho_n s_n y_n^\top) H_n (I - \rho_n y_n s_n^\top) + \rho_n s_n s_n^\top. \quad (2.3.8)$$

Similarly, for the Hessian inverse H_n^{-1} iteration, we have:

$$H_{n+1}^{-1} = H_n^{-1} + \frac{s_n^\top y_n + y_n^\top H_n^{-1} y_n}{s_n^\top y_n} s_n s_n^\top - \frac{H_n^{-1} y_n s_n + s_n y_n H_n^{-1}}{s_n^\top y_n}. \quad (2.3.9)$$

2.3.5 Gradient Descent and its Variants

We first introduce gradient descent as a specialized version of (1.1.2). Let $f : X \rightarrow \mathbf{R}$ be differentiable, and $\tau : X \rightarrow Y$. The idea for gradient descent is to take the steepest direction for the function to decrease to its minimal value.

Algorithm 2 Gradient Descent

Require: ϵ, x_0, f, τ .

Ensure: f_{\min} .

```

1: function GD( $f, x_0, \epsilon, \tau$ )
2:    $x \leftarrow x_0$ .
3:   while  $|\nabla f(x)| > \epsilon$  do
4:      $\tau \leftarrow \tau(x)$ 
5:      $x \leftarrow x - \tau \nabla f(x)$ .
6:   end while
7: end function

```

Note that the variable step size update

$$x_{n+1} = x_n - \tau_n \nabla f(x_n)$$

can be treated as a forward gradient descent.

Next, we study backward descent:

$$x_{n+1} = (I + \alpha_n \nabla f)^{-1} x_n, \quad (2.3.10)$$

provided that $(I + \alpha_n \nabla f)^{-1}$ is inexpensive to compute. Literatures similar to Parikh and Boyd (2014) referred backward gradient (2.3.10) as proximal gradient method.

Now, consider the problem:

$$\text{minimize} \quad g(x) + h(x), \quad (2.3.11)$$

over $x \in X$, where g is convex, continuous, and differentiable with continuous Lipschitz gradient, and h is convex, differentiable, and proper. The proximal method suggests that

$$x_{n+1} = \text{prox}_{\gamma_n h}(I + \gamma_n \nabla g)x_n,$$

where the proximal operator is defined by:

$$\text{prox}_{\gamma_n h}(x) = (I + \gamma_n \nabla g)^{-1}(x).$$

Similarly, one can develop Stochastic Proximal Gradient Descent, e.g. Nitanda (2014).

Remark 2.3.1. *There are ways to do gradient descent without evaluating gradient vector. For example, Malitsky and Mishchenko (2020) introduces a method adaptive to the local geometry, with convergence guarantees depending only on the smoothness in a neighborhood of a solution.*

A variant for Gradient Descent is the conjugate gradient (CG) method. We discuss CG in Section 5.1, which uses the geometry of function to increase convergence smoothness.

Then, we consider large scale optimization, with ROM-like gradient method,

$$x = x - \tau \tilde{\nabla} F(x).$$

Stochastic gradient (mimi-batch) method Consider the optimization

$$\text{minimize} \quad \sum_k g_k(x). \quad (2.3.12)$$

Let I_i be a randomly selected overlapped sub-indices of $I = \cup I_i$. We solve

$$\text{minimize} \quad \sum_{k \in I_i} g_k(x), \quad (2.3.13)$$

over the block I_i , by the gradient method

$$x_{i+1} = x_i - \sum_{k \in I_i} \beta_k g'_k(x_i),$$

where $\{\beta_k\}$ solves

$$\text{minimize} \quad \sum_{k \in I_i} g_k(x_i - \sum_{k \in I_i} \beta_k g'_k(x_i)), \quad (2.3.14)$$

or

$$\text{minimize} \quad \left| \sum_{k \in I_i} \beta_k g'_k(x_i) \right|^2. \quad (2.3.15)$$

If we select $\beta_k = \beta$, then

$$\text{minimize} \quad \sum_{k \in I_i} g_k(x_i - \beta \sum_{k \in I_i} g'_k(x_i)). \quad (2.3.16)$$

2.3.6 Nonlinear Kaczmarz Method

Kaczmarz method Strohmer and Vershynin (2009) helps to solve an overdetermined system $F(x) = 0$, for $F : X \rightarrow Y$. Now, we may take a random sub-indices sampling $j \in I_\ell \subset \{1, \dots, \dim(Y)\}$. Given $x_c \in X$, then we perform

$$\text{minimize} \quad \sum_{I \in I_\ell} |J_i(x - x_c) + F_i(x_c)|^2 + \alpha(x, x), \quad (2.3.17)$$

over $x - x_c \in D = \text{Span}\{d_1, \dots, d_m\}$, for each step, where

$$J_i d_j = \frac{F_i(x_c + t d_j) - F_i(x_c)}{t}.$$

is a difference approximation of $F'd_j$. One can solve them in a sequential (or parallel manner) to obtain sampling solutions x_ℓ , for $1 \leq \ell \leq L$ by

$$x_\ell = x_{\ell-1} + J_\ell^*(J_\ell^*J_\ell)^{-1}F_\ell(x_{\ell-1})$$

and ensemble them to obtain the update

$$x_{L+1} = \sum_{\ell=1}^L \alpha_\ell x_\ell.$$

2.4 Direct Iterative Inversion in the Iterative Subspace (DIIS)

We introduce DIIS as the method for updating solution through Anderson acceleration. DIIS Pulay (1980) updates the current solution as a linear combination from previous iterations. Consider the case of $F(x) = b - Ax$ with the goal of minimizing $|F(x)|^2$. Given basis vectors $(x_1, \dots, x_m) \in X \times \dots \times X$, with $e_i = b - Ax_i$ for $i \in \{1, \dots, m\}$, we have:

$$\begin{aligned} &\text{minimize} && \left| \sum_{i=1}^m c_i e_i \right|^2 \\ &\text{subject to} && \sum_i c_i = 1. \end{aligned} \tag{2.4.1}$$

With $B_{ij} = (e_i, e_j)$, one formulate the Lagrangian associated to (2.4.1) as:

$$L(c, \lambda) = c^\top Bc - 2\lambda(\sum_i c_i - 1).$$

Taking partial derivatives of L with respect to the coefficients and the multiplier and equating to 0 leads to:

$$\begin{bmatrix} B_{11} & \cdots & B_{1m} & -1 \\ \vdots & \ddots & \vdots & \vdots \\ B_{m1} & \cdots & B_{mm} & -1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{2.4.2}$$

Equating minus sign on λ and treating $-\lambda$ as λ results in:

$$\begin{bmatrix} B_{11} & \cdots & B_{1m} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ B_{m1} & \cdots & B_{mm} & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \quad (2.4.3)$$

We update the variable from the coefficients of (2.4.3) by

$$x_{m+1} = \sum_{i=1}^m c_i x_i. \quad (2.4.4)$$

Remark 2.4.1. Given $\epsilon > 0$, one can include a regularization sub-matrix ϵI_m into the (2.4.3) so that one solves

$$\begin{bmatrix} B_{11} + \epsilon & \cdots & B_{1m} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ B_{m1} & \cdots & B_{mm} + \epsilon & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad (2.4.5)$$

to update the variable coefficients for the desired system with c_1, \dots, c_m .

Chapter 3

Reduced Order Method

We develop Reduced Order Method (ROM), defined by Definition (1.2.1), from Section 1.2.2, and claim that ROM improves Anderson acceleration on efficiency and robustness, and then give some theorems that ROM help accelerate, regularize, and stabilize algorithms.

We consider ROM as a method to generate a sequence of solution by any iterative methods. In chapter 2, we have considered Newton's method in Section 2.3.3, kaczmarz's method in Section 2.3.6, Broyden update in Section 2.3.4 etc. The highlight is that we use a batch of historical solution as basis to generate a new solution iterate.

We now discuss regularized ROM. Let $X = \mathbf{R}^n$. ROM is applied to accelerate iterative scheme procedure, for example, with $\gamma \geq 0$,

$$\text{minimize} \quad |F(x)|^2 + \gamma\Psi(x), \tag{3.0.1}$$

over $x \in X$.

- When $\gamma = 0$, it is the least square problem to solve $F(x) = 0$.
- When $\gamma > 0$, the term $\gamma\Psi(x)$ regularizes the function $F(x)$. For example, $\Psi(x) = |x|^2$.

Definition 3.0.1 (Reduced Order Method). *Given $F : X \rightarrow X$ and $\{x_k\}_{k=1}^m \in X$*

generated from an iterative scheme, ROM is to solve the following optimization problem:

$$\text{minimize} \quad |F(\sum_{k=1}^m \alpha_k x_k)|^2 + \gamma \Psi(\sum_{k=1}^m \alpha_k x_k), \quad (3.0.2)$$

over coefficients $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbf{R}^m$, and we set

$$x_{m+1} = \sum_{k=1}^m \alpha_k^* x_k,$$

where $\alpha^* = (\alpha_1^*, \dots, \alpha_m^*)$ is an optimizer of (3.0.2).

3.1 ROM and Anderson Type Acceleration Comparison

In this section, we consider Anderson type acceleration for iterative scheme, including fixed point iterate. We consider ROM as a merit function based method for Anderson.

ROM gives an expression for Anderson-type acceleration, thus we compare ROM and Anderson-type acceleration. ROM justifies what Anderson does. Now, we assume the constraint

$$\sum_{k \leq m} \alpha_k = 1,$$

on ROM. Let $\{x_k\}_{k=1}^m \in X$ as basis generated by an iterative method. Let $F : X \rightarrow X$ be a nonlinear function. Let $r_k = F(x_k)$, and $\bar{\alpha} = (\bar{\alpha}_1, \dots, \bar{\alpha}_m)$ be a minimizer of

$$\begin{aligned} \text{minimize} \quad & \left| \sum_{k=1}^m \alpha_k r_k \right| \\ \text{subject to} \quad & \sum_{k=1}^m \alpha_k = 1. \end{aligned} \quad (3.1.1)$$

Now, we set a base point for Anderson approximation to ROM as

$$\bar{x} = \sum_{k=1}^m \bar{\alpha}_k x_k.$$

Note that from Taylor polynomial expansion, we have:

$$F\left(\sum_{k=1}^m \alpha_k d_k\right) \approx F(\bar{x}) + F'(\bar{x}) \sum_{k=1}^m (\alpha_k - \bar{\alpha}_k) x_k + 2\text{nd order}. \quad (3.1.2)$$

As a consequence, for $\bar{x} = \sum_k \bar{\alpha}_k x_k$, we have:

$$\sum_k \alpha_k r_k = \sum_k \alpha_k (F(x_k) - F(\bar{x})) + F(\bar{x}) \quad (3.1.3)$$

$$\sim \sum_k \alpha_k F'(\bar{x})(x_k - \bar{x}) + F(\bar{x}) + 2\text{nd order} \quad (3.1.4)$$

$$\sim F'(\bar{x}) \left(\sum_k \alpha_k x_k - \bar{x}\right) + F(\bar{x}) + 2\text{nd order} \quad (3.1.5)$$

$$\sim F\left(\sum_k \alpha_k x_k\right) + 2\text{nd order}, \quad (3.1.6)$$

and

$$2\text{nd order} \sim \frac{1}{2} \sum_k \alpha_k (H(\bar{x})(x_k - \bar{x}), x_k - \bar{x}). \quad (3.1.7)$$

Therefore, under the same constraint $\sum_k \alpha_k = 1$, Anderson (3.1.1) step approximates ROM step (3.0.2), assuming 2nd order term (3.1.7) being negligible, which resembles to (2.3.1) for the Newton-Krylov method. That is,

$$F\left(\sum_k (\alpha_k - \bar{\alpha}_k) x_k\right) \sim \sum_k J(\bar{x}) (\alpha_k - \bar{\alpha}_k) x_k + F(\bar{x}) + 2\text{nd order},$$

where

$$2\text{nd order} \sim \frac{1}{2} (H \sum_k (\alpha_k - \bar{\alpha}_k) x_k, \sum_k (\alpha_k - \bar{\alpha}_k) x_k),$$

and

$$F'(\bar{x}) x_k \sim J x_k = \frac{F(\bar{x} + t x_k) - F(\bar{x})}{t}, \quad t > 0.$$

In summary:

- Anderson solution can be used as a ROM solution, if it provides enough descent. ROM gives a theoretical justification on practical implementation of Anderson.

When F is nonlinear, the Anderson method approximates solution of ROM (3.0.2) over $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbf{R}^n$, together with $\bar{x} = \sum_k \bar{\alpha}_k d_k$,

$$\text{minimize} \quad \left| \sum_k J(\bar{x})^\top (\alpha_k - \bar{\alpha}_k) x_k + F(\bar{x}) \right| \quad (3.1.8)$$

over $\alpha = (\alpha_1, \dots, \alpha_n)$.

- In addition, (3.1.8) is a reduced order (subspace) Gauss-Newton method based on the basis $\{x_k\}$. For the linear case it is equivalent to the Krylov subspace method Saad and Schultz (1986). Thus, its performance depends on the conditioning of the $\{r_k\}$. If $\{r_k\}$ is nearly singular, the Anderson method is slowly convergent. If $\{r_k\}$ is linearly dependent, then we have a finite step convergence.
- When F is linear, Anderson method (1.2.6) is the same as ROM. Given a linear function $F : X \rightarrow X$, then solving least square by ROM

$$\begin{aligned} \text{minimize} \quad & |F(\sum_k \alpha_k x_k)|, \\ \text{subject to} \quad & \sum_k \alpha_k = 1, \end{aligned} \quad (3.1.9)$$

over $\alpha = (\alpha_1, \dots, \alpha_m)$ is equivalent to solving Anderson by

$$\begin{aligned} \text{minimize} \quad & \left| \sum_k \alpha_k F(x_k) \right| \\ \text{subject to} \quad & \sum_k \alpha_k = 1, \end{aligned} \quad (3.1.10)$$

over $\alpha = (\alpha_1, \dots, \alpha_m)$. As a consequence, we may use Anderson and ROM interchangeably when solving a linear F with acceleration methods.

Proof. When F is a linear function, we have that

$$F(\sum_k \alpha_k x_k) = \sum_k \alpha_k F(x_k).$$

Properties of norm suggests that

$$\left| F(\sum_k \alpha_k x_k) \right| = \left| \sum_k \alpha_k F(x_k) \right|,$$

which implies that solving least square by ROM is equivalent to solving least square by Anderson. \square

Therefore, one can interchangeably use Anderson as ROM when F is a linear function.

3.2 Anderson and ROM Acceleration as Evolutional Fixed point

In this section we develop the acceleration of Anderson type for the standard fixed point iterate

$$x_{i+1} = G(x_i).$$

With $a \in \text{dom}(G)$ as a fixed point for g , we modify $r_k = G(x_k) - x_k$ by setting $r_k = G(x_k) - G(a)$ with the optimization problem:

$$\text{minimize} \quad \left| \sum_{k \leq n} \alpha_k r_k \right|^2, \tag{3.2.1}$$

together with the sequential update x_{n+1} defined by

$$x_{n+1} = \sum_{k \leq n} \alpha_k^* x_k. \tag{3.2.2}$$

where α^* is an optimizer of (3.2.1). It is different from the original form of the Anderson update for fixed point iterate Walker and Ni (2011). The step (3.2.2) regularizes and stabilizes the fixed point update, i.e. the derivative gets smaller when considering the ROM update than the usual update. It works without assuming the contraction of fixed point iterate.

Proposition 3.2.1 (Stability to a Fixed Point). *Let $G : X \rightarrow X$ be a map with fixed point(s). Given $x_0 \in X$, with Banach iterate to generate $(x_1, \dots, x_n) \in X \times \dots \times X$. By solving (3.2.1) with minimizer $\alpha^* = (\alpha_1^*, \dots, \alpha_n^*) \in \mathbf{R}^n$, we have obtained a more stabilized solution. Given $a \in X$ as a fixed point for G , $a \in \text{Span}\{x_1, \dots, x_n\}$, and*

$G(x_n) \in \text{Span}\{x_1, \dots, x_n\}$, we have:

$$|G(\sum_{k \leq n} \alpha_k^* x_k) - a|_X \leq |G(G(x_n)) - a|_X.$$

Proof. Note that, with definition of α^* , for any $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbf{R}^n$, we have:

$$|\sum_k \alpha_k^* r_k| \leq |\sum_k \alpha_k r_k|. \quad (3.2.3)$$

Therefore, with (3.1.6), we get

$$|F(\sum_k \alpha_k^* x_k)| \leq |F(\sum_k \alpha_k x_k)|.$$

Apply the definition of F to get:

$$|G(\sum_k \alpha_k^* x_k) - a| \leq |G(\sum_k \alpha_k x_k) - a|.$$

Since $g(x_n) \in \text{Span}\{x_1, \dots, x_n\}$, there exists $\tilde{\alpha} = (\tilde{\alpha}_1, \dots, \tilde{\alpha}_n) \in \mathbf{R}^n$ such that $G(x_n) = \sum_k \tilde{\alpha}_k x_k$. Apply $\tilde{\alpha}$ as α to get the desired result, i.e.

$$|G(\sum_{k \leq n} \alpha_k^* x_k) - a|_X \leq |G(G(x_n)) - a|_X.$$

□

Corollary 3.2.1 (Stability to a Neighbourhood for a Fixed Point). *Under the same assumptions from the above theorem, suppose that we have $y_m = \sum_{k=1}^n \alpha_k^m x_k$ such that $\{y_m\}$ converges to a . Then, for all large enough m , we have*

$$|G(\sum_{k \leq n} \alpha_k^* x_k) - G(y_m)|_X \leq |G(G(x_n)) - G(y_m)|_X.$$

More studies of Anderson on fixed point convergence on non-contractive mapping Pollock and Rebholz (2021), Xue (2022).

3.3 Variants of ROM

We develop ROM variants to see further opportunities to accelerate ROM.

3.3.1 Nested ROM

We perform the nested ROM, i.e. we progressively generate sample solution sequence $\{z_i\}$ with $1 \leq i \leq m$ and $m \in \mathbf{Z}_{\geq 2}$, by the Anderson updates. It follows the reduced order element method Ito and Ravindran (1998) with the iterates $\{x_k\}$ as basis, i.e. the linear manifold spanned by $\{x_k\}$ captures the solution well by minimizing

$$\phi\left(\sum_{k \leq n} \alpha_k x_k\right) = \left|F\left(\sum_{k \leq n} \alpha_k x_k\right)\right|^2 + \beta \Psi\left(\sum_{k \leq n} \alpha_k x_k\right), \quad (3.3.1)$$

where $\beta \geq 0$ and Ψ is a regularization function.

We discuss Nested Anderson with an application on improving the fixed point algorithm. Suppose that we want to find a fixed point for a given contractive map $\phi : \mathbf{R}^n \rightarrow \mathbf{R}^n$. Define $F(x) = x - \phi(x)$. Let the Anderson method define a fixed point map ϕ by

$$\phi(\theta) = \theta^*, \quad (3.3.2)$$

where θ is the initial and θ^* is the corresponding the m -step Anderson iterate. In general, the Anderson map (3.3.2) has a much better contraction than ϕ . That is, we apply the Anderson method for the fixed point problem for ϕ as a nested iterate.

ROM with m step can be nested as: let $y_1 = y$ be 2-step ROM solution be defined by

$$y = \sum_{k=1}^2 \alpha_k^* x_k,$$

with $\alpha_k^* \in \mathbf{R}$ as optimized weight for iterate x_k . Then, based on the previous solutions, we generate an additional r based on the initial condition y and use 3-step ROM to get y_2 , so on. Then, after m times, we progressively obtain a sampled solution sequence $\{y_i\}$.

We obtain refine approximation by nested ROM update

$$z = \sum_{k=1}^m \beta_k^* y_k,$$

where $\beta^* = (\beta_1^*, \dots, \beta_m^*) \in \mathbf{R}^m$ minimizes

$$|F(\sum_{k=1}^m \beta_k^* y_k)|^2.$$

Also, we do an additional refinement based on $\{z_i\}_{1 \leq i \leq m}$, as a nested ROM. It notes that this improves accuracy and the convergence very much and that they produce a monotone descent approximation sequences for

$$\text{minimize } |F(x)|^2. \tag{3.3.3}$$

3.3.2 Sampled ROM

Sampled ROM is a similar method to Nested ROM but need to restart after each m -step ROM. We repeat m -step ROM for sometime with update and we collect the update together as a new basis for our new ROM. Mathematically speaking, Sampled ROM step can be described as: let $y_1 = y$ be m -step ROM solution be defined by

$$y_1 = \sum_{k=1}^m \alpha_k^* x_k,$$

with $\alpha_k^* \in \mathbf{R}$ as optimized weight for iterate x_k . Then, we restart ROM from y_1 and repeat the m -step ROM method with initial y to get y_2 , so on. Then, we progressively obtain a sampled solution sequence $\{y_i\}$. We obtain a refined approximation by nested ROM update

$$z = \sum_{k=1}^m \beta_k^* y_k,$$

where $\beta^* = (\beta_1^*, \dots, \beta_m^*) \in \mathbf{R}^m$ minimizes

$$|F(\sum_{k=1}^m \beta_k y_k)|^2.$$

3.3.3 Residual Based ROM

We develop Residual based ROM update. We first repeat an iterative scheme for m times to get residual direction r_k with $(1 \leq k \leq m)$ and x_n . For $\min |F(x)|^2$, we perform ROM step:

$$\text{minimize} \quad |F(x_n + \sum_{k \leq m} \alpha_k r_k)|^2, \quad (3.3.4)$$

over α , and let $\alpha^* = (\alpha_1^*, \dots, \alpha_m^*) \in \mathbf{R}^m$ be the optimal coefficient. We assume that r_k will get closer to 0. This has an advantage that it defines the prediction x_{n+1} starting from the current iterate x_n and uses the history of residual direction $\{r_k\}$. The optimization step corresponds to the case of nonlinear optimization (5.1.2): find α so as to

$$\begin{aligned} \text{minimize} \quad & f(x_n + \sum_{k \leq m} \alpha_k r_k) \\ \text{subject to} \quad & x_n + \sum_{k \leq m} \alpha_k r_k \in \mathcal{C}. \end{aligned} \quad (3.3.5)$$

3.3.4 ROM Extrapolation

Consider the least square problem of

$$\text{minimize} \quad |F(x)|^2, \quad (3.3.6)$$

over $x \in X$. We introduce the ROM update

$$\text{minimize} \quad |F(x_m + \sum_{i=1}^{m-1} \alpha_i r_i)|^2, \quad (3.3.7)$$

or

$$\text{minimize} \quad |F(\sum_{i=1}^m \alpha_i x_i)|^2, \quad (3.3.8)$$

over α , where the basis (x_i, p_i) is generated by a well-posed linear system $\bar{A}x - b = 0$ for the (\bar{A}, b) instead of $F(x) = 0$, i.e. we generate basis functions by the alternative nominal equation $\bar{A}x - b = 0$ for the original (ill-posed) problem of minimizing $|F(x)|^2$,

and update x_{k+1} from x_k by

$$x_{k+1} = x_k + \alpha_k p_k, \quad p_k = b - \bar{A}x_k.$$

One can generate the diverse free basis by minres and one can use ROM to find α to

$$\text{minimize} \quad |F(\sum_k \alpha_k x_k)|^2, \quad (3.3.9)$$

or by the Galerkin projection

$$(F(\sum_k \alpha_k x_k), x_j) = 0 \text{ for all } x_j.$$

3.3.5 ROM with Preconditioning

In this section we discuss the preconditioning of $Ax - b = 0$ by a pre-conditioner B , i.e.,

$$B^{-1}(Ax - b) = 0,$$

where

$$B^{-1}A = I + B^{-1}(A - B), \quad |B^{-1}(A - B)| < 1.$$

We generate the basis $\{(x_k, r_k)\}$ with an iterative method for solving $Bx = b$:

$$x_{k+1} = x_k + \alpha_k p_k, \quad p_k = b - Bx_k.$$

We approximate B^{-1} by the polynomial approximation P defined by

$$B^{-1}x \sim Px = x + \sum_{k=1}^m \alpha_k^* x_k,$$

where α_k^* minimizes

$$|B(x + \sum_k \alpha_k x_k) - b|^2.$$

Then, we use the variable fixed point for $P(Ax - b) = 0$ by

$$x_{n+1} = x_n + \beta (P(Ax_n - b)).$$

We apply ROM method to accelerate the convergence by minimizing

$$\text{minimize} \quad |A(\sum_{k=1}^m \beta_k x_k) - b|^2, \quad (3.3.10)$$

over $\beta = (\beta_1, \dots, \beta_m)$, and let $\beta^* = (\beta_1^*, \dots, \beta_m^*)$ be its solution and update x_{m+1} by

$$x_{m+1} = \sum_{k=1}^m \beta_k^* x_k.$$

3.3.6 Back-projection Method (Dual) A^*b

In the inverse scattering problems we use the the filtered back projection

$$A^*Pb$$

for the least square solution $A^*(AA^*)^{-1}b$ of minimizing $|Ax - b|^2$, where

$$(AA^*)^{-1} = \text{pseudo differential operator}$$

and is very badly conditioned. We replace it by a properly chosen differential operator P , e.g., $(-\Delta)^\gamma$) We use the fixed point iterate of the form

$$x_{i+1} = x_i - A^*P(Ax_i - b).$$

Note that

$$((I - A^*PA)x, x) \leq (1 - \omega) |x|^2,$$

for a properly chosen P . That is, the standard fixed point method converges. We apply the ROM acceleration to improve the convergence.

We consider solving a better posed problem by ROM as an effective to solve the actual problem.

3.4 Applications

We list some possible applications in mind.

(1) (Damped Newton Update) We apply ROM on damped Newton update. We assume the (damped) Newton updates u_i ($1 \leq i \leq m$), with damped step size $\{\alpha_i\}$ to generate a basis by one parameter family:

$$u_{i+1} = u_i - \alpha_i J_i^* (J_i J_i^*)^{-1} F(u_i),$$

where J_i is an approximated Jacobian for $F(u_i)$. We consider the refinement (ROM) step

$$\text{minimize} \quad |F(\sum_{k \leq m} \beta_k u_k)|^2, \tag{3.4.1}$$

over $\{\beta_k\}$ based on the current updates $\{u_k\}$. This optimization over β_k is performed by Gauss-Newton (3.1.8). We update

$$u_{m+1} = \sum_{k=1}^m \beta_k^* u_k,$$

where $\{u_k\}$ is the history of solutions generated by the Newton's iterate and followed by the ROM optimization step. This guarantees the decent property of F .

In general, $\{u_n\}$ is a sampling (sequentially generated) sequence of estimated solutions and forms a reduced order linear solution manifold over which $|F|^2$ is minimized. Also, we include the reduced order elements $\{\tilde{u}\}$ to the solution manifold as in Remark (3) below.

As discussed in Section 3.1, the Anderson extrapolation solves approximately ROM step.

(2) (Damped Fixed Point Problem) For the fixed point problem one considers we let $\{u_i\}$ be fixed point solution to one parameter family $\{\theta_i\}$ of damped fixed points:

$$u_i - \theta_i g(u_i) = 0, \quad 0 < \theta_i < 1,$$

and u_i is the corresponding damped basis for the damped fixed point problem.

(3) (Equations with Internal Variables) Also, we include the reduced order elements $\{\tilde{u}\}$ to the linear solution manifold as a Reduced order element method Ito and Ravindran (1998). Let $F_\theta(x) = 0$ be the parameter dependent equation and driven sampled parameters θ , and let $x = (x_1, \dots, x_n)$ be the corresponding solution. Then look for α to minimize

$$|F_\theta(\sum_i \alpha_i x_i)|^2.$$

(4) (AR auto-regressive) We introduce the prediction method as the time-series of the form

$$u_{i+1} = \sum_{k \leq i} \beta_k u_k,$$

where β_k is determined by minimizing $|y_k - C(u_k)|^2$, or, in general minimizing $|F(u_k)|^2$.

Chapter 4

Variable Step Update

In this chapter, we focus on developing variable step method (1.1.2) to generate basis solutions. For example, gradient method from Section 2.3.5 and Netwon's method from Section 2.3.3 are examples of variable step size update. The importance is variable step ROM update make iteration converge when we consider basis element in the reduced space is updated iteration by iteration from line search. It accelerates the convergence and is monotonically decreasing step by step. For each iteration, we use the Cauchy step (1.2.9) as the optimal step size. Similar to results in Chapter 3, we do variable step size update from an unconstrained convex optimization context, even for the fixed point. The acceleration method is one shot, meaning that every step is unique. During the process of doing iteration, we want scaling, stabilizing, and sufficient decrement Goldfarb (1970) Wolfe (1969).

Throughout the chapter, we assume that $(X, |\cdot|_X)$ is a Banach space.

4.1 Variable Step Size Update

In this section, we study methods with use of Cauchy step. Let $F : X \rightarrow X$, we minimize $|F(x)|^2$ over $x \in X$.

Definition 4.1.1 (Variable Residual Error Step Update). *Given F and $x_0 \in X$, the*

variable residual error step update is an iterate with the form:

$$x_{n+1} = x_n + \alpha_n F(x_n), \quad (4.1.1)$$

where α_n is the variable step-size.

Example 4.1.1. Variable Step Size Update We provide examples to give different interpretations of making a variable step size update.

1. Let $G : x \in X \rightarrow G(x) \in X$ so that there exists $x^* \in X$ so that $G(x^*) = x^*$, and let $F : X \rightarrow X$ be defined by:

$$F(x) = x - G(x). \quad (4.1.2)$$

We interpret (4.1.2) as a fixed point iterate scheme.

2. Given a matrix A and vector b , both with comfortable dimensions, we consider $F : X \rightarrow X$ defined by $F(x) = b - Ax$. With a given $\alpha_n \in \mathbf{R}$, we have

$$x_{n+1} = x_n + \alpha_n(b - Ax_n). \quad (4.1.3)$$

We interpret (4.1.3) as a scheme to solve $Ax = b$ iteratively.

3. When there exists a differentiable function $g : X \rightarrow X$ so that $F(x) = \nabla g(x)$, with $g_n = F(x_n)$, we interpret the scheme

$$x_{n+1} = x_n + \alpha_n g_n. \quad (4.1.4)$$

as the gradient descent iterate.

Next, We define ‘‘Cauchy step’’ Cauchy (1847).

Definition 4.1.2 (Cauchy Step). Let $F : X \rightarrow X$. The Cauchy step $\alpha_n \in \mathbf{R}$ at $x_n \in X$ along direction $p_n \in X$ is defined as

$$\alpha_n^* = \min |F(x_n + \alpha p_n)|^2,$$

over $\alpha \in \mathbf{R}$.

We consider the use of Cauchy step into numerical algorithms as Cauchy's method.

4.1.1 Cauchy's Method

We derive Cauchy step formulas for least square problems and optimization problems.

Example 4.1.2. Consider a linear system $F : x \in X \rightarrow F(x) \in X$ defined by

$$F(x) = b - Ax,$$

with $A \in \mathbf{R}^{n \times n}$ and $b \in \mathbf{R}^n$. We aim to solve

$$\text{minimize } |F(x)|^2, \tag{4.1.5}$$

over $x \in X$. On the n -th iteration, set $r_n = b - Ax_n$. In the context of (4.1.1), we have the Cauchy Step on iteration n as:

$$\alpha_n = \frac{(r_n, Ar_n)}{(Ar_n, Ar_n)}. \tag{4.1.6}$$

Proof. Set $G(\alpha) = |b - A(x_n + \alpha_n d_n)|_2^2$, evaluate $\frac{dG}{d\alpha_n}$, and set $\frac{dG}{d\alpha_n} = 0$ to get

$$(b - A(x_n + \alpha_n d_n))^\top (Ad_n) = 0.$$

With re-arrangement, one gets

$$\alpha_n = \frac{(b - Ax_n, Ad_n)}{(Ad_n, Ad_n)} = \frac{(r_n, Ad_n)}{(Ad_n, Ad_n)}.$$

In the context of (4.1.1), we have $d_n = r_n$, then we have:

$$\alpha_n = \frac{(r_n, Ar_n)}{(Ar_n, Ar_n)}.$$

□

We now calculate Cauchy step for nonlinear least square problems. Consider a nonlinear map $F : X \rightarrow X$, We wish to determine the optimal step size α that minimizes

$$|F(x + \alpha p)|^2$$

over $\alpha \in \mathbf{R}$. This is a scalar minimization problem. Put

$$x_{n+1} = x_n + \alpha_n p_n,$$

we have α_n is a step size that minimizes

$$|F(x_n + \alpha_n p_n)|^2.$$

Since $F(x + \alpha p) - F(x) \sim \alpha Jp$, where:

$$Jp = \frac{F(x + tp) - F(x)}{t}, \quad t > 0.$$

Approximately, we are solving

$$\text{minimize} \quad |F(x) + \alpha Jp|^2, \tag{4.1.7}$$

over α . Back into the context of ROM, we put $r_n = F(x_n)$, $p = p_n$. Using a similar approach from Example 4.1.2, we have an approximated Cauchy step

$$\alpha_n = -\frac{(r_n, Jp_n)}{(Jp_n, Jp_n)},$$

where we update J on each new step.

Merit Function Optimization: For a badly conditioned positive and symmetric matrix A with $f : X \rightarrow \mathbf{R}$, we consider

$$\text{minimize} \quad f(x) = \frac{1}{2}(x, Ax) - b^\top x, \tag{4.1.8}$$

over $x \in X$. The necessary optimally condition is $Ax - b = 0$.

Theorem 4.1.1. *The Cauchy step for (4.1.8) is*

$$\alpha_k = -\frac{g_k^\top g_k}{g_k^\top A g_k},$$

where $r_k = Ax_k - b$, $g_k = r_k$, and $p_k = g_k$ is the search direction for each iterate.

Proof. Let $f : x \in \mathbf{R}^n \rightarrow f(x) \in \mathbf{R}$ be defined as

$$f(x) = \frac{1}{2}(x, Ax) - b^\top x.$$

For all $d \in \mathbf{R}^n$ and for all $\alpha \in \mathbf{R}$, we have:

$$f(x + \alpha d) = \frac{1}{2}(x + \alpha d)^\top A(x + \alpha d) - b^\top (x + \alpha d).$$

After simplification, we get:

$$f(x + \alpha d) = \frac{1}{2}(x^\top Ax + 2\alpha x^\top Ad + \alpha^2 d^\top Ad) - \alpha b^\top d - b^\top x.$$

Solve $\frac{\partial f}{\partial \alpha} = 0$ will lead to

$$\alpha_k = -\frac{g_k^\top g_k}{g_k^\top A g_k}.$$

□

4.1.2 Variants to Cauchy Methods

We discuss methods related to Cauchy's methods. For example, we can compute by inexact line search. We now look at some examples of inexact line search method.

- In a quasi-Newton iteration,

$$x_{n+1} = x_n + \alpha_n B_n^{-1} F(x_n),$$

where B_n is some approximation of the Jacobian matrix $F'(x_n)$, e.g. Broyden Up-

date. We also consider the feedback form:

$$x_{n+1} = x_n - \alpha_n G_n F(x_n),$$

where G_n is a feedback gain operator.

- Barzilai Borwein method by Barzilai and Borwein (1988) seeks to combine the simplicity from Gradient and the computational speed from Newton. Therefore, it provides an approximation to quasi-Newton. With $y_{n-1} = g_n - g_{n-1}$ and $s_{n-1} = x_n - x_{n-1}$, we have Algorithm (3).

Algorithm 3 Barzilai-Borwein

Require: $x_0 \in \mathbf{R}^n$, $f : \mathbf{R}^n \rightarrow \mathbf{R}$, ϵ

Ensure: x

```

1: for  $k = 0, \dots$  do
2:    $g_k = \nabla f(x_k)$ 
3:    $d_k = -g_k$ .
4:   if  $|g_k| > \epsilon$  then
5:      $\alpha_k = \operatorname{argmin}_{\alpha} |F(x_k + \alpha d_k)|^2$ .
6:      $\alpha_k^{(I)} = \frac{s_{k-1}^\top y_{k-1}}{y_{k-1}^\top y_{k-1}}$  or  $\alpha_k^{(II)} = \frac{s_{k-1}^\top s_{k-1}}{s_{k-1}^\top y_{k-1}}$ .
7:      $x_{k+1} = x_k + \alpha_k d_k$ .
8:   else
9:     Stop the algorithm and return  $x$ .
10:  end if
11: end for

```

Remark 4.1.1. *The conjugate gradient Cauchy step in Example 4.1.1 corresponds to $\alpha_k^{(II)}$ in Barzilai-Borwein Algorithm (3); The variable fixed point Cauchy step in Theorem 4.1.2 corresponds to $\alpha_k^{(I)}$ in Barzilai-Borwein Algorithm (3).*

Pre-conditioned Conjugate Gradient (PCG), given by

$$\begin{aligned}\alpha_k &= \frac{(r_k, z_k)}{(Ap_k, p_k)}, \quad x_{k+1} = x_k + \alpha_k p_k, \\ r_{k+1} &= r_k - \alpha_k Ap_k, \quad z_{k+1} = P^{-1}r_{k+1}, \\ \beta_k &= \frac{(z_{k+1}, r_{k+1})}{(z_k, r_k)}, \quad p_{k+1} = z_{k+1} - \beta_k p_k,\end{aligned}$$

where P is the pre-conditioner and z_k is the pre-conditioned residual, accelerates merit function optimization convergence (4.1.8). If we start with $x_0 = 0$, then x_k minimizes Paquette and Trogon (2023)

$$x_k = \operatorname{argmin}\{(x - y, A(x - y)) : y \in \operatorname{Span}\{b, Ab, \dots, A^{k-1}b\}\}. \quad (4.1.9)$$

An alternative Cauchy Step to minimize $|b - Ax|^2$ over $x \in X$: Let \bar{A} be an approximated matrix for A . Then, we set

$$\alpha_n = \frac{(r_n, \bar{A}r_n)}{(\bar{A}r_n, \bar{A}r_n)}. \quad (4.1.10)$$

We update x_k by

$$x_{k+1} = x_k + \alpha_k r_k, \quad r_k = b - Ax_k.$$

After m updates, update x_{m+1} by solving (2.4.3) with update (2.4.4). We discuss more details in Section 7.2.2.

4.1.3 Along with ROM

Finally, we develop the ROM method to accelerate the convergence for least square problems and merit function optimization case. For the least square problem optimization, the procedure is very similar to the fixed step size case. We solve

$$\operatorname{minimize} \quad |F(\sum_{k \leq n} \beta_k x_k)|^2, \quad (4.1.11)$$

over β . Then, with β^* as optimizer, one restarts the iterate with x_{n+1} as the initial for a new iterate. ROM step is the post-conditioning and does accelerate the convergence

and results in the monotone convergent algorithm. It converges very rapidly provided that the condition number of A is not too large. Otherwise, we need to pre-condition the problem.

But, one can improve the convergence of CG with ROM acceleration. Also, one can use conjugate iterate without assuming A being positive definite and symmetric with a modified step size α_k , say Cauchy step with ROM acceleration. We repeat this for m times, to generate $(x_1, \dots, x_m) \in \mathbf{R}^n \times \dots \times \mathbf{R}^n$. That is, with $r_k = b - Ax_k$, ROM minimizes

$$|\sum_{k \leq m} \alpha_k r_k|,$$

compared with (4.1.9). With $\alpha^* = (\alpha_1^*, \dots, \alpha_m^*) \in \mathbf{R}^m$ as optimal solution, we let

$$x_{m+1} = \sum_{k \leq m} \alpha_k^* x_k.$$

4.2 Multi-Step Subspace Update

We generalize derivation of Cauchy step of with 1 directions from Section 4.1 into 2 directions. In order to motivate our discussions, we consider linear system $F(x) = b - Ax$, and the case of (3.3.4) when $m = 2$: we solve

$$\text{minimize } |b - A(x_n + \alpha_1 r_1 + \alpha_2 r_2)|, \quad (4.2.1)$$

over $(\alpha_1, \alpha_2) \in \mathbf{R}^2$. We do the subspace update spanned by $\{r_1, r_2\}$:

$$x_{n+1} = x_n + (\alpha_n r_1 + \beta_n r_2). \quad (4.2.2)$$

However, we may improve (4.2.2) by considering two arbitrary directions $\{r_n, p_n\}$ for the n -th iterate, i.e. For the linear system $F(x) = Ax - b$, given $\{r_n, p_n\}$, when trying to

$$\text{minimize } |A(x_n + \alpha_n r_n + \beta_n p_n) - b|^2, \quad (4.2.3)$$

over $(\alpha, \beta) \in \mathbf{R}^2$. By optimality condition, with $r_n = b - Ax_n$, using the approach to solve Example 4.1.2, we have:

$$\begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} = \begin{pmatrix} (Ar_n, Ar_n) & (Ar_n, Ap_n) \\ (Ap_n, Ar_n) & (Ap_n, Ap_n) \end{pmatrix}^{-1} \begin{pmatrix} (b - Ax_n, Ar_n) \\ (b - Ax_n, Ap_n) \end{pmatrix}.$$

We do the subspace update spanned by $\{r_n, p_n\}$:

$$x_{n+1} = x_n + (\alpha_n r_n + \beta_n p_n). \quad (4.2.4)$$

As an example, we do the subspace update spanned by $\{r_n, Ar_n\}$:

$$x_{n+1} = x_n + (\alpha_n r_n + \beta_n Ar_n), \quad (4.2.5)$$

where with $r_n = b - Ax_n$, $p_n = Ar_n$, we have

$$\begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} = \begin{pmatrix} (Ar_n, Ar_n) & (Ar_n, Ap_n) \\ (Ap_n, Ar_n) & (Ap_n, Ap_n) \end{pmatrix}^{-1} \begin{pmatrix} (r_n, Ar_n) \\ (r_n, Ap_n) \end{pmatrix}. \quad (4.2.6)$$

In order to improve the convergence, one considers the one-step conjugate direction update for the search direction p_n :

$$p_n = r_n - \frac{(r_n, p_{n-1})}{(p_{n-1}, p_{n-1})} p_{n-1} \implies (p_n, p_{n-1}) = 0. \quad (4.2.7)$$

Similarly, as an alternative, for step size update, we have:

$$\begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} = \begin{pmatrix} (\bar{A}r_n, \bar{A}r_n) & (\bar{A}r_n, \bar{A}p_n) \\ (\bar{A}p_n, \bar{A}r_n) & (\bar{A}p_n, \bar{A}p_n) \end{pmatrix}^{-1} \begin{pmatrix} (r_n, \bar{A}r_n) \\ (r_n, \bar{A}p_n) \end{pmatrix}. \quad (4.2.8)$$

We replace A by an approximated matrix \bar{A} in the matrix for inversion. For updating directions, we consider directions r_n , and $\bar{A}r_n$, with $r_n = b - Ax_n$.

4.3 Applications

4.3.1 Regularization

As a first application, we derive Cauchy step for minimizing F with a regularization function Ψ by sequentially generating $\{x_1, \dots, x_m\}$ by (4.2.5) or (4.2.7) and use it as a basis for the solution. That is, given $\beta \geq 0$, we use the reduced order method (ROM) Ito and Ravindran (1998):

$$\text{minimize} \quad |F(\sum_k \gamma_k x_k)|^2 + \beta \Psi(\sum_k \gamma_k x_k), \quad (4.3.1)$$

over $\gamma = (\gamma_1, \dots, \gamma_m) \in \mathbf{R}^m$, and set the extrapolation of $\{x_k\}$ by

$$x_{m+1} = \sum_{k \leq m} \gamma_k^* x_k \quad (4.3.2)$$

It also defines the stabilization regularization of iterates $\{x_k\}$ and we select an appropriate Ψ to enhance such properties.

Specifically, we provide a scheme to evaluate a case for a linear function $F : \mathbf{R}^n \rightarrow \mathbf{R}^n$ defined by $F(x) = b - Ax$ iteratively. In order to motivate our discussions, we consider linear system $Ax = b$ and $\psi(x) = |x|^2$. Then, we can cast the regularized least squares problem

$$F(x) = |b - Ax|^2 + \beta |x|^2, \quad \beta \geq 0. \quad (4.3.3)$$

We consider the iterative variable fixed point method

$$x_{n+1} = x_n + \alpha_n (b - Ax_n), \quad r_n = b - Ax_n.$$

The step size α_n minimizes

$$|A(x_n + \alpha_n p_n) - b|^2 + \beta |x_n + \alpha_n p_n|^2, \quad p_n = b - Ax_n,$$

over $\alpha_n \in \mathbf{R}$. If $\beta \neq 0$, with similar approach to , we have the Cauchy step:

$$\alpha_n = \frac{r_n^\top Ap_n - \beta x_n^\top p_n}{|Ap_n|^2 + \beta |p_n|^2},$$

and with $F(x) = |b - Ax|^2 + \beta |x|^2$, we have

$$F(x_{n+1}) = F(x_n) - \frac{(\beta x_n^\top p_n - r_n^\top Ap_n)^2}{|Ap_n|^2 + \beta |p_n|^2}.$$

Similarly, regarding to (4.3.3), one can do the two directions $\{p_n, q_n\}$ update by considering

$$x_{n+1} = x_n + \alpha_n r_n + \beta_n p_n.$$

The step size pair $(\alpha_n, \beta_n) \in \mathbf{R}^2$ minimizes

$$|A(x_n + \alpha_n p r_n + \beta_n p_n) - b|^2 + \beta |x_n + \alpha_n r_n + \beta_n p_n|^2.$$

Then, we have:

$$\begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} = \begin{pmatrix} (Ar_n, Ar_n) + \beta(r_n, r_n) & (Ar_n, Ap_n) + \beta(p_n, r_n) \\ (Ar_n, Ap_n) + \beta(r_n, p_n) & (Ap_n, Ap_n) + \beta(p_n, p_n) \end{pmatrix}^{-1} \begin{pmatrix} (b - Ax_n, Ar_n) \\ (b - Ax_n, Ap_n) \end{pmatrix}.$$

4.3.2 Saddle Point Problem

We consider Saddle Point Problem as a special case of (4.3.3). Given a symmetric positive definite matrix A , consider the constrained minimization problem:

$$\begin{aligned} & \text{minimize} && \frac{1}{2}(x, Ax) - (a, x) \\ & \text{subject to} && Ex = b. \end{aligned} \tag{4.3.4}$$

Given $\epsilon > 0$, define penalty function associated from above as:

$$L(x) = \frac{1}{2}(x, Ax) - (a, x) + \frac{1}{2\epsilon}|Ex - b|^2.$$

Then, we have:

$$Ax_\epsilon + \frac{1}{\epsilon}E^\top(Ex_\epsilon - b) = a.$$

With $\lambda_\epsilon = \frac{Ex_\epsilon - b}{\epsilon}$, it follows that the optimal pair $(x_\epsilon, \lambda_\epsilon)$ satisfies the regularized saddle-point problem $Aw = c$, where

$$A = \begin{pmatrix} A & E^* \\ E & -\epsilon I \end{pmatrix}, w = \begin{pmatrix} x_\epsilon \\ \lambda_\epsilon \end{pmatrix}, c = \begin{pmatrix} a \\ b \end{pmatrix}.$$

Example 4.3.1. Given a symmetric matrix D , if $F(x) = \frac{1}{2}(x, Dx) - (f, x)$ and $E(x) = Ex - g$, then we formulate Lagrangian $L(x, \lambda) = \frac{1}{2}(x, Dx) - (f, x) + \lambda^\top(Ex - g)$ and take derivative on x and λ to have a linear saddle point problem $Au = b$, where

$$A = \begin{pmatrix} D & E^* \\ E & 0 \end{pmatrix}, u = \begin{pmatrix} x \\ \lambda \end{pmatrix}, b = \begin{pmatrix} f \\ g \end{pmatrix}.$$

In general, for a symmetric positive definite matrix D of size $n \times n$ and E of size $m \times m$ of full rank, in terms of A of size $2(n + m) \times 2(n + m)$ of the symmetric form:

$$A = \begin{pmatrix} D & E^* \\ E & F \end{pmatrix}, \tag{4.3.5}$$

we have n positive and m negative eigenvalues. With x and b of appropriate size, and A of the form 4.3.5, we refer the problem of solving $Ax = b$ as a **Saddle Point Problem**.

In Section 7.2, we perform an example of Saddle Point problem of a specific class, Stokes problem, with no-slip wall boundary condition, since its numerical discretizations gives a saddle point problem form.

Chapter 5

Convex Optimization

In this chapter, as further applications to ROM, we discuss convex optimization. We study conjugate gradient method, conjugate residual method, and proximal gradient method.

5.1 Conjugate Gradient Method

In this section, we develop CG with ROM on a convex optimization problem. First, recall that Hestenes and Stiefel Hestenes and Stiefel (1952) introduced Conjugate Gradient (CG) to solve

$$\text{minimize} \quad \frac{1}{2}x^\top Ax - b^\top x, \quad (5.1.1)$$

over $x \in \mathbf{R}^m$, for a symmetric positive definite matrix $A \in \mathbf{R}^{m \times m}$. The necessary and sufficient optimality condition is $Ax = b$. It upgrades step size and search direction in each iteration for both the current solution and residual.

Algorithm 4 Conjugate Gradient Method

Require: $x_0, p_0, A, \mathbf{x}_0, b$

Ensure: x so that $Ax = b$.

```
function CG( $x_0, p_0, x_0, A, b$ )  
   $r_0 = -(Ax_0 - b), p_0 = r_0;$   
  for  $k = 0, \dots$  do  
     $\alpha_k = \frac{r_k^\top r_k}{p_k^\top A p_k}$   
     $x_{k+1} = x_k + \alpha_k p_k$   
     $r_{k+1} = r_k - \alpha_k A p_k$   
     $\beta_{k+1} = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}$   
     $p_{k+1} = r_{k+1} + \beta_k p_k.$   
  end for  
end function
```

Next, we introduce Nonlinear Conjugate Gradient Method. Consider a convex minimization problem for $f : X \rightarrow \mathbf{R}$:

$$\text{minimize } f(x), \tag{5.1.2}$$

over $x \in \mathcal{C}$, where \mathcal{C} is a closed, nonempty, convex subset of X .

Repeat process below until convergence:

1. Update direction by

$$\Delta x_n = -\nabla_x f(x_n);$$

2. Update variable step by, e.g. Fletcher-Reeves Fletcher and Reeves (1964):

$$\beta_n^{\text{FR}} = \frac{\Delta x_n^\top \Delta x_n}{\Delta x_{n-1}^\top \Delta x_{n-1}}; \tag{5.1.3}$$

3. Update direction by

$$s_n = \Delta x_n + \beta_n s_{n-1};$$

4. Calculate step size by

$$\alpha_n = \operatorname{argmin} f(x_n + \alpha s_n);$$

5. Update solution by

$$x_{n+1} = x_n + \alpha_n s_n.$$

On the 4th step for nonlinear CG, one can use the Preconditioned Conjugate Gradient Method (pcg) in Section 4.1.2. In addition, on step 2, one can upgrade β by Polak–Ribiere formula Polak and Ribiere (1969) since it provides sufficient descent, where

$$\beta_n^{\text{PR}} = \frac{\Delta x_n^\top (\Delta x_n - \Delta x_{n-1})}{\Delta x_{n-1}^\top \Delta x_{n-1}}, \quad (5.1.4)$$

instead of the Fletcher–Reeves formula (5.1.3).

In summary, the conjugate nonlinear convex optimization with Polak-Ribiere is as follows:

1. Direction update:

$$\delta_{k+1} = \begin{cases} -\nabla f(x_k) \\ -\nabla f(x_k) + \chi_k \delta_k, \end{cases} \quad (5.1.5)$$

where χ_k satisfies the (5.1.4), where $p_j = \nabla f(x_j)$, and $\chi_j = \max\{0, \chi_j\}$.

2. $\beta_k = \operatorname{argmin} f(x_{k-1} + \beta_k \delta_k)$.

3. $x_{k+1} = x_k + \beta_k \delta_k$.

For ROM step, we minimize a convex function $f : X \rightarrow \mathbf{R}$ when we consider:

$$\operatorname{minimize} \quad f\left(\sum_k \alpha_k x_k\right) \quad (5.1.6)$$

Or,

$$\begin{aligned} &\operatorname{minimize} \quad \sum_k \alpha_k f(x_k) \\ &\text{subject to} \quad \sum_k \alpha_k x_k \in \mathcal{C}, \end{aligned} \quad (5.1.7)$$

and let

$$x_{m+1} = \sum_{k \leq m} \alpha_k^* x_k.$$

5.2 Conjugate Residual Method

Next, we develop the Nonlinear Conjugate Residual method Hestenes and Stiefel (1952). Consider the equality constrained optimization

$$\begin{aligned} & \text{minimize} && f(y) \\ & \text{subject to} && E(y) = 0. \end{aligned} \tag{5.2.1}$$

The necessary optimality system for $x = (y, \lambda)$ is

$$F(y, \lambda) = \begin{pmatrix} f'(y) + E'(y)^* \lambda \\ E(y) \end{pmatrix} = 0.$$

Multiplication of vector by the Jacobian F' can be approximated by the difference of two residual vectors by

$$\frac{F(x_k + t y) - F(x_k)}{t} \sim A y \text{ with } t |y| \sim 1.$$

Thus, we have the nonlinear version of the conjugate residual method:

Nonlinear Conjugate Residual method One can apply the conjugate residual method for

$$\text{minimize} \quad |F(\sum_k \beta_k r_k)|^2. \tag{5.2.2}$$

Let P^{-1} is the preconditioning of F .

- Calculate s_k for approximating Ar_k by

$$s_k = \frac{F(x_k + t r_k) - F(x_k)}{t}, \quad t = \frac{1}{|r_k|}.$$

- Update the direction by

$$p_k = P^{-1}r_k - \beta_k p_{k-1}, \quad \beta_k = \frac{(s_k, r_k)}{(s_{k-1}, r_{k-1})}.$$

- Calculate

$$q_k = s_k - \beta_k q_{k-1}.$$

- Update the solution by

$$x_{k+1} = x_k + \alpha_k p_k, \quad \alpha_k = \frac{(s_k, r_k)}{(q_k, P^{-1}q_k)}.$$

- Calculate the residual

$$r_{k+1} = F(x_{k+1}).$$

ROM step gives you the convergence based on the criterion: $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbf{R}^m$ minimizes

$$|F(\sum_{k \leq m} \alpha_k x_k)|^2.$$

With optimizer $\alpha^* = (\alpha_1^*, \dots, \alpha_m^*) \in \mathbf{R}^m$, we have:

$$x_{m+1} = \sum_{k \leq m} \alpha_k^* x_k.$$

5.3 Proximal Gradient Algorithm

We interpret Proximal Gradient Algorithm Parikh and Boyd (2014) as a fixed point iteration. Suppose $g : \mathbf{R}^n \rightarrow \mathbf{R}$ is convex, and continuously differentiable, $h : \mathbf{R}^n \rightarrow \mathbf{R}$ closed proper and convex. Let $f : \mathbf{R}^n \rightarrow \mathbf{R}$ be defined by $f(x) = g(x) + h(x)$. We solve

$$\text{minimize } g(x) + h(x), \tag{5.3.1}$$

over $x \in X$. Let x^* be a solution to (5.3.1), then:

$$0 \in \lambda(\nabla g(x^*) + \partial h(x^*)).$$

Then, we have:

$$0 \in \lambda(\nabla g(x^*) + \partial h(x^*)) - x^* + x^*.$$

Therefore,

$$(I + \lambda \nabla g)(x^*) \in (I + \lambda \partial h)(x^*).$$

By the relation of prox and operator ∇f , we have:

$$x^* = \text{prox}_{\lambda^k h}(x^* - \lambda^k \nabla g(x^*)).$$

Therefore, given $x_0 \in X$, one has

$$x_{k+1} = x_k - t_k G_k(x_k),$$

where

$$G_k(x) = \frac{1}{t} [x - \text{prox}_{th}(x - t \nabla g(x))],$$

and

$$\text{prox}_{th} = (I + t \partial h)^{-1}.$$

ROM step update: we minimize

$$g\left(\sum_{k \leq m} \alpha_k x_k\right) + h\left(\sum_{k \leq m} \alpha_k x_k\right),$$

over α . With optimizer $\alpha^* = (\alpha_1^*, \dots, \alpha_m^*) \in \mathbf{R}^m$, we have:

$$x_{m+1} = \sum_{k \leq m} \alpha_k^* x_k.$$

Chapter 6

Convergence

We develop arguments about convergence acceleration for fixed point iterate.

6.1 Linear Function

We develop ROM convergence for a linear function. Let $A \in \mathbf{R}^{n \times n}$, let $g : X \rightarrow X$ be a linear function defined by $g(x) = b - Ax$, and let the function $F : x \in X \rightarrow F(x) \in X$ be defined by $F(x) = x - g(x)$. Recall that Anderson is the same as ROM in Section 3.1 if the residual function F is linear. In particular, we have:

$$x_{m+1} - g(x_{m+1}) = \sum_{k \leq m} \alpha_k^* r_k = r_{m+1}^*. \quad (6.1.1)$$

If the sequence $\{x_n\}$ is uniformly bounded, then by the continuity of g , the sequence $\{g(x_n)\}$ is also uniformly bounded. Consequently, there exists a subsequence pair $\{(x_{n_k}, r_{n_k})\} \in X \times X$ and $(x^*, r^*) \in X \times X$ so that $(x_{n_k}, r_{n_k}) \rightarrow (x^*, r^*)$, where $r_k = x_k - g(x_k)$, and $r^* = x^* - g(x^*)$.

For the linear case, it is equivalent to the Krylov subspace method. Thus, its performance depends on the conditioning of the $\{r_k\}$. If $\{r_k\}$ is nearly singular in finite step, the Anderson method results in slow convergence $r_k \rightarrow r^*$. If $\{r_k\}$ is linearly dependent,

then we may terminate algorithm in finite step.

For the case A is nearly singular, if condition number of A is too large, the ROM method results in slow convergence $r_k \rightarrow r^*$, and r^* is not necessarily equal to 0. For singular A , Brown and Walker (1997) give conditions under which the GMRES iterates converge safely to a least-squares solution or to the pseudoinverse solution.

If the matrix A is singular, by singular value decomposition, there exists U, S, V such that $A = USV^\top$. In particular, one can write V as $V = V_1 + V_2$, so that

$$|USV_1| \approx 0, \quad \min |USV_2x - b| > 0.$$

Conclusion: If $\dim(X)$ is finite, and condition A is not too large, then $\text{Span}\{r_i\}$ stay the same. Consequently, $(x_k, r_k) \rightarrow (x^*, r^*)$.

6.2 Nonlinear Function

In this section, we bring a bound for $\{x_k\}$ and discuss how Anderson-type acceleration can help ROM in practice. In order to bound the update $\{x_k\}$, we perform the convex optimization

$$\begin{aligned} & \text{minimize} && |F(\sum_{k \leq i} \alpha_k x_k)|^2 \\ & \text{subject to} && \alpha_k \geq 0, \\ & && \sum_k \alpha_k = 1. \end{aligned} \tag{6.2.1}$$

Then, with $x_{i+1} = \sum_{k \leq i} \alpha_k x_k$ and an arbitrary $\bar{x} \in X$, we have:

$$|x_{i+1} - \bar{x}| \leq \alpha_k \max_{k \leq i} |x_k - \bar{x}| \leq \max_{k \leq i} |x_k - \bar{x}|,$$

which provides the uniform bound of $\{x_k\}$.

We now show a different update scheme will provide better convergence result. Let

the update scheme be

$$x_{m+1} = \alpha_m^* g(x_m) + \sum_{k < m} \alpha_k^* x_k.$$

Then, we have the the following estimate:

$$\begin{aligned} |x_{m+1} - \bar{x}| &= |\alpha_m^* g(x_m) - \alpha_m^* \bar{x} + \sum_{k < m} \alpha_k^* x_k - \sum_{k < m} \alpha_k^* \bar{x}| \\ &\leq \alpha_m^* |g(x_m) - \bar{x}| + \sum_{k < m} \alpha_k^* |x_k - \bar{x}|. \end{aligned}$$

When $\{x_k\}$ descents, we have $|g(x_m) - \bar{x}| \leq |x_k - \bar{x}|$ for each $k \in \{1, \dots, m\}$, then we conclude that we have obtained a lower upper bound than the previous result.

If the Anderson solution that minimizes

$$\begin{aligned} &\text{minimize} && \left| \sum_{k \leq i} \alpha_k r_k \right|^2 \\ &\text{subject to} && \sum_{k \leq i} \alpha_k = 1. \end{aligned} \tag{6.2.2}$$

is a decent solution to ROM (6.2.1), then accept the step. Otherwise, we apply a decent (gradient) method to find a local minimum of (6.2.1) as in Section 6.3.

6.3 Sequential Method

In this section, we develop sequential method. In order to find a local minimizer of ROM, we initialize α by

$$\text{minimize} \quad \left| \sum_k \alpha_k F(x_k) \right|^2, \tag{6.3.1}$$

over α , and let

$$\bar{x} = \sum_k \bar{\alpha}_k x_k,$$

where $\bar{\alpha}$ is the optimal solution to Anderson solution (6.3.1).

Then, we use the Gauss-Newton method in Section 2.3.3 at \bar{x} to solve ROM . We

have one rank update with

$$d_n = F(x_n),$$

and we obtain the sequential update

$$x_{n+1} - \bar{x} = \beta_n d_n, \tag{6.3.2}$$

where $\beta_n \in \mathbf{R}$ minimizes

$$\text{minimize } |F(\bar{x}) + \beta_n d_n|^2 \tag{6.3.3}$$

and

$$x_{n+1} = \bar{x} + \beta_n F(x_n)$$

Here,

$$F'(\bar{x})d_n \sim Jd_n = \frac{F(\bar{x} + t d_n) - F(\bar{x})}{t}, \quad t > 0.$$

Then, we use the Gauss-Newton to solve

$$\text{minimize } |\beta_n Jd_n + F(\bar{x})|^2, \tag{6.3.4}$$

over β_n , and we have an approximate solution

$$\beta_n = -\frac{(F(\bar{x}), Jd_n)}{(Jd_n, Jd_n)}.$$

6.4 Strong Convergence under Coercivity

Coercivity assumption:

$$(x - y - (g(x) - g(y)), x - y) \geq \omega|x - y|^2.$$

Let x^* be optimal solution so that $|x^* - g(x^*)| \leq |x - g(x)|$, for all $x \in X$. Therefore,

$$((x_n - g(x_n)) - (x^* - g(x^*)), x - x^*) = (r_n - r^*, x - x^*).$$

By assumption, we have

$$(r_n - r^*, x_n - x^*) \geq \omega |x_n - x^*|^2.$$

Since $r_n \rightarrow r^*$, we have

$$|x_n - x^*| \rightarrow 0,$$

as $n \rightarrow \infty$.

Example 6.4.1. Operator Equation We consider the operator equation of the form: $Lx + f(x) = 0$ where L is bounded invertible and f is Lipschitz on X . It is formulated as a fixed point problem

$$F(x) = x - L^{-1}f(x) = 0,$$

and the variable fixed point

$$x_{n+1} = x_n - \alpha_n F(x_n).$$

Assume

$$(Lx - f(x) - (Ly - f(y)), x - y) \geq \omega |x - y|^2.$$

Then, for some $c > 0$,

$$\omega |x_i - x^*| \leq c |r_i - r^*|.$$

For example,

$$L = -\Delta, \quad f(u) = -\nabla \cdot (\phi(|\nabla u|^2) \nabla u),$$

where ϕ is convex. with $X = H_0^1(\Omega)$ Thus,

$$(Lu + f(u) - (Lv + f(v)), u - v)_{L^2} \geq (L(u - v), u - v)_{L^2}.$$

and thus,

$$|x_n - x^*|_X \leq |r_n - r^*|_{X^*}.$$

Chapter 7

Applications in Scientific Computing

We test ROM/Anderson type method for solving equations motivated from science and engineering. We incorporate Anderson type method to drive the system to a desired state. We do so by solving a matrix Riccati equation, and we gradually maximize the positive coefficient to the quadratic term until no solution exists, or converges slowly. We find that Anderson type contributes to greatly enhance the level of performance. On the other side, most scientific computing can be modeled as a Saddle Point problem. We focus on solving linear steady state Navier Stokes like equation, with homogeneous boundary condition on the velocity component. We use finite difference scheme to have a squared matrix A , with a velocity component and a pressure component, and vector b . Our computation suggests that Anderson type method accelerates convergence very much. In addition to Anderson type method, we refine Anderson type acceleration with vector subspace method. For linear systems in general, we can select A and b as we want, especially for large scale squared matrix A .

7.1 Nonlinear System for Matrix Riccati Equation

In this section, we demonstrate the use of Anderson map as a contraction continuation function on a matrix Riccati equation. A matrix riccati equation behaves like Navier

Stokes equation. Specifically, we consider a matrix Riccati equation of the form

$$\dot{u} = -b u u + Q, \tag{7.1.1}$$

where u is an unknown matrix of size $n \times n$, Q is a symmetric positive matrix of $n \times n$, and $b > 0$ is the internal variable for the map. It is a nonlinear system of quadratic equation and is parameter dependent with parameter b . Here, $b > 0$ is like $1/Re$ in the Navier Stokes system of equations with quadratic nonlinearity, where Re denotes the Reynolds number. So, there are many issues associated with it. For example, large b significantly magnifies nonlinearity, which brings challenges to find the matrix equation solution. Therefore, we are concerned about developing computational methods to solve a matrix Riccati equation. We solve (7.1.1) by performing contraction composition on $F(u) = -b u u + Q$ and make sure F is contractive for large positive scalar b . We use fixed point iterate for u . Start with small b , we assume that the map is contractive. We see that large b stops contraction after contraction range. We use Anderson map with Method of continuation to enlarge b . Anderson does better job with relatively large b , while the standard case not work for large b . In summary, Anderson works for solving parameter dependent nonlinear equations with enlarged b . Anderson did magic.

As a parallel side of experiment, we enlarge n of the matrix size u and shall see that Anderson map still helps to enlarge b with method of continuation.

Example 7.1.1. Numerical Continuation on b . We show the effect of numerical continuation on (7.1.1) through the composition of contractive maps by internal and external update. We start with a small $b > 0$.

1. External update: We update external variable u with a given iterative scheme until the mapping is shown contractive. If fails, we terminate the algorithm.
2. Internal update: We heuristically upgrade internal variable b as a backward update and go back to the previous step.

We compare the value of b to see the effect of standard Banach iterate and the incorporation of Anderson type acceleration into Banach iterate. We first let $n = 10$, and randomly generate a symmetric and positive definite (SPD) matrix Q . Then, we pick an

initial $b > 0$ such that contraction can happen. We now describe sensitivity for the coefficient b . For each given $b \in \mathbf{R}_+$ with different SPD Q (even of the same dimension), the sensitivity range of b for convergence will be different. So, b is an important parameter to tune beyond the contraction constant. We start with a b that is positive yet close to 0, so that the fixed point iterate will converge, to make sure that b is able to grow up. We move b forward so that external update works. Otherwise, we stop the entire algorithm.

We also generalize from the case of $n = 10$ to $n = 200$ to see that Anderson acceleration methods work for the large scale Riccati equation in general.

We first discuss how the algorithm works when u is an arbitrary squared size. We do initialization on a symmetric positive definite matrix Q first to generate the starting iterate u .

1. We generate Q as follows:
 - (a) We make Q a symmetric matrix by $Q = \frac{1}{2}(Q + Q^\top)$.
 - (b) We compute the smallest eigenvalue of Q and add appropriate regularization constant times identity to make sure Q is a positive definite matrix. We make the minimum eigenvalue between 0 and 1.
2. Initialization: With $u = 0$, fix $b \in \mathbf{R}_+$ such that the algorithm may converge, say $b = 0.05$. Now, we look at range of b for which Anderson works.

We now look at contraction mappings to update the external variable u . We first introduce the Banach iterate. Fix $\epsilon > 0$ as tolerance level, upgrade u until $|u_{k+1} - u_k| < \epsilon$. Let

$$u_{k+1} = -b u_k u_k + Q. \tag{7.1.2}$$

One would perform (7.1.2) with a batch of size m to accelerate the convergence.

Now, we observe the use of Nested Anderson map, with step size $m \geq 2$, from Section 3.3.1. We generate u_1 and u_2 with (7.1.2). For $i = 1, 2$, we store residual vectors $r_i = -b u_i u_i + Q - u_i$ and perform vectorization to transform r_i from $n \times n$ matrix to a $n^2 \times 1$ vector, and we store m of vectroized r_i in matrix r as column vectors. With

$B_{ij} = (r_i, r_j)$, we perform (2.4.3) to determine weight and upgrade iterate by (2.4.4) as new basis elements. By taking a linear combination of past iterates, we update the current iterate. We implement successive fixed point iterates with Anderson depth of 2, 3, 4 and printed out $|u_{k+1} - u_k|$ at each iterate to monitor acceleration convergence. At last, we solve for 4 mixing weights and form the final accelerated iterate as restart.

We list number of iterates with $|u_{k+1} - u_k|$ for different algorithms taking to converge on certain values of b by making Table 7.1 for standard Banach iterate with 3 steps, Table 7.2 for Nested Anderson with 4 step, and Table 7.3 for Nested 7-step Anderson. In each table, we first fix a symmetric positive definite matrix Q of size 10×10 , then we upgrade $b \in \mathbf{R}$ and we report the first $|u_{k+1} - u_k|$ below the tolerance level on the indicated iteration amount after the entire loop is finished. Anderson acceleration method works not only for small matrix of size 10×10 but also for the large scale Riccati equation in general. We illustrate this by considering u as a matrix of 200×200 unknowns to solve, and acceleration methods works quite well, which shows the applicability of Anderson for large scale problems for general pattern in practices. Table 7.4 shows that the Anderson method still works with the method of continuation for a large scale nonlinear system of equation when u is of size 200×200 .

We stop as long as $|u_{k+1} - u_k|$ fails to converge to 0. We observe that the original fixed point iterate with large b (say 0.5) is never contractive. Employing Nested Anderson map makes contraction lasts longer, and Anderson does not require fixed point map contraction for the original function to hold. Therefore, we use the method of continuation that Anderson map accelerates contraction.

b	Iterate	$ u_{k+1} - u_k $
0.05	2	1.4918×10^{-10}
0.1	11	8.7325×10^{-7}

Table 7.1: Contraction continuation with standard Banach Iterate with 3 steps ($n = 10$).

Remark 7.1.1. *Each time with a different Q , all tables would differ. A general pattern is that the larger b becomes, the more iterate it will take to get to the desired level of convergence. We do regularization to guarantee the existence of weight coefficients, shown*

b	Iterate	$ u_{k+1} - u_k $
0.05	3	4.6539×10^{-7}
0.1	3	1.1879×10^{-7}
0.5	6	2.0847×10^{-7}
1	6	6.9332×10^{-7}
1.5	8	8.9559×10^{-7}

Table 7.2: Contraction continuation with Nested 4 step Anderson acceleration ($n = 10$).

b	Iterate	$ u_{k+1} - u_k $
1	4	1.4598×10^{-8}
2	5	1.0643×10^{-7}
5	7	8.2790×10^{-10}
10	9	1.7867×10^{-9}
20	12	2.8622×10^{-8}
40	19	3.5862×10^{-9}
45	18	7.8087×10^{-8}
47.5	17	2.7625×10^{-9}

Table 7.3: Contraction continuation with Nested 7 step Anderson acceleration ($n = 10$).

in Section 2.4, if necessary.

Remark 7.1.2. *Since it is possible that the algorithm will converge to the desired tolerance level before the last step of iterate, it is possible to have redundant steps. We ignore the effects of redundant steps on the last iteration, whenever difference $|u_{k+1} - u_k|$ hits below the reaches the stopping criteria in some stage.*

We bring summary of the section with the following observations with unknown matrix u of size 10×10 :

1. Observation 1: We use standard fixed point iterate: $\dot{u} = -b u u + Q$. Small b implies a fixed point contraction will happen, and Anderson acceleration help accelerate convergence fairly very much. For example, in the case when $b = 0.1$, the standard case needs 9 more iterate than the Nested case.
2. Observation 2: Test of Anderson for b increment suggests that contraction can happen beyond the contraction range for the original equation.

b	Iterate	$ u_{k+1} - u_k $
0.05	5	1.4765×10^{-8}
0.075	3	4.2009×10^{-7}
0.1	4	2.6738×10^{-7}
0.125	4	2.2680×10^{-8}
0.15	4	3.0553×10^{-7}
0.20	4	9.1989×10^{-7}
0.25	4	4.0601×10^{-7}
0.30	9	4.2750×10^{-7}

Table 7.4: Contraction continuation with Nested 7 step Anderson acceleration ($n = 200$).

3. Observation 3: Given a large b , Anderson stop converging, therefore we adapt the idea of continuation: we enlarge b sequentially, then we apply acceleration. As shown in Table 7.2 and Table 7.3, Anderson shows contraction with continuation further.

At last, we refined matrix u of size 200×200 unknowns for Riccati model and the same observation from holds as shown in 7.4, which illustrates that Anderson map works for a nonlinear system in general. Therefore, we have remedied contraction map with Anderson to make contraction work work for nonlinear system.

In summary, Anderson and method of continuation work for nonlinear systems. We may use a different method to remedy small contraction range of b when using the standard fixed point method.

7.2 Saddle Point Problems

In this section, we focus on large scale semi ill-posed linear system of equations $Ax = b$, specifically the Saddle Point Problem, with a given $n \in \mathbf{Z}_+$, and $A \in \mathbf{R}^{n \times n}$, and $b \in \mathbf{R}^n$. We solve it by Cauchy's method and its variant, along with ROM, and Sampled ROM. We report the relative residual for each algorithm. We specify Stokes-like systems as an example for Saddle Point Problem.

7.2.1 Saddle Point Problem 1

Example 7.2.1. Recall Example 4.3.1 in Section 4.3.2 in the case of Stokes system, given D as a symmetric positive definite matrix, we consider the saddle point system given by a specified (D, E) as a constrained minimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(x, Dx) - (f, x) \\ \text{subject to} \quad & Ex = g. \end{aligned} \tag{7.2.1}$$

We consider a divergence free case, i.e. the case when $g = 0$. With simplification on necessary optimality condition, we obtain $Ax = b$:

$$\underbrace{\begin{bmatrix} D & E^\top \\ E & \mathbf{0} \end{bmatrix}}_A \underbrace{\begin{bmatrix} u \\ p \end{bmatrix}}_x = \underbrace{\begin{bmatrix} f \\ 0 \end{bmatrix}}_b. \tag{7.2.2}$$

We consider homogeneous boundary condition for u . Let $n \in \mathbf{Z}_+$ be the dimension of for 2nd order central difference matrix T_n given by

$$T_n = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}. \tag{7.2.3}$$

We use the homogeneous boundary condition on (D, E) . We first model D by:

$$\mathbf{D} = I_n \otimes T_n + T_n \otimes I_n,$$

where I_n is the identity matrix, and T_n , defined by (7.2.3), is the central difference approximation of $-u_{xx}$ with homogeneous boundary condition on u of size $n^2 \times n^2$. We do not consider the pressure constraint. D is a positive definite matrix with $\min(\text{eig}(D)) = 0.1620$. Here D gives energy for minimization, and E gives the divergence free constraint $Ex = 0$ for u_x with homogeneous boundary condition on u .

Now, we construct E by:

$$\mathbf{E} = I_n \otimes C_n + C_n \otimes I_n,$$

where we set the first order central difference matrix C_n of size $n^2 \times n^2$ by:

$$C_n = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \cdots & 0 \\ 0 & -1 & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix}. \quad (7.2.4)$$

That is, given different condition with appropriately defined the corresponding (D, E) . For $n = 10$, we have the matrix A of size $2n^2 \times 2n^2$ in (7.2.2). Since $\min(\text{eig}(A)) < 0$ and $\max(\text{eig}(A)) > 0$, A is an indefinite matrix.

We discuss experiment setup. We set the regularization constant 10^{-30} and $m = 15$ as the amount of loop to go over. For ROM, we store m residual vectors into a residual matrix, and perform Anderson acceleration; For Sampled ROM, we store m vectors into a residual matrix to perform the first round of Anderson acceleration, and their resulting performance m vectors into another residual vector to perform another round of Anderson acceleration.

We plot relative residual for all iterate and we shall see that: ROM accelerates convergence, when performing a one-direction update (c.f. Figure 7.1), or a two direction update (c.f. Figure 7.2). As suggested from pictures, we see that further improvement of ROM can be done through the Sampled ROM. In addition, the performance with a two directions' update as a multi step subspace update will significantly improve the performance of a one direction update.

Remark 7.2.1. *For general boundary condition (e.g. von-Neumann condition), we adjust D and E appropriately.*

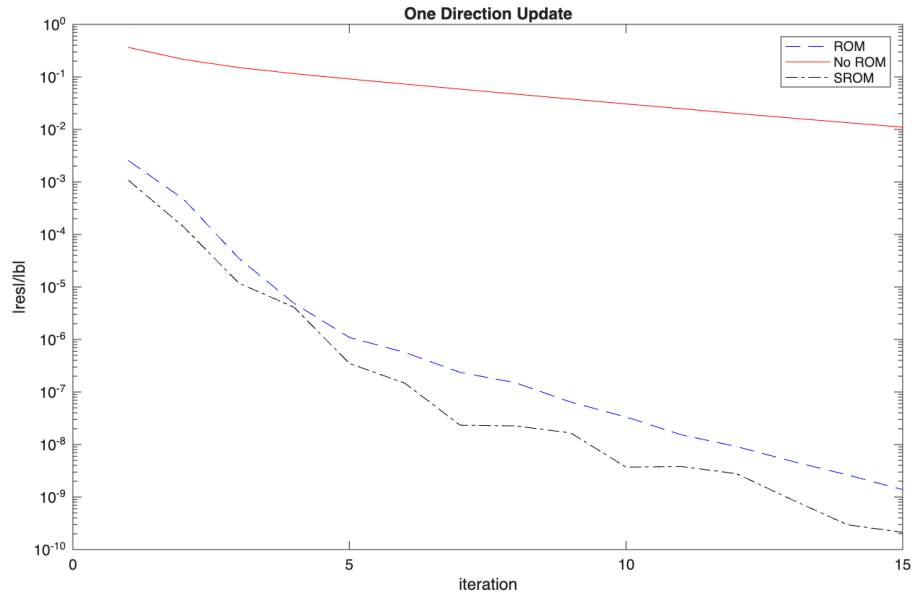


Figure 7.1: Saddle Point Problem 1 (One direction update: $r = b - Ax$): comparison of standard case, accelerated case, and sampled case.

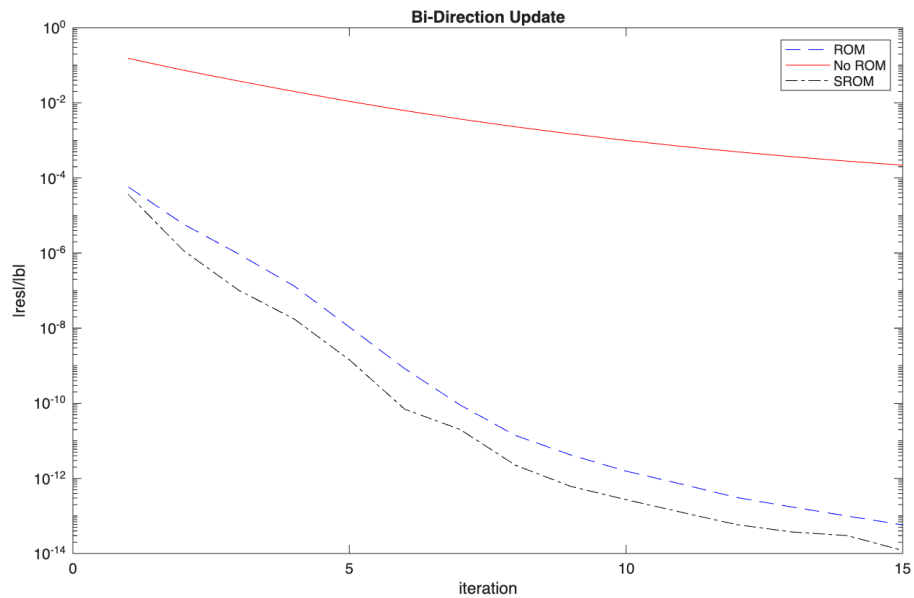


Figure 7.2: Saddle Point Problem 1 (Two directions' update: $r = b - Ax, Ar$): comparison of standard case, accelerated case, and sampled case.

Remark 7.2.2. *We generalize Example 7.2.1 as follows:*

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(u, Du) - (f, u) + \frac{1}{2}(Dv, v) - (g, v) \\ \text{subject to} \quad & E_1u + E_2v = h. \end{aligned} \tag{7.2.5}$$

We use the homogeneous boundary condition and still consider the divergence free condition to hold by setting $h = 0$ to obtain $E_1u + E_2v = 0$. Consequently, we have:

$$\begin{bmatrix} D & 0 & E_1^\top \\ 0 & D & E_2^\top \\ E_1 & E_2 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \\ 0 \end{bmatrix}. \tag{7.2.6}$$

With D and C_n same as in Example 7.2.1 and with $E_1 = C_n \otimes I_n, E_2 = I_n \otimes C_n$, our experiment found that same conclusion from Example 7.2.1 still holds.

7.2.2 Saddle Point Problem 2

In this section, we use a variant of Cauchy step size update for solving $Ax = b$. We generate an approximated matrix \bar{A} for A . In one direction update, we use an alternative step size (4.1.10) than Cauchy step (4.1.6) as a variable step size update. By using \bar{A} , we sacrifice optimality in each step in exchange for efficiency. Cauchy method (4.1.6) gives optimal step size of the steepest descent, and (4.1.10) gives a cheaper surrogate that approximates optimal step size of descent direction but allows for faster computation with cost reduction. We iterate with \bar{A} and residual from the original equation $r = b - Ax$ to compute variable step size update in (4.1.10), then correct with a ROM step that uses a richer space of prior residuals to better approximate the true solution trajectory. For example, in saddle-point systems (like Stokes or Navier–Stokes), solving with A is hard due to null space or ill-conditioning.

We shall treat \bar{A} as a preconditioner of A , but we make use of \bar{A} as a preconditioner in an unconventional way without inverting matrix \bar{A} . We use \bar{A} to approximate the action of A in computing variable step size update, which improves computational efficiency in place of exact information from A . In other words, we have applied a cheap surrogate for A to implicitly modify the step size of a given search direction, which matches the

goal of applying a preconditioner with cheaper replacement for A to modify the search direction implicitly. We use Anderson to accelerate our proposed method.

In terms of two directions' subspace update implementation, one considers (4.2.8) than the Cauchy step (4.2.6). With exactly the same argument from one direction update, we find that the same conclusion from 1 direction subspace update also hold for 2 directions subspace update.

Example 7.2.2. Consider the saddle point problem

$$A = \begin{pmatrix} D + \delta F & E^* \\ E & 0 \end{pmatrix}, \quad (7.2.7)$$

with the same D , E as in Example 7.2.1. We set the perturbation δF on D by:

$$\delta F = \mathbf{1}^\top E,$$

where $\mathbf{1}$ is the matrix of all ones.

For \bar{A} , we have:

$$\bar{A} = \begin{pmatrix} D & E^* \\ E & 0 \end{pmatrix}. \quad (7.2.8)$$

Here, A is a low rank perturbation of \bar{A} , it is easier to work with \bar{A} to reduce computational cost. We shall see that the update with (4.1.10) and (4.2.8) are nearly optimal due to low rank perturbation between A and \bar{A} .

We plot relative residual and we shall see that: ROM speeds up the convergence, when performing a one-direction update (c.f. Figure 7.3), or a two direction update (c.f. Figure 7.4). Further improvement can be done through the Sampled ROM. In addition, the performance with a two directions' update will boost the performance of a one direction update.

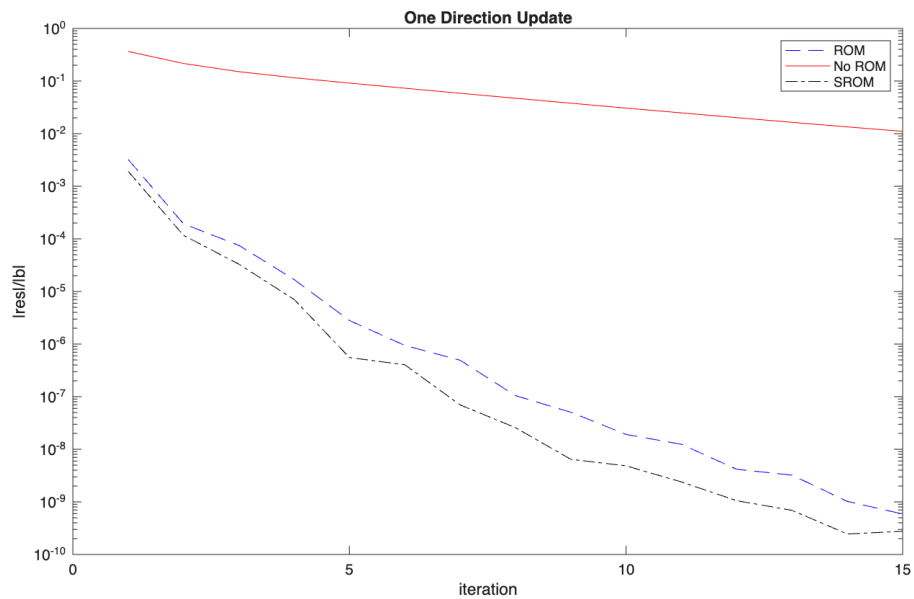


Figure 7.3: Saddle Point Problem 2 (One direction update $r = b - Ax$): comparison of standard case, accelerated case, and sampled case.

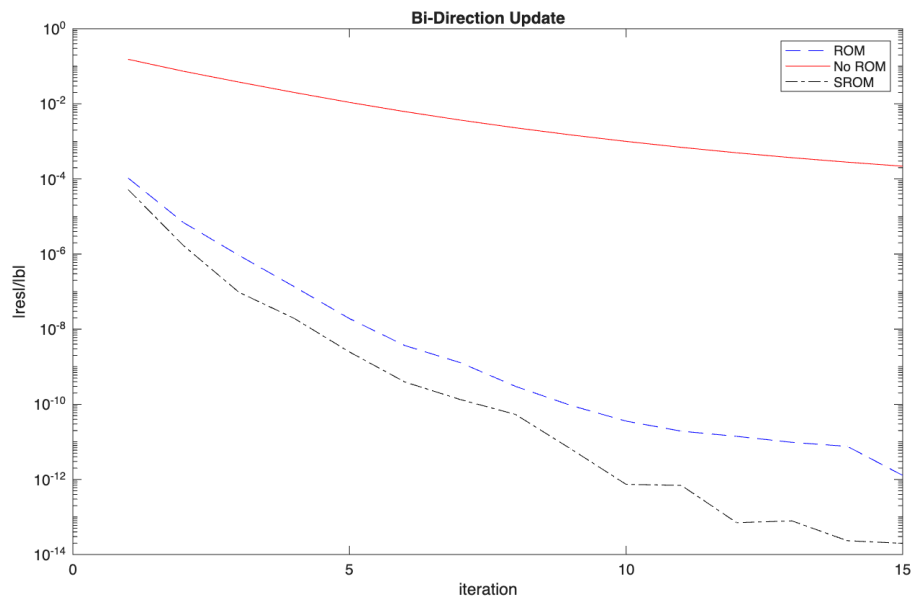


Figure 7.4: Saddle Point Problem 2 (Two directions' update $r = b - Ax, \bar{A}r$): comparison of standard case, accelerated case, and sampled case.

Chapter 8

Applications in Neural Network

In this chapter, we test Anderson type acceleration method for neural network optimization, i.e. we build an input-output model through Deep Neural Network (DNN) and Convolution Neural Network (CNN) by incorporating Anderson type acceleration.

Anderson Acceleration (AA), also called Anderson mixing, is a method for the acceleration of the convergence rate of fixed-point iterations. It works by combining the most recent iterates and update steps in a fixed-point iteration to improve the convergence properties of the sequence. AA is particularly effective in scenarios where the fixed-point iteration might be slow or unstable, leading to faster convergence and improved solution quality. The incorporation of Anderson acceleration works quite remarkably.

Neural network is named and motivated by neural networks mimic the human brain by using interconnected nodes called neurons to process data and learn patterns. They create connections and adjust the weights of those connections through a process called training, which involves feeding the network large datasets. This training allows the network to refine its weights to minimize errors between its predictions and actual values, similar to how humans learn to recognize patterns.

We use Deep Neural Network (DNN) to formulate the forward path and we use category (digit) classification accuracy level as the performance. It results in optimization in the neural network. The stochastic (mini-batch) gradient decent method is applied as

the learning method. Anderson type acceleration method optimizes the learning and results in accurate weights in the network design to improve prediction accuracy.

We first formulate a Neural Network with a fullyconnected layer (DNN) as an input-output model. We describe the general philosophy for input-output model based on a Neural Network as a forward-backward path algorithm. We first model a forward path. We design DNN map with internal variables (W, b) to connect from input to output. Given (W, b) as weight-bias parameter pair for performing a transformation, we build a neural network transformation ϕ of input vectors $X_0 \in \mathbf{R}^{d_0 \times N}$ to output data $Y \in \mathbf{R}^{d_L \times N}$ by the discrete time dynamics. For the k -th layer, we have:

$$X_{k+1} = \phi_k(W_k X_k + b_k), \quad X_0 = A, Y = H(X_L), \quad (8.0.1)$$

where the label is $X_k \in \mathbf{R}^{d_{k-1} \times N}$, ϕ_k is an activation function such as $\text{Relu} : x \rightarrow \max(0, x)$, N is the sample size, L is the number of the layers, and d_k is dimension of X_k for a fixed sample, in the k -th layer with a given sample. In addition, the weight matrices $W_k \in \mathbf{R}^{d_k \times d_{k-1}}$ and the bias vectors $b_k \in \mathbf{R}^{d_k \times N}$ are the internal variables to be determined. Finally, H is the target constraint map. For example, one can let the target constraint map H be the softmax function (8.0.2) that transforms X_L to the predicted probability distribution that gives prediction probability $p = H(X_L)$ of each label.

Next, we formulate ideas of a forward path for a convolution neural network. One enriches input data by applying a given amount of filters of certain size to each sample. Convolution Neural Network (CNN) Fukushima (1969) LeCun et al. (1990) transfers images from $A \rightarrow B$ first by convolution and then $B \rightarrow Y$, where B has much richer information than A , and $B \rightarrow Y$ is performed by a fully-connected layer. A fully-connected layer is a single layer DNN. The data compression and filtered decomposition are the key aspects of convolution operation to make a convolution neural network work. Recall from DNN that we set (W, b) as design parameters. We can use the same idea in Convolution Neural Network (CNN) with (W, b) as design parameters, with activation function as RELU, and construct the least square criterion. We borrow statement from ‘‘one hot’’ to make sure that we can do least square formulation. We can adapt it as a saddle point problem, but due to large scale of the given matrix, we do optimization with mini batches. We use standard descent in terms of batch. We repeat with non-overlapping mini-batches.

It can be a randomly shuffled sample for each epoch, or in fixed order for each epoch. Then, we apply epoch-wise iterate, and we observe slow convergence in general. Here, we use Anderson method to accelerate convergence. Given general design with residual, we use a sequence of design from a given amount of epochs (4 in our case) to form total residual minimization over weights, and do Anderson update of the current solution with optimal weight. We formulate basis with 4 epochs and restart. We show details of stochastic gradient in Section 8.1 and show in Sections 8.2 and 8.3 that Anderson improves a lot. For our experiments of DNN and CNN, we define the design parameters and cost function. One can apply Anderson acceleration on an arbitrary given Neural Network (like U-Net, Denet from before) by carefully identify the design parameters from Neural Network and cost function.

We now introduce some generalization to design of a Neural Network formulation for a forward path, beyond the DNN and CNN formulation. For example, on an image segmentation task, U-Net Ronneberger et al. (2015) is for the segment transfer. A U-Net has 2 parts: encoder part, and decoder part. The encoder part uses convolution layers and maxpooling to minimize the training loss for mini-batch and compress image size to increase channel and extract features from training data. Afterwards, the decoder part uses the features extracted from the encoder part to restore the original picture. Other model based network such as DEnet Tychsen-Smith and Petersson (2017), Voltera net Marmarelis and Zhao (1997) and mass transportation can be formulated similar to DNN as forward-path models, e.g. prediction. Applications of the neural network formulation and Anderson type method to inverse medium and material design and optimal control as the underline model is elliptic type. Model based method (PINN) uses the neural network basis elements with the internal design variables to represent solutions and the least squares method to equate the model equations. Thus, the optimization is carried over the internal variables (W, b) . For example, in Ito et al. (2021) we apply to solve the Hamilton-Jacobi equation for the game theory we accurately compute the feedback law via a neural network function.

After formulating a Neural network, we can optimize the design of internal variables through the backward propagation by internal variables' subspace update for an optimization problem. In the case of category (digit) classification, first fix a sample. Let Y be the actual probability distribution for each outcome to happen, with the actual

outcome y . In general, the constraint map H depends on the cost criterion and defines performance of the network. We now provide a least squares formulation:

$$\mathcal{L}(y, x_L) = \frac{1}{2N} |H(x_L) - Y|^2, \text{ and } H(z) = \frac{1}{\sum_{j=1}^{n_L} e^{z_j}} (e^{z_1}, \dots, e^{z_{n_L}}), \quad (8.0.2)$$

where we vectorize the actual result y with “one-hot” to Y : We represent categorical data as binary vectors Y where only one element in the vector is “hot” (set to 1), indicating the presence of a specific category, and all other elements are “cold” (set to 0). This technique is used to convert categorical variables into a numerical format suitable for use in machine learning algorithms that require numerical input. Another common loss function \mathcal{L} to compare predicted output p_k with the ground truth y_k as binary masks (pixel value is 1 when pixel contains object) is the categorical cross-entropy loss:

$$\mathcal{L}(y, p) = - \sum_{k=1}^{n_L} y_k \odot \ln(p_k), \quad (8.0.3)$$

where p_k is the predicted probability of class k by a Neural Network, and y_k is the true probability of class k . In the case of single label, we call (8.0.3) as a binary cross-entropy map.

We put the Neural Network formulation of input-output model into more generalized context. First, one may put $\theta = (W, b)$ as the internal variables. We concern to optimize the design for our internal variables when we train the Neural Network. Since our network of concern has at least 1 layer layer and a sufficient number of neurons, Universal Approximation Theorem Hornik et al. (1989) suggests that one can model DNN map as a composition of a sequence of Neural Networks. In general, the model-based transport map T_θ can formulate the least squares problem for the transport problem with given input and output pair (g_0, g_1) :

$$\text{minimize} \quad |T_\theta(g_0) - g_1|^2 + \Theta(\theta), \quad (8.0.4)$$

over θ , where T_θ is the transport map and $\Theta(\theta)$ is the transport cost. The map represents the tomographic map, e.g. Computer Tomography (CT), Electric Impedance Tomography (EIT), Forward Scattering Tomography, and θ is the corresponding medium. On the other hand, we have DNN modeling of the input-output map T_θ using pairs of input and output

data (g_0, g_1) . Also, the map can represent the Monge-Kantorovich mass transport and the value and mass transport by PDE based model Benamou and Brenier (2000). Thus, DNN is a particular case of the transportation problem as a model free method. That is, the method and algorithms developed in the paper can be applied to the general case (e.g. inverse medium problems).

We develop the Anderson type method that accelerates the learning of the neural network and it uses the optimal assembling, bootstrapping and extrapolating by the optimal combination of the sequence of network designs. We minimize the weighted total sum of the residuals corresponding to the sequence of designs over the weights. We generate the sequence of designs for randomly sorted epochs sequentially. We then apply the Anderson type acceleration. It should decrease the merit function based on the reduced order method (ROM), which defines the performance of the linear combination of the sequence of designs (ROM). Like GMRES, we restart the algorithm with the Anderson update to improve the performance.

We measure the performance by accuracy level as follows: We feed the input X with trained internal variables to get Y . Given Y as prediction for an outcome, one outputs the coordinate location that gives maximal probability as result, and compare it with the original label. If the result matches, then we count. We report accuracy by dividing the counting from the total amount of sample number.

For our test, we use the original MNIST (Modified National Institute of Standards and Technology database) dataset by LeCun and Cortes (2005), which has a training set of 60000 images, and a separate testing set of 10000 images. Served as a typically simple and clean benchmark for evaluating algorithms, MNIST dataset is used for testing proof-of-method. For example, for self testing, one can also use a subset of the training set (say first 5000) for both a training and a testing set. We do this self testing in Section 8.2. We may also do a self test by considering the whole training set. We do this self testing in Section 8.3. One also can consider the whole MNIST training set as training set, and the whole MNIST testing set as testing set. For actual testing, we test our algorithms with this choice. We test a given standard algorithm and its incorporation with Anderson acceleration as an improvement to the given standard algorithm.

In Section 8.1, we first discuss the subspace method as a parallel form of the stochastic

gradient method and mini-batch descent method. We then extend the discussion by the standard backpropagation with non-overlapping mini-batch descent and stochastic gradient decent method for the neural network. By extrapolating results, we discuss Anderson type acceleration formulation. In Section 8.2, we consider using Anderson acceleration on a fully connected network with mini-batch gradient descent; in Section 8.3, we consider using Anderson acceleration on a convolution neural network, with the use of stochastic gradient descent.

8.1 Algorithms of Interest

Throughout all discussions below, we call algorithms, regardless of layer design, with no incorporated acceleration procedure (c.f. Definition 1.2.2) as a **Standard Algorithm**. For a standard algorithm, with loss function formulated as least square problem, one first makes each label as a distribution of class. Then, one condenses an image sample from a high dimension to the dimension of label amount. Eventually, one picks the maximum coordinate as the final predicted result. One counts the correct prediction amount to be divided with total amount of testing sample as amount of accuracy. The backward propagation works by propagating the error (difference between predicted and actual output) backward through the network to update the parameters.

We first introduce mini batch descent in deterministic order and stochastic order, as part of backpropagation. Our method is effective. However, we further improve the prediction performance through Anderson acceleration.

8.1.1 Mini-batch Gradient Descent Method

We first describe a special form of stochastic gradient descent method – mini-batch gradient descent method. We make training order every time the same. Consider the optimization of the form

$$\text{minimize} \quad \sum_{k \in I} f_k(x), \quad (8.1.1)$$

where I is the sample set with size N . Let I_i be a non-overlapped sub-indices of $I = \cup I_i$. We solve

$$\text{minimize} \quad \sum_{k \in I_i} f_k(x) \tag{8.1.2}$$

over the block I_i by the subspace method

$$x_{i+1} = x_i - \sum_{k \in I_i} \beta_k f'_k(x_i),$$

where $\{\beta_k\}$ minimizes

$$\sum_{k \in I_i} f_k(x_i - \sum_{k \in I_i} \beta_k f'_k(x_i))$$

in which x_i is hold on I_i . Or,

$$\begin{aligned} &\text{minimize} \quad \left| \sum_{k \in I_i} \beta_k f'_k(x_i) \right|^2 \\ &\text{subject to} \quad \sum_k \beta_k = 1. \end{aligned} \tag{8.1.3}$$

If we select $\beta_k = \beta$, then

$$\text{minimize} \quad \sum_{k \in I_i} f_k(x_i - \beta \sum_{k \in I_i} f'_k(x_i)), \tag{8.1.4}$$

where

$$\frac{1}{m} \sum_{k \in I_i} f'_k(x_i)$$

is the average of gradients over mini-batch I_i .

Remark 8.1.1. *One may use the average of $\sum_{k \in I} f_k(x)$ as objective function.*

8.1.2 Stochastic Gradient Descent (SGD) Method

Now, we generalize mini-batch gradient descent with a fixed sample order into a random sample order. Stochastic gradient descent is important for the success of a NN optimization. We shuffle the entire sample every time. Recall that the standard gradient method

updates solution x_i by

$$x_{i+1} = x_i - \alpha \sum_{k=1}^N \nabla f_k(x_i)$$

where $\alpha > 0$ is the learning rate and ∇f_k is the gradient. For example, given an input-output sample pair (X_k, y_k) , we convert the index y_k to a vector Y_k to have:

$$f_k = |\psi((W, b), X_k) - Y_k|^2 \text{ over a sample } k,$$

where $\theta = (W, b)$ be the design parameters. In general, a sample size N is large and so is the dimension m of design parameters θ .

We evaluate the gradient by the back projection (adjoint) method in Section 8.1.3. We could, however, train parameters with lower cost by optimizing internal variables with non-overlapping batches from the original sample. We use sequentially proceed on batches by batches after the shuffling for the entire data. Consider the minimization of the form

$$\text{minimize } \sum_{k=1}^b f_k(x), \tag{8.1.5}$$

over samples $k \in \{1, \dots, b\}$, where f_k is the loss functions for sample k , until each sample has been trained. Then, we say that the training has completed an **epoch**. After the end of an epoch, we reshuffle the training set ready for use in the next epoch.

In summary, we have the Stochastic Gradient Descent Algorithm Goodfellow et al. (2016):

8.1.3 Backpropagation Algorithm

We finally launch the **Backpropagation Algorithm** Rumelhart et al. (1986). It extends the (8.0.1), also known as forward pass, with an additional step of internal variables update, commonly referred as training. Given:

- Training data $\{(x_i, y_i)\}_{i=1}^m$, with m as batch size;
- Fixed Learning rate η ;

Algorithm 5 Stochastic Gradient Descent (SGD)

Require: Dataset $\{(x_i, y_i)\}_{i=1}^N$, learning rate $\eta > 0$, initial parameters θ_0 , number of epochs T , loss function \mathcal{L} , forward mapping f .

Ensure: Optimized parameters θ

- 1: Initialize $\theta \leftarrow \theta_0$
 - 2: **for** $t = 1 \cdots T$ **do**
 - 3: random permutation of the dataset
 - 4: **for** (x_i, y_i) in each mini-batch **do**
 - 5: Compute gradient $g \leftarrow \nabla_{\theta} \mathcal{L}(f(x_i; \theta), y_i)$
 - 6: Update parameters $\theta \leftarrow \theta - \eta g$
 - 7: **end for**
 - 8: **end for**
-

- Initialized Weights W_l and biases b_l for layer l with $l \in \mathbf{Z}_+ \cap \mathbf{Z}_{\leq L}$ and L as total number of layers;
- Loss Function \mathcal{L} used to train a Neural Network.

For each training example (x, y) , perform the following steps:

1. Forward Pass: We implement (8.0.1) step-by-step.

$$x_1 = x$$

$$\text{For } l = 2 \text{ to } L : \quad z_l = W_l x_{l-1} + b_l$$

$$x_l = \phi_l(z_l).$$

2. Backward Pass: Using chain rule from calculus, we compute the gradient of the loss function \mathcal{L} over mini-batches with respect to each weight W and bias b to perform gradient descent from Section 8.1.1 and Section 8.1.2:

$$\delta_L = \nabla_{x_L} \mathcal{L}(x_L, y) \odot \phi'(z_L)$$

$$\text{For } l = L - 1 \text{ to } 2 : \quad \delta_l = (W_{l+1})^\top \delta_{l+1} \odot \phi'(z_l),$$

where \odot denotes the Hadamard product of two matrices with the same dimension. That is, given two matrices $A = [a_{ij}] \in \mathbb{R}^{m \times n}$ and $B = [b_{ij}] \in \mathbb{R}^{m \times n}$, their **Hadamard product**

(denoted by $A \odot B$) is defined as:

$$A \odot B = [a_{ij} \cdot b_{ij}] \in \mathbb{R}^{m \times n}.$$

3. Parameters' Update:

$$\begin{aligned} \text{For } l = 2 \text{ to } L : \quad & W_l \leftarrow W_l - \eta \delta_l (x_{l-1})^\top \\ & b_l \leftarrow b_l - \eta \delta_l. \end{aligned}$$

8.1.4 Anderson Acceleration

By extrapolating internal variables from training, we introduce the incorporation of Anderson acceleration into the training for a Neural Network. With a trained NN of epoch i , we forward the testing data X to generate X_L with output $P_i = H(X_L)$, and let $r_i = P_i - Y$ with internal variables $x_i = \{(W_j, b_j)\}_{j=1}^L$, with vectorized actual testing label Y with prediction P_i for epoch i . We apply a sequence of epochs to obtain better design of internal variables $\{(W_j, b_j)\}_{j=1}^L$. After m epochs, we do:

$$\begin{aligned} \text{minimize} \quad & \left| \sum_{i=1}^m \alpha_i r_i \right|^2 \\ \text{subject to} \quad & \sum_i \alpha_i = 1, \end{aligned} \tag{8.1.6}$$

over α . With α^* is the optimal solution to (8.1.6), define the Anderson update

$$x_{m+1} = \sum_{i=1}^m \alpha_i^* x_i.$$

Note that, we increase accuracy by decreasing residual. We measure the level of accuracy by counting the number of right prediction. More specifically, given P as prediction for an outcome, one outputs the coordinate location that gives maximal probability as the predicted result, and subtract it from the original label. If the result is a zero, we count the prediction as an accurate prediction. We output the accuracy level by dividing accurate prediction amount with the total amount of samples for prediction.

8.2 A Fully Connected Neural Network

In this section, we consider a special type of fully connected layer called single-layer neural network that applies a linear transformation, then an activation function, and finally performs classification, with an activation function.

For a fully-connected neural network, the number of parameterizations (W_k, b_k) can be very large and expensive. In order to reduce the complicity without losing the performance, we may constraint weight W by the sparsity and bandwidth by using ℓ^0 sparsity penalization to constrain the number of nonzero elements in W .

$$\text{minimize} \quad |H(x_L) - Y|^2 + \beta|W|_0. \quad (8.2.1)$$

8.2.1 Algorithm Description

In this section, we provide description of the proposed algorithm for DNN. We keep the ordering for all samples same each time. With \mathcal{L} defined in (8.0.2), the algorithm consists of the forward path and the backward projection as:

- Initialization of internal variables and input data scaling with division over maximum number in the training matrix.
- Forward path by DNN: We transport X_0 to Y by

$$X_{k+1} = \phi(W_k X_k + b_k), \quad X_0 = A, \quad Y = H(X_L).$$

with Relu $\phi : x \rightarrow \max(0, x)$ and H is the softmax function defined in (8.0.2).

- Backward projection for the gradient with respect to W and b by

$$\Lambda_L = Y - \bar{Y}, \Lambda_k = \phi'(\Lambda_{k+1})$$

$$gW_k = \phi'(\Lambda_k), gb_k = \Lambda_k.$$

- Solution update by the gradient decent method

$$W_k^+ = W_k - \alpha gW_k, \quad b_k^+ = b_k - \alpha gb_k$$

by a sequence of mini-batches.

Remark 8.2.1. *In our case, we take average with sample size N so that the gradient descent step has become:*

$$W_k^+ = W_k - \frac{\alpha}{N} gW_k, \quad b_k^+ = b_k - \frac{\alpha}{N} gb_k.$$

We perform testing by doing **accuracy measurement** by looking at the image of a vector and see the difference between the predicted value and the actual value. If they are the same, we count; we do not count otherwise. We report the number of accuracy after finishing one epoch before moving onto the next epoch, and stop as long as accuracy gets 100%, or meet stop criterion. For example, we may stop when accuracy level gets above 99.8%, or epoch number gets 1000.

8.2.2 Algorithm Implementation

In this section, we discuss how we implement algorithms through Anderson acceleration. So, the main focus is on neural network architecture, and initialization of internal variables, residual storage for Anderson acceleration matrix.

We discuss dimensions settings as an essential part to the architecture of Neural Network: For layer design, we let $n_0 = 28 \times 28 = 784$ be the image pixel size, let $n_1 = 128$ be the hidden layer size, let $n_2 = 10$ be classification layer size. Now, we discuss the sample and mini batch amount: Let $N = 5000$ be sample size by picking the first 5000 from MNIST training set; Let $m = 64$ be the batch size. For the variable step update, we fix the learning rate $\alpha = 0.01$.

We now talk about data initialization: We regularize the input data by making it a matrix with all entries in between 0 and 1. Since for all (i, j) we have $\max_{i,j}[X_{ij}] = 255$ and $[X_{ij}] \geq 0$, we let $X_0 = \frac{1}{255}X_0$ be our initialization. Afterwards, we switch attention

internal variables. We put weight parameters (W_1, W_2) as random matrices of comfortable dimension, and set (b_1, b_2) as zeros matrices of comfortable dimension. For the discussion of dimension set up, the beginning of Chapter 8 gives dimension for (W_1, W_2) and (b_1, b_2) . We use mini-batch gradient descent to train our Neural Network.

We discuss residual storage for self testing. For the k -th epoch from testing (same as training), we use $R_k(\cdot) = P_k(\cdot) - Y(\cdot)$ with vectorization as residual to be collected for Anderson acceleration, where P_k is the predicted probability distribution, and Y is the true probability distribution for testing (same as training) epoch k . We apply Anderson acceleration into our training and see that it improves prediction accuracy very much.

For self testing, we do **restarted Anderson**: We do Anderson acceleration after collecting residual from 4 epochs, and then update parameters by minimizing total sum of residuals before the restart. To accelerate, we repeat the entire process for m times. Afterwards, to accelerate further, one repeats the above process 4 times. In tables involving self testing, we record $4 \times m$ as epoch number. We also perform this algorithm on CNN in the next section.

8.2.3 Results

We run DNN with 1000 epochs and it achieves 99.64% accuracy.

Epoch	Standard	Anderson
4	57.94%	60.14%
8	60.74%	81.04%
20	86.52%	91.12%
40	87.00%	92.28%
100	91.28%	94.02%
500	97.86%	99.52%
1000	99.64%	99.46%

Table 8.1: DNN Self Test: Accuracy Comparison.

Summary: Table 8.1 explains the performance of Standard vs Anderson acceleration

with self testing data with Anderson improves a lot. Anderson acceleration has robust performance compared to the standard ones. We include more details here:

1. DNN converges slowly to the desirable accurate solution, e.g. in the standard case, achieving accuracy level over 99% requires 1000 epochs.
2. Anderson acceleration method converges monotonically with faster convergence, e.g. with 500 epochs, we achieve near 100% accuracy.
3. DNN can be applied to more general cases. For example, we perform Anderson acceleration on CNN with the full training set as self test, as shown in Table 8.2.

In conclusion, Anderson acceleration works what we predicted, i.e. Anderson can be used to accelerate DNN, as what we assumed, in general.

Remark 8.2.2. *We use Anderson acceleration independent of the learning net, i.e. residual for each design parameters. We use it on a convolution neural network in Section 8.3.*

Remark 8.2.3. *When we consider taking mini-batch and learning from it to make predictions, Anderson accelerates inner loop convergence.*

8.3 Convolution Neural Network (CNN)

In this section, we develop Anderson type acceleration on a CNN (convolution neural network). CNN transfers image $A = X_0$ to B with a reduced dimension by applying convolution and max-pooling operations. Then we connect $B \rightarrow Y$ by a single layer DNN. The forward path and back projection is governed by convolution and adjoint convolution. In addition, we consider performing convolution operations and batchnormalization in our neural network definitions. This simplifies the training process, but significantly increases the complexity of NN structure. We condense with reduced order and make the prediction distribution to distribution. We apply Anderson acceleration with the same goal of increasing the testing accuracy than standard algorithms.

Now, we describe the convolution operations. First, we apply the zero-padding with zeros around the border of image X . The convolution $X^+ \in \mathbf{R}^{n \times n \times c}$ of $X \in \mathbf{R}^{n \times n}$ with

the weight W of size $B = b \times b$ is given by

$$X_{i,j,k}^+ = \sum_{(i',j') \in B} W_{i+i',j+j',k} X_{i',j'} \quad , \quad (8.3.1)$$

where B is a window with size $b \times b$ for the k -th channel (filter) $1 \leq k \leq c$. We may slide indices (i, j) by slide size s so that results in a reduced size array X^+ . Otherwise, the dimension $n \times n \times c$ increases as filter size c results in large storage and increase cost.

We give more interpretation on convolution operation. First, the depth of the output volume is a hyperparameter: it corresponds to the number of filters we use, and each layer will have different input. For example, if the first Convolution Layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color. We will refer to a set of neurons that are all looking at the same region of the input as a depth column (some people also prefer the term fiber). Second, we must specify the stride with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

Afterwards, we discuss **Batch Normalization** Ioffe and Szegedy (2015), a procedure that accelerates and stabilizes training by reducing internal covariance shift. Let d be the amount of pixel. Let B denote a mini-batch of size m , let $x_i = (x_i^1, \dots, x_i^d)$, so that the mean of batch B is $\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$, and $\sigma_B^2 = \frac{1}{m} \sum_i (x_i - \mu_B)^2$. For $k \in \{1, \dots, d\}$, and a given $\epsilon > 0$, we have:

$$\hat{x}_i = \frac{x_i^k - \mu_B^k}{\sqrt{(\sigma_B^k)^2 + \epsilon}}. \quad (8.3.2)$$

Then, one sets internal variables of a Neural Network as offset parameter β and scale parameter γ so that $y_i = \gamma \hat{x}_i + \beta$. In terms of the target, instead of a vector with target classes, we consider to use a vectorized target: binary classification with SVM, or multi-class classification with softmax function as a distribution. A multi-class classification generalizes binary classification when it has more than 2 classes of data for making classification.

8.3.1 Algorithm Description

In this subsection, we switch to our focus **Convolution Neural Network (CNN)** LeCun et al. (1990). A purpose of applying a CNN architecture is to find patterns and features in images to recognize objects, classes, and categories Goodfellow et al. (2016). It requires a few components, which are input data, a filter and a feature map. In CNN, we have the feature detector, also known as a kernel or a filter, to move across the receptive fields of the image, checking if the feature is present. They are important for reducing the dimensions to the desired ones.

1. Convolution Layer:

Given input feature map $X \in \mathbf{R}^{H \times W}$ and kernel $K \in \mathbf{R}^{k_H \times k_W}$, the convolution output at location (i, j) is:

$$Z_{i,j} = (X * K)_{i,j} = \sum_{m=0}^{k_H-1} \sum_{n=0}^{k_W-1} X_{i+m,j+n} K_{m,n}.$$

For multiple channels and filters:

$$Z_k^{(l)} = \phi^{(l)} \left(\sum_{c=1}^C X_c^{(l-1)} * K_{k,c}^{(l)} + b_k^{(l)} \right).$$

where, for each $1 \leq l \leq L$:

- $X_c^{(l-1)}$: input feature map from channel c at layer $l - 1$;
- $K_{k,c}^{(l)}$: kernel connecting input channel c to output channel k at layer l ;
- $b_k^{(l)}$: bias for the k -th filter at layer l ;
- $\phi^{(l)}$: activation function [e.g., ReLU (Rectified Linear Unit)] at layer l .

2. Activation Function: Apply a commonly used activation function, say RELU Householder (1941):

$$\phi(x) = \max(0, x) \quad (\text{ReLU}), \tag{8.3.3}$$

3. Fully Connected Layer: After flattening the feature maps, for $l \in \mathbf{Z}_+ \cap \mathbf{Z}_{\leq L}$, we have:

$$a^{(l)} = \phi^{(l)}(W^{(l)} a^{(l-1)} + b^{(l)}).$$

Typically one uses softmax Bridle (1990a)Bridle (1990b) as the last activation function for classification tasks: Let $j \in \mathbf{Z}_{\geq 1} \cap \mathbf{Z}_{\leq n_L}$, and $i \in \mathbf{Z}_+ \cap \mathbf{Z}_{\leq N}$, for the i -th sample, let a_i^j be the j -th coordinate,

$$\hat{y}_i = \frac{1}{\sum_j e^{a_i^j}} (e^{a_i^1}, \dots, e^{a_i^{n_L}}).$$

4. Loss Function: For classification with softmax from above:

$$\mathcal{L}(y, \hat{y}) = \frac{1}{2N} \sum_i |y_i - \hat{y}_i|^2.$$

8.3.2 Algorithm Implementation

We describe variables in our convolution neural network:

- Convolution2dLayer: $W_1 : 3 \times 3 \times 1 \times 8; b_1 : 1 \times 1 \times 8$.
- Batchnormalization: After getting \hat{x} from (8.3.2), one sets the Offset parameter β and the Scale parameter γ so that $y = \gamma\hat{x} + \beta$. Here $\gamma : 1 \times 1 \times 8$, and $\beta : 1 \times 1 \times 8$.
- FullyconnectedLayer: $W_2 : 10 \times 6272; b_2 : 10 \times 1$, where $6272 = 8 \times 784$.

For self test and testing, we test all training set from the original MNIST. For testing, we load the original MNIST dataset LeCun and Cortes (2005): a total training sample as 60000, with testing sample 10000. We set 64 as size for each mini-batch with learning rate $\alpha = 0.01$. We do stochastic gradient descent to optimize internal variables.

We discuss residual storage for testing. For the k -th epoch from testing, we use $R_k(\cdot) = P_k(\cdot) - Y(\cdot)$ as residual with vectorization to be collected for Anderson acceleration, where P_k is the predicted probability distribution of epoch k , and Y is the true probability distribution. We apply Anderson acceleration into our training and see that it improves prediction accuracy very much.

For self testing and testing, we do **restarted Anderson**: We do Anderson acceleration after collecting residual from 4 epochs, and then update parameters by minimizing total

sum of residuals before the restart, we repeat the process m times. In both cases, we record $4 \times m$ as epoch number.

8.3.3 Results

We compare accuracy results from performing Anderson acceleration and not performing Anderson Acceleration.

Epoch	Standard	Anderson
4	99.31%	99.41%
8	99.83%	99.83%

Table 8.2: CNN Self Test: Accuracy Comparison.

Epoch	Standard	Anderson
4	97.28%	97.69%
8	97.87%	98.02%

Table 8.3: CNN Test: Accuracy Comparison.

Tables 8.2 and 8.3 suggest that Anderson acceleration has better performance: The improvement is remarkable after 2 times of applying Anderson acceleration. Since we require way less epochs to achieve accuracy, CNN will be better design for Neural Network.

We bring more observations to conclude the chapter. First we compare the performance without the use of acceleration:

1. The use of CNN and DNN to train a Neural network exhibits the use of universal approximation property.
2. CNN tables suggest that using a rich architecture on a Neural Network may provide a more accurate prediction result, despite CNN requires a high storage demand

has increased by the architecture of CNN, e.g. filter number amplifies the data structures significantly, as we can see from the size of weight matrix for the fully connected layer. DNN uses matrix as data, while CNN uses two dimension image as data. Therefore, DNN does quite good which connect input to output, while cnn transfers images to digit classification directly.

3. The performance is much better for CNN than DNN. Compared to DNN, CNN converges with 100% with much fewer number of epochs.
4. CNN performs better in terms of accuracy in fewer epochs. DNN per epoch cheaper than CNN. For self testing with all MNIST training samples (60000) as testing set in standard case, operation time per epoch is less for DNN than CNN. DNN: 400 epochs finish in 206.29 seconds; CNN: 8 epochs finish in 178.773 seconds. Therefore, in terms of speed, we have DNN: 0.516 second/epoch; CNN: 22.34 seconds/epoch.
5. Variants of design can be tried, one can, for example, redefine the Neural Network by including a maxpooling layer. our test indicates that the simplest CNN works quite well with respect to cost and CPU time.

Now, we bring aspects from the use of Anderson acceleration.

1. Anderson does good job in the convergence of DNN and CNN. We have used mini-batch gradient (on a DNN) descent and stochastic gradient descent (on a CNN) with Anderson acceleration, and Anderson makes things much improved.
2. The performance is much better for CNN than DNN; So, CNN achieves the level of accuracy with much less CPU time.
3. Anderson performs very well and is much effective for DNN case. Because of richness of CNN architecture improves Anderson less, but does improve.

Chapter 9

Conclusion and Future Work

We give concluding remarks on the thesis and some relevant future work.

9.1 Conclusion

We have

- developed and investigated iterative schemes based on Reduced Order Method to generate basis and accelerate algorithms.
- advanced analyses of Anderson type acceleration method based on the reduced order merit function.
- modeled numerical algorithms on convex optimization problems with ROM.
- tested algorithms to use for applications, including steady state Navier Stokes, a semi-ill posed linear system with specialized structure.
- tested the Anderson type acceleration for Neural Network optimization for deep learning network, with very promising performance. We accelerated the stochastic gradient method through Anderson acceleration.

9.2 Possible Future Work

Further application of Anderson type acceleration for more general class of applications, such as nonlinear system of saddle point problems, and numerical optimization including control and inverse and optimal design problems. Specifically, the idea of DNN is quite powerful and we extend to further applications with Anderson acceleration, e.g. inverse medium problem, control problems, PINNs, and scientific computing in general.

REFERENCES

- Arnoldi, W. E. (1951). The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29.
- Barzilai, J. and Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148.
- Benamou, J. and Brenier, Y. (2000). A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84:375–393.
- Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.
- Bridle, J. S. (1990a). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fogelman-Soulié, F. and Héroult, J., editors, *Neurocomputing: Algorithms, Architectures and Applications*, pages 227–236. Springer. Presented at NATO ASI on Neurocomputing, Les Arcs, France, 1989.
- Bridle, J. S. (1990b). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 211–217, San Mateo, CA. Morgan Kaufmann.
- Brown, P. N. and Walker, H. F. (1997). Gmres on (nearly) singular systems. *SIAM Journal on Matrix Analysis and Applications*, 18(1):37–51.
- Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593.
- Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90.
- Cauchy, A.-L. (1847). *Analyse mathématique. – méthode générale pour la résolution des systèmes d'équations simultanées.*
- Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322.
- Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154.
- Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–327.

- Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435.
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.
- Householder, A. S. (1941). A theory of steady-state activity in nerve-fiber networks: I. definitions and preliminary lemmas. *The Bulletin of Mathematical Biophysics*, 3(2):63–69.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR.
- Ito, K. and Jin, B. (2014). Inverse problems. *Series on Applied Mathematics*.
- Ito, K. and Ravindran, S. S. (1998). A reduced-order method for simulation and control of fluid flows. *Journal of Computational Physics*, 143:403–425.
- Ito, K., Reisinger, C., and Zhang, Y. (2021). A neural network-based policy iteration algorithm with global h2-superlinear convergence for stochastic games on domains. *Found. Comput. Math.*, 21(2):331–374.
- Kelley, C. T. (2003). *Solving Nonlinear Equations with Newton’s Method*, volume 1 of *Fundamentals of Algorithms*. SIAM, Philadelphia.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems (NeurIPS)*, 2:396–404.
- LeCun, Y. and Cortes, C. (2005). The mnist database of handwritten digits.
- Malitsky, Y. and Mishchenko, K. (2020). Adaptive gradient descent without descent. In *Proceedings of the 37th International Conference on Machine Learning, ICML’20*. JMLR.org.
- Marmarelis, V. Z. and Zhao, X. (1997). Volterra models and three-layer perceptrons. *IEEE Transactions on Neural Networks*, 8(6):1421–1434.

- Nitanda, A. (2014). Stochastic proximal gradient descent with acceleration techniques. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.
- Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition.
- Paquette, E. and Trogon, T. (2023). Universality for the conjugate gradient and min-res algorithms on sample covariance matrices. *Communications on Pure and Applied Mathematics*, 76(5):1085–1136.
- Parikh, N. and Boyd, S. (2014). Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239.
- Polak, E. and Ribiere, G. (1969). Note sur la convergence de méthodes de directions conjuguées. *Revue française d’informatique et de recherche opérationnelle. Série rouge*, 3(R1):35–43.
- Pollock, S. and Rebholz, L. (2021). One-step convergence of inexact anderson acceleration for contractive and non-contractive operators. *IMA Journal of Numerical Analysis*, 41(2):2841–2872.
- Pulay, P. (1980). Convergence acceleration of iterative sequences. the case of scf iteration. *Chemical Physics Letters*, 73(2):393–398.
- Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Saad, Y. and Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869.
- Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656.
- Sherman, J. and Morrison, W. J. (1949). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Annals of Mathematical Statistics*, 20(4):621–622.

- Strohmer, T. and Vershynin, R. (2009). A randomized kaczmarz algorithm for linear systems with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278.
- Tychsen-Smith, L. and Petersson, L. (2017). Denet: Scalable real-time object detection with directed sparse sampling. *CoRR*, abs/1703.10295.
- Walker, H. F. and Ni, P. (2011). Anderson acceleration for fixed-point iterations. *SIAM J. Numer. Anal.*, 49:1715–1735.
- Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235.
- Woodbury, M. A. (1950). Inverting modified matrices. Technical Report Report 42, Statistical Research Group, Princeton University. Memorandum Rept., Princeton University, 1950.
- Xue, F. (2022). One-step convergence of inexact anderson acceleration for contractive and non-contractive mappings. *ETNA - Electronic Transactions on Numerical Analysis*, 55:285–309.