CONTROL TECHNIQUES FOR ANALYSIS AND SYNTHESIS OF ROOT SOLVING ALGORITHMS

BY

GAYATHRI MOHAN

B.Eng., Anna University, 2006

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Adviser:

Professor Srinivasa M Salapaka

Abstract

The application of control theory to the design and analysis of numerical algorithms has not been extensive. In this thesis, we have studied and presented methods based on control theoretic tools that can be used to develop and analyze root solving algorithms. Root solving algorithms can be viewed as dynamical systems with feedback, which enables us to employ techniques that are used to design feedback control laws. We have primarily focused on using Lyapunov stability concepts to generate iterative methods in a systematic fashion, as against predominant ad hoc procedures used in literature. We have drawn comparisons between the newly designed algorithms and the conventional gradient methods, to evaluate their convergence rates. Additionally, we present an optimal control formulation to analyze existing numerical algorithms and synthesize new ones. Furthermore, we have incorporated Lyapunov stability tools to develop a scheme that can estimate an eigenvector of symmetric matrices. We have demonstrated all of the methods on several examples with poor condition numbers. Our results show fast convergence rates which are comparable to the existing gradient algorithms.

To mom and dad

Acknowledgements

Here goes the most read section of the thesis. I take this opportunity to express my gratitude to my teachers, family and friends. I would first like to thank my adviser, Dr. Srinivasa Salapaka, for giving me the inspiration to pursue work in the field of controls. My special thanks to him for motivating me at times when it was much needed. His immense support and constructive guidance have been vital in making this work possible. I have thoroughly enjoyed working with him and think it a great pleasure to continue working under his guidance. Thanks to him for his critical comments and suggestions during the thesis writing process.

Amongst my fellow graduate students, I would like to thank my lab mates, present and past, especially, Chibum Lee. His valuable inputs at several instances were of great help. I would also like to thank my adviser and lab mates for the numerous interesting discussions, not just academic. A word of thanks to the MechSE department for giving me the opportunity to interact with some of the great minds at UIUC.

I am thankful to all my dear friends, in particular Vignesh for keeping my enthusiasm soaring. My friends at UIUC have made life so much nicer and made me feel at home. Thanks a ton for the wonderful times. Special thanks to Vijay for helping me learn LaTeX. I would like to convey my thanks to my brother Mani, for the cheerful long-distance phone chats.

Finally, I am grateful to my parents, Mr. Mohan and Mrs. Visalakshi Mohan, for their undying love and encouragement at all times. I cannot thank them enough for making me what I am today!

Table of Contents

List of Figures

Chapter 1

Introduction

The role of numerical algorithms in scientific studies and technological applications is becoming increasingly dominant due to their rapidly rising use since a few decades. The development of computer technology over the last few decades in terms of its speed has had dramatic impact on scientific computations. Some equations that were not possible to solve before, such as Navier Stokes equations in fluid dynamics, have become accessible to numerical computations in relatively short times. The science and technology that has developed from analysis of these solutions and numerical simulations of the underlying processes has in turn engendered new areas of research and applications. These new studies, in addition to traditionally computationally complex problems, have motivated formulations of new problems that are computationally very intensive and require efficient algorithms to address them. Some of the areas that require large computation requirements include data classification algorithms relevant to combinatorial optimization problems in battle field management resource allocation problems, genomics, combinatorial drug discovery, social networks, pattern recognition, as well as network related problems as in cyber networks and sensor networks.

A simplified description of a numerical algorithm for a given problem is an iterative method that converges to the solution of the problem. The performance of a numerical algorithm is mainly characterized by computation-time or rate of convergence, to reach the solution within a tolerance threshold. Most research related to development of numerical algorithms consist of proposing modifications to existing general principles, such as Newton's steepest-descent principle, that aim at improving this criterion. In spite of many general

principles that are known today, there is no systematic unified framework to analyze, evaluate, modify, and develop new algorithms. In particular, there are scant existing principles for evaluating and designing for algorithms that are insensitive to discretization (quantization) errors, implementation platforms (finite precision or not), and structural variations in the underlying equations. This thesis develops a framework that will address most of these deficiencies, especially for the root-solving algorithms.

In most of the application areas mentioned above, and in many others, one of the most time-intensive numerical steps involves finding roots of a system of equations. Most of the root-solving algorithms are *iterative*, a vast majority of which are variants of Newton's algorithm based on descent methods. It should be emphasized that even though there is a large evidence of sophisticated use of numerical algorithms in the control systems theory, there is relatively scant use of control systems tools in the literature on numerical algorithms.

There are a few instances in literature where the possible role of control and dynamical systems tools with respect to numerical analysis are discussed and some preliminary approaches suggested. However, they have not been expanded to form a systematic framework. Tsypkin [1], [2] identified the centrality of the role of gradient dynamical systems in iterative algorithms for numerical analysis. They presented a control perspective of the problem of adaptation and learning. Unfortunately, this line of thought was not pursued actively by the controls community . Another notable contribution is Sobolev's book [3] which demonstrates the spurt method, a variable structure method for solving a linear system of equations, as a case of variable control applied to a Krylov method.

In the last decade, there are relatively more consolidated evidences, but still few, of viewing iterative algorithms as dynamical systems . In [4], a dynamical system based approach to numerical algorithms is proposed. However, it does not consider control inputs and therefore does not attempt to modify the iterative-method dynamics. The book by Moore et al [5], tackles a wide class of constrained optimization problems, exploiting techniques from parallel computing and neural networks, following a dynamical systems approach. The

articles [6], [7] show how to map data associated with a linear programming (LP) problem into a dynamical system, whose states evolve to the solution of the LP. These results are extended to find systems that can solve generic combinatorial optimization problems and it also provides an algorithm to diagonalize symmetric matrices.The book [8] elucidates the applications of dynamical systems in search and optimization using ideas of renormalization. The most recent work by Bhaya et al [9] consolidates their contributions in employing a controls perspective on numerical algorithms. A control system approach to derive existing algorithms along with new continuous time algorithms and their discrete counterparts has been reported. The work presented in this thesis is also along the same theme.

In our proposed framework, iterative algorithms are viewed as control systems, that will facilitate analysis and synthesis of the root-solving algorithms by using tools and perspectives from the control and dynamic systems The platform that we propose focuses particularly on root solving algorithms and introduces a systematic framework by the way of feedback mechanisms. In addition, these mechanisms will automate key aspects of existing root-solving procedures which currently are designed ad-hoc and require significant experience and familiarity with the underlying structure in the problem.

There are several advantages of the control systems viewpoint as it easily incorporates the notions of robustness to quantization errors and 'imprecision or noise' in equations, and therefore sensitivity of the algorithms to specific parameters in the equation. For instance, it is common in traditional root-solving algorithms, typically based on descent methods, to add ad hoc variations called *acceleration factors* that make the algorithm converge quickly. These factors are problem-specific, added after a few trials, and their number depends entirely on the experience of the person coding the algorithm and his/her familiarity to the equation of interest. The proposed systems viewpoint would help to automate these factors, besides laying down general architectures to develop new algorithms.

Additionally, this framework provides powerful conceptual constructs that can be extended to analysis and design of numerical algorithms. For instance, in the same lines as

the design of control-Lyapunov functions that facilitate design of feedback laws in controls theory, we develop formulations of control-energy functions, whose time-derivatives when made negative, will yield algorithms that drive the initial conditions to the solution of equations. Another approach that we are developing is to formulate optimal-control problems that minimize a cost function that measures the deviation of the current state from the error. Furthermore, using Lyapunov stability theory, we present methods that can determine an eigenvector of symmetric, positive-definite matrices.

This thesis is structured as follows. Our work on deriving existing and new numerical algorithms for root solving, using the Lyapunov function based approach and the Optimal Control formulation based approach, is discussed in detail in Chapter 2. In Chapter 3, the use of Lyapunov function based techniques to determine eigenvalues of matrices and preliminary results on preconditioning using similar methods are presented. Finally, Chapter 4 summarizes the results and elaborates the scope for future work.

Chapter 2

Control Systems Framework for Numerical Algorithms

2.1 Root-solving Algorithms

In this section, we present a dynamical system viewpoint of numerical algorithms and demonstrate the performance of new algorithms derived under this viewpoint. Our focus lies on root-solving algorithms, that is algorithms that find zeros or roots x of possibly nonlinear vector-valued equations of the type $f(x) = 0$. Numerical algorithms for root solving are predominantly iterative, i.e. they attempt to find successive approximations to the solution starting from an initial guess $x(0)$. A large class of the root solving algorithms fall under the class of *descent methods*, which include the well known steepest descent and conjugate gradient methods. The iterations in the descent methods are characterized by the form

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.1}$$

where d_k is a suitably chosen direction and α_k is the step size measuring the step along the direction d_k. To guarantee that the iterates approach the solution, the direction d_k is chosen to satisfy $d_k^T \nabla f(x_k) < 0$ whenever the gradient $\nabla f(x_k)$ of f at x_k is non-zero. Different choices of d_k and α_k gives rise to several root-solving algorithms.

The convergence of descent methods are heavily dependent on the choice of the parameters α_k, the step size and d_k, the direction. These parameters in most existing descent methods are designed following a general greedy approach and are ad hoc at most times. In order to facilitate a systematic routine to generate these parameters, we view the root-solving

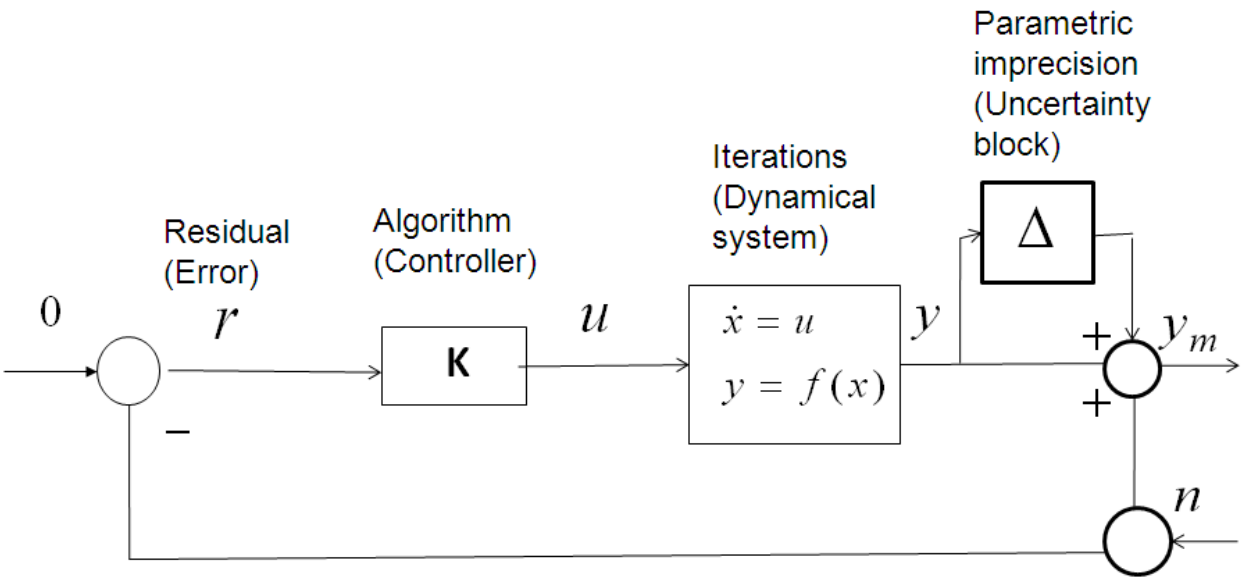


Figure 2.1: Control system viewpoint for root solving algorithms

algorithms in a dynamical system setup. This viewpoint has several advantages, a vital one being that it enables application of control theoretical tools to systematically synthesize new algorithms.

2.2 Dynamic System Framework

The dynamical system viewpoint of a root-solving algorithm can be realized, by writing it as the following state and output equation pair,

$$\begin{aligned} \dot{x} &= u \\ y &= f(x). \end{aligned} \tag{2.2}$$

Figure 2.1 depicts the control-system viewpoint for the root-solving algorithms. The goal of the control-design is to design a dynamic system that drives the output y of the system (2.2) to zero. Under this framework generating the algorithm parameters such as the step size and

direction can be made more systematic in addition to providing us with tools to develop new algorithms. Since a root-solving algorithm aims to reduce the residue or the error between the solution and the iterate at each step, it can be compared to the regulation problem in control systems. Regulation deals with the problem of keeping one or more variables of the system at a constant value through feedback.Thus, by interpreting the feedback signal as the residue in the iterative methods, we can translate root solving algorithms to the dynamic system framework. More specifically, in Figure 2.1 the control input u is designed to regulate $f(x)$ to zero. The error,

$$r = -f(x), \tag{2.3}$$

which is the difference between the actual and the desired value of y is an input to the control feedback law K.

The operation of root solving algorithms can be compared to the problem of regulation in control theory as follows.

- The initial condition $x(0)$ in the dynamic system is analogous to the initial guess x_0 in iterative methods.
- The error signal in Figure 2.1 is comparable to the residue which is driven to zero in the iterative methods.

Therefore any iterative root solving algorithm can be expressed as a control system by suitably choosing the controller K and the dynamic system. An example of the steepest descent method is shown next.

Example: The steepest descent or steepest gradient algorithm is given by:

$$x_{k+1} = x_k + \alpha_k r_k; \qquad x(0) = x_0; \tag{2.4}$$

$$r_{k+1} = r_k + \alpha_k A r_k; \tag{2.5}$$

$$\alpha_k = \frac{r_k^T r_k}{r_k^T A r_k} \tag{2.6}$$

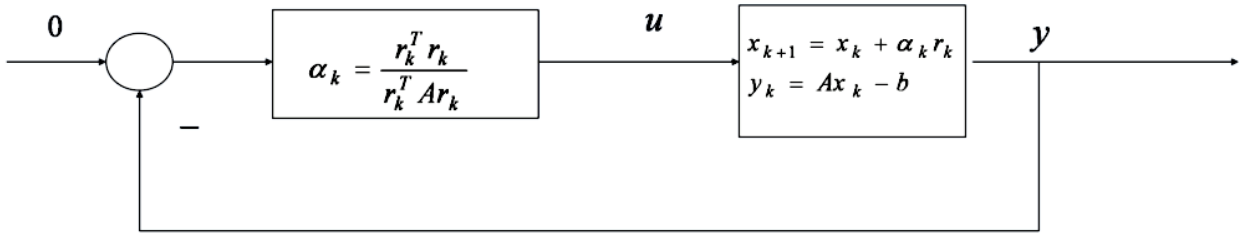


Figure 2.2: Steepest descent or gradient method in the control system viewpoint. Showing an interconnection of linear plant and nonlinear controller

The steepest descent method can be transformed into a control system with feedback as shown in Fig 2.2. It must be noted that this is a discrete-time system unlike the general model discussed earlier which was in continuous time. In this example the controller generates the step size parameter α_k. Similarly, a dynamic system representation of the conjugate gradient method (CGM) as a proportional derivative controller has been presented in [9].

An additional advantage of using this viewpoint is that the algorithms can be analyzed in terms of robustness to quantization and modeling errors. These errors can be treated as noise and model uncertainties respectively. There is well developed theory available in control literature for studying robustness to such errors. In Fig 2.1 n represents the quantization noise and Δ denotes the uncertainties in the plant/model (2.2). A more detailed discussion on this using the steepest descent method as an example is provided at the end of the chapter.

In control theory, there are several tools used to analyze and design controllers. The Control Lyapunov function (CLF) is a popular scheme to design stabilizing controllers for dynamical systems. The use of this method to derive existing root solving algorithms and design new ones is discussed in the next section.

2.3 Control Lyapunov Function Based Approach

In our Control-Lyapunov-Function based approach for root-solving algorithms, we cast the underlying problem in a *continuous-time* setting and then approximate the resulting controller by a discrete-time model for implementing it digitally. The continuous-time formulation is useful since the tools and theory for analysis of differential equations are better developed than those for the discrete counterparts. This section first broaches the notion of stability in dynamical systems. Consider the dynamical system,

$$\dot{x} = f(x), \text{ where } x(0) = x_0, \tag{2.7}$$

where without loss of generality, zero is considered to be the equilibrium point.

Definition **[10]** The equilibrium point $x = 0$ is said to be

- **Stable** if for arbitrary t_0 and for every $\epsilon > 0$, there exists $\delta = \delta(\epsilon, t_0) > 0$ such that $||x(t_0)|| < \delta$, implies $||x(t)|| < \epsilon$ for all $t \geqslant t_0$
- **Asymptotically stable**, if it is stable and for arbitrary t_0 there exists $\beta(t_0)$ such that $||x(t_0)|| < \beta(t_0)$ implies $\lim_{t\to\infty} x(t) = 0$
- **Exponentially stable**, if in addition to stability, $||x(t_0)|| < \beta(t_0)$ implies $||x(t)|| \leqslant K||x(t_0)||e^{-\alpha(t-t_0)}$, for α and K greater than zero.

In Definition 2.3, asymptotic stability AS essentially means, if we start close enough to the equilibrium, we eventually reach the equilibrium. This concept of AS best suits our purpose because it is the most useful to mimic the regulation problem. A phenomenal toolset to study or determine the stability of an equilibrium point in a dynamical system are Lyapunov functions. A candidate Lyapunov function may be defined as a continuous, positive definite function such that $V(0) = 0$ and $V(x) > 0$ in a neighborhood of $x = 0$. The stability definitions can be rewritten in terms of conditions on Lyapunov functions.

Several such Lyapunov stability results exist. For our purposes, we focus on the version for continuous, time invariant systems.

Theorem 2.3.1 [10] *Consider the system in* (2.7) *and let* $V(x)$ *be a positive definite real valued function defined on* $\mathcal{D}$*, a closed bounded region containing the origin of* $\mathbb{R}^{n\times n}$*. The zero solution of* $\dot{x} = f(x)$ *is*

- ***Stable*** *if* $\dot{V} = (\nabla V)^T f(x) \leqslant 0$
- ***Asymptotically Stable*** *if* $\dot{V}$ *is negative definite*
- ***Globally asymptotically stable*** *if* $\mathcal{D} = \mathbb{R}^{n\times n}$
- ***Exponentially stable*** *if stable and* $\alpha_1||x||^2 \leqslant V(x) \leqslant \alpha_2||x||^2$ *and* $-\alpha_3||x||^2 \leqslant \dot{V}(x) \leqslant -\alpha_4||x||^2$ *for positive* $\alpha_1, \alpha_2, \alpha_3$ *and* α_4*.*

More specifically, a Control Lyapunov function $V(x)$ is a continuous, positive definite and radially unbounded function in x. In general a Control Lyapunov Function (CLF) is used to check whether a given system is feedback stabilizable. This is done by considering a candidate Lyapunov function $V(x, u)$ and checking if

$$\exists u \text{ such that } \dot{V}(x, u) < 0 \qquad \forall x \neq 0. \tag{2.8}$$

Therefore, a CLF checks if a controller is capable of making the given dynamical system asymptotically stable (Theorem 2.3.1) or not. We design the numerical algorithms in a similar manner,though the functions that we choose for the design are not Lyapunov functions. They are similar to the Lyapunov functions except that they are not positive definite. Therefore, we will call the function that is chosen for our purposes to be a Lyapunov based function. The La Salle's theorem [11] provides us with the necessary stability results for such Lyapunov based functions.

Theorem 2.3.2 (*La Salle's theorem* [10]) *Let $\mathbf{\Omega} \subset \mathcal{D}$ be a compact set that is positively invariant with respect to $\dot{x} = f(x)$. Let $V : \mathcal{D} \rightarrow \mathbb{R}$ be a continuously differentiable function such that $\dot{V}(x) \leq 0$ in $\mathbf{\Omega}$. Let E be the set of all points in $\mathbf{\Omega}$ where $\dot{V}(x) = 0$. Let M be the largest invariant set in E. Then every solution starting in $\mathbf{\Omega}$ approaches M as $t \rightarrow \infty$.*

The procedure that has been prescribed for generating the numerical algorithms, involves choosing a suitable candidate Lyapunov based function $V(x)$ according to the structure of $f(x)$ such that $V(x) = 0$ implies $f(x) = 0$, and then determining a control input u that will make the time derivative $\dot{V}(x)$ negative along the trajectories of the system. In order to elucidate the Control Lyapunov Function based scheme better we will consider the following example. In this example, we derive the Newton's method using the Lyapunov function approach.

Example:

1. The equation $f(x) = 0$, where $f(x)$ is continuous, is posed as a regulation problem by considering the dynamical system

$$\begin{aligned} \dot{x} &= u, \\ x(0) &= x_0, \\ y &= f(x), \end{aligned} \tag{2.9}$$

   in which the output y is regulated to zero.

2. A candidate energy function $V(x) = \frac{1}{2} f(x)^T f(x)$ is considered since $f(x)$ is continuous.

3. The time derivative $\dot{V}(x)$ along the trajectory x is computed to be

$$\dot{V}(x) = (\frac{\partial V}{\partial x})\dot{x} = f(x)^T (\frac{\partial f}{\partial x})u.$$

4. The next step would be to choose a control input u such that $\dot{V}(x)$ is made negative

for all $x \neq 0$. The following choice for u,

$$u = -\alpha(\frac{\partial f}{\partial x})^{-1} f(x) \qquad \text{where } \alpha > 0, \tag{2.10}$$

is a valid candidate for the control input.

5. u in (2.10) makes $\dot{V}(x) = -\alpha||f(x)||^2$, which is less than zero for $f(x) \neq 0$. This would make the dynamic system in (2.9)

$$\dot{x} = -\alpha(\frac{\partial f}{\partial x})^{-1} f(x). \tag{2.11}$$

6. Using forward difference to discretize the continuous time system results in the following discrete time equations with time step γ_k,

$$\begin{aligned} \frac{x_{k+1} - x_k}{\gamma_k} &= -\alpha(\frac{\partial f}{\partial x})^{-1} f(x) \\ \Rightarrow x_{k+1} &= x_k - \alpha\gamma_k(\frac{\partial f}{\partial x})^{-1} f(x). \end{aligned} \tag{2.12}$$

This is same as the Newton's iterative method, also, several of its variants can be obtained by manipulating the values of α and γ_k. In this procedure the forward difference scheme for discretization has been used. The scope of discretizing using other methods has not yet been explored.

In the scheme discussed, the energy function or the candidate Lyapunov function has been chosen as a function of x. However most stability results based on Lyapunov functions have been carried out about a zero equilibrium point. To achieve this the system is redefined in terms of the equilibrium $r = 0$. The equivalent system with respect to the residue r is

given by

$$\begin{aligned} \dot{r} &= -(\frac{\partial f}{\partial x})\dot{x} \\ r(0) &= -f(x(0)) \end{aligned} \tag{2.13}$$

The following section draws comparisons between several admissible control inputs to (2.13) that stabilize the system. The CPU time taken to execute the algorithm on MATLAB is considered as the measure of convergence time.

2.4 Developing Numerical Algorithms through Controllers for Linear Systems

In several scientific applications, linear system of equations with large matrices have to be dealt with. From here on, our attention lies on the linear systems of equations of the form $Ax = b$, where, $A \in \mathbb{R}^{n\times n}$, x and b belong to $\mathbb{R}^n$. For the linear system case, (2.2) can be simplified as,

$$\begin{aligned} \dot{r} &= Au \\ r(0) &= Ax(0) - b \end{aligned} \tag{2.14}$$

Now, a candidate energy function can be chosen and the procedure explained earlier to derive numerical algorithms, can be executed there on. Depending on the choice of the Lyapunov based energy function, both static and dynamic controllers can be designed. The following portions include an analysis and comparison of static and dynamic controllers that give rise to new root-solving algorithms.

2.4.1 Static Controllers

This section contains the synthesis of static controllers, that is control laws where u is a function of only the residue r and not its time derivatives. In order to design static-control feedback law for (2.14) that solve $Ax - b = 0$ we choose the quadratic function

$$V(r) = \frac{1}{2}||r||_2^2 \tag{2.15}$$

as our energy function. The time derivative of V along r would be $r^T\dot{r}$. The next step according to the CLF approach defined earlier would be to design a control u that makes this derivative negative. For the chosen Lyapunov function several such choices of u can be admitted.

- The simplest choice of u that makes $\dot{V}$ negative is $-\gamma r$, where γ is a positive number. In this case low eigenvalues of A would result in lower derivative of the energy function since

$$u = -\gamma r \quad \text{for } \gamma > 0 \tag{2.16}$$

$$\Rightarrow \dot{V} = -\gamma r^T A r. \tag{2.17}$$

  For the feedback control law (2.16), a low condition number implies longer time for convergence. This is because,

$$\begin{aligned} \dot{V} &\leq -\gamma \lambda_{min}(A) r^T r \\ &\leq -\gamma \lambda_{min}(A) 2V, \end{aligned}$$

  and by Gronwall's inequality [12],

$$\Rightarrow \quad V(t) \leq V(0) e^{-2\gamma \lambda_{min}(A) t}. \tag{2.18}$$

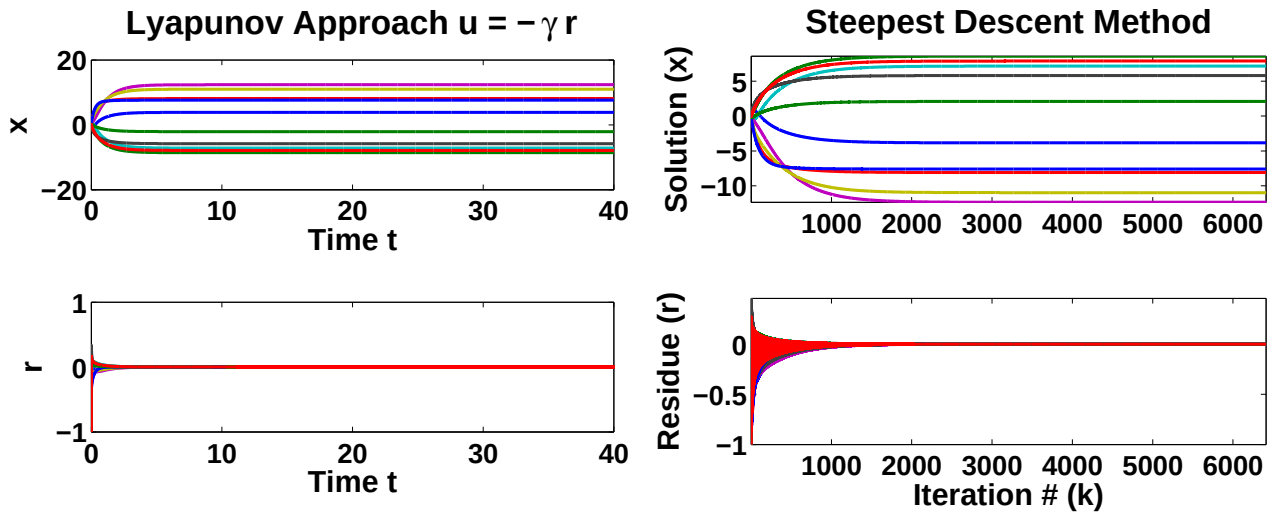


Figure 2.3: (Left) Lyapunov function based controller $u = -\gamma r$, $\gamma = \frac{1}{\lambda_{min}(A)}$. (Right) Steepest Descent method. Plots show the state trajectories x and residue r, for A of size 10 and $\rho = 6.1465 \times 10^3$.

However, the effect of the lowest eigenvalue $\lambda_{min}(A)$ in the time derivative $\dot{V}$, (2.17), can be mitigated by choosing the value of γ close to the reciprocal of $\lambda_{min}(A)$. This in fact has been validated through simulations in MATLAB for matrices of different dimensions and condition numbers.

An example of a 10×10 symmetric, positive-definite matrix, A is shown in Figure 2.3. Here A has a condition number of $\rho = 6.1 \times 10^3$. For a γ value of one, the algorithm performs poorly as expected. Upon using a value of $\gamma = \frac{1}{\lambda_{min}(A)}$ the convergence time improves. Although convergence time is still higher compared to the steepest descent method, we consider this feedback control only to establish proof of concept. It is useful to note that the Lyapunov based controller has a definite drawback in that it requires the knowledge of at least an estimate of $\lambda_{min}(A)$ a priori. A Lyapunov function based approach to determine the lowest eigenvalue of symmetric, positive definite matrices has been described in Chapter 3.

- In the following, we present another design for a static controller that overcomes the dependance of the dynamics of $V(t)$ on $\lambda_{min}(A)$. This feedback control law is given

by,

$$u = -\frac{r^T r}{r^T A r} r \tag{2.19}$$

$$\Rightarrow \dot{V} = -r^T r \tag{2.20}$$

From the equation (2.20) we observe that,

$$\dot{V} = -2V$$

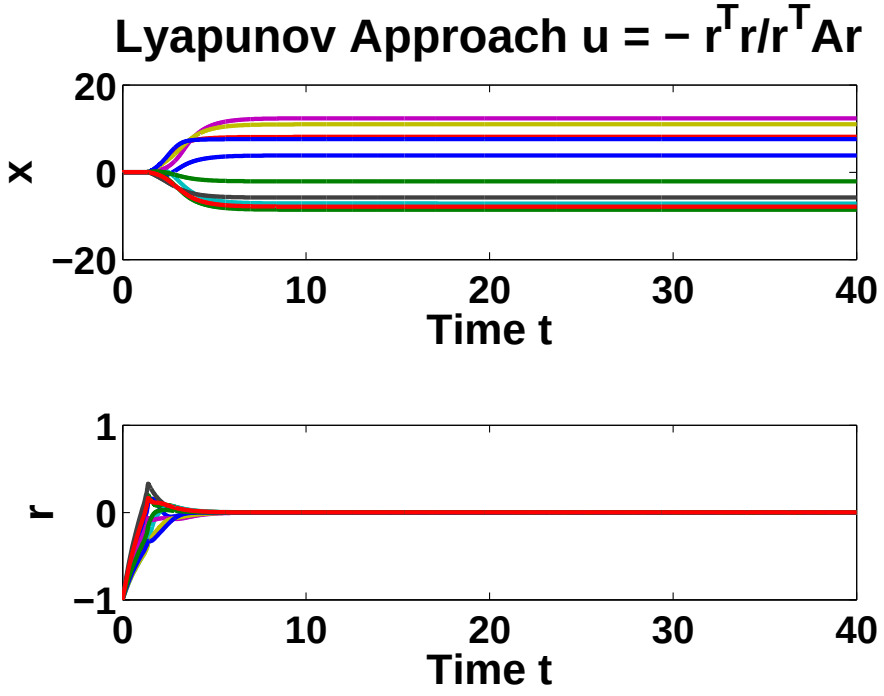
$$\Rightarrow V(t) = V(0)e^{-2t}. \tag{2.21}$$

Therefore, the dependance on $\lambda_{min}(A)$ has been eliminated (2.21), resulting in faster convergence rate than the control law (2.16). It must be noted that there still exists the risk of high cost of control (2.19) for low eigenvalues of A. The results of applying (2.19) to matrices of size 10 is presented in Figure 2.4. The convergence times for systems in Figures 2.4(a) and 2.4(b), in terms of CPU time in Matlab are 367 and 290 seconds, respectively. These observations show that the controller (2.19) four times as fast as (2.16).

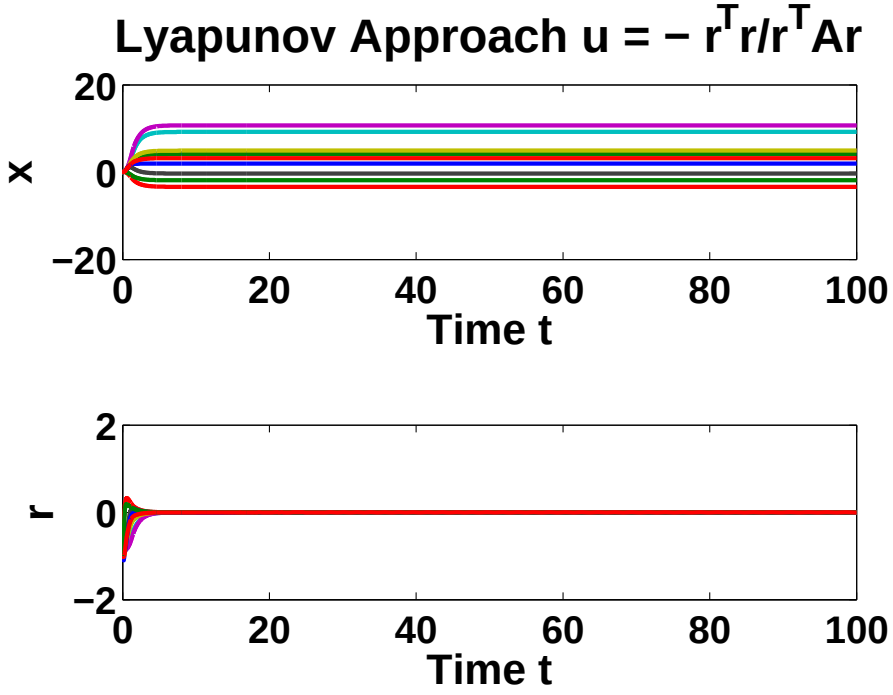
So far, we have shown the synthesis and performance of static controllers for root-solving. In the following section, we extend this to explore the option of dynamic controllers.

2.4.2 Dynamic Controllers

We refer to the class of controllers in this section as dynamic, since they involve the dynamics of the control input, which form a part of the state equations. This section analyzes the design of dynamic controllers for the problem of solving linear systems of equations, as in Section 2.4.1. The procedure for design of dynamic controllers is very similar to that for the static controllers, except for the modification in the system equations, as mentioned above.



(a) Matrix of size 10 and condition number $\rho = 6.15 \times 10^3$.



(b) Matrix of size 10 and condition number, $\rho = 756$

Figure 2.4: Controller in (2.19).

Our dynamic system in (2.2) is now rewritten as

$$\begin{aligned} \dot{x} &= \xi, \\ \dot{\xi} &= u, \\ y &= f(x), \end{aligned} \tag{2.22}$$

where ξ is the additional state that defines the dynamics of the control input. The Lyapunov based energy functions are also appropriately modified owing to the changes in the state equations. We propose the following energy function,

$$V(r,\xi) = \frac{1}{2}\alpha r^T r + \beta \xi^T \xi. \text{ Therefore its derivative is given by,} \tag{2.23}$$

$$\dot{V} = \alpha r^T \dot{r} + \beta \xi^T \dot{\xi}. \tag{2.24}$$

The term $\xi^T\xi$ in Lyapunov function V implies that ξ will be small when the control is designed to make $\dot{V}$ negative definite. This ensures that control values do not become too large, which in turn, guarantees good numerics for implementation of the the system in (2.22). This is in contrast to static controllers, where the values of u can become very large and the resulting algorithm may require solving system of stiff differential equations. We can add more flexibility to the energy function V by introducing a symmetric positive definite matrix,

$$P = \left[\begin{array}{c|c} P_{11} & P_{12} \\ \hline P_{12} & P_{22} \end{array}\right]. \tag{2.25}$$

The resulting energy function is,

$$V(r,\xi) = \begin{bmatrix} r^T & \xi^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix} \begin{bmatrix} r \\ \xi \end{bmatrix} \tag{2.26}$$

$$\dot{V} = r^T P_{11} A\xi + r^T P_{12}\dot{\xi} + \xi^T A^T P_{12}\xi + \xi^T P_{22}\dot{\xi} \tag{2.27}$$

To designate a control input for such an energy function, we adopt the universal Sontag's formula for stabilization with bounded controls [13]. If we define,

$$\begin{aligned} \frac{\partial V}{\partial r}\dot{r} &= L_{vf}, \\ \frac{\partial V}{\partial \xi} &= L_{vg}, \end{aligned} \tag{2.28}$$

then, from (2.27) we get,

$$\begin{aligned} L_{vg} &= r^T P_{12} + \xi^T P_{22}; \\ L_{vf} &= r^T P_{11} A\xi + \xi^T P_{12} A\xi. \end{aligned} \tag{2.29}$$

The Sontag's universal feedback formula is given by the following equation,

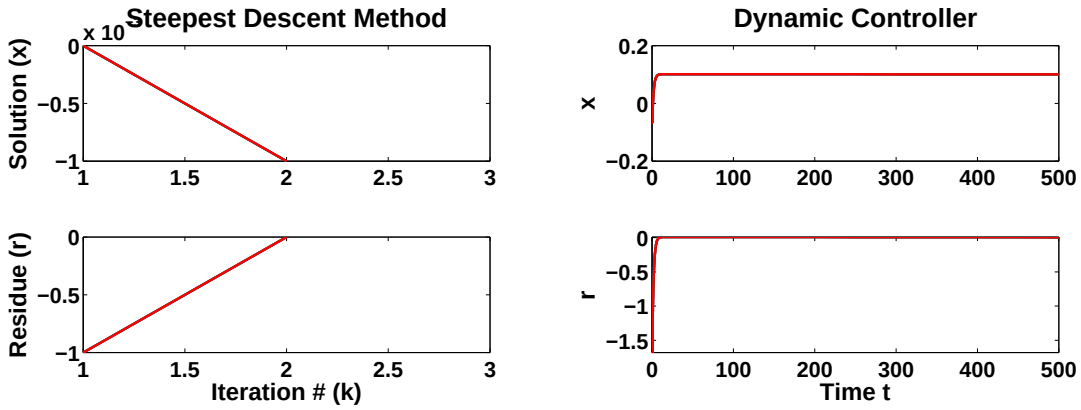
$$u = -L_{vg}^T \left[L_{vf} + \frac{\sqrt{||L_{vf}||^2 + ||L_{vg}||^4}}{||L_{vg}||^2} + k_0 \right] \text{ , where } k_0 \text{ is a positive real number} \tag{2.30}$$

Usage of this feedback signal ensures that the time derivative of our energy function is negative. The component blocks of the P matrix are chosen to be,

$$\beta = ||A||_F \tag{2.31}$$

$$P_{11} = \frac{1}{\beta^2} I; \;\; P_{12} = \frac{1}{\beta} I; \;\; P_{22} = \psi I \tag{2.32}$$

where, β is the Frobenius norm of the matrix A, I is an identity matrix of the size of A and ψ is a positive real. This choice of parameters is sufficient for P to be positive definite. As an example to demonstrate this design, a poorly conditioned A is considered. The matrix is symmetric, positive definite of size three and condition number, $\rho = 1 \times 10^5$. The CPU time taken for convergence reduces by one-third that of the time taken by the static control law in (2.19). In fact, it is comparable to the steepest descent method. The dynamic control logic takes on the order of 160 seconds of CPU time, while the steepest descent method takes



(a) (b)

Figure 2.5: The steepest descent method (a) and dynamic controller (b) performance on a matrix with $\rho = 1$

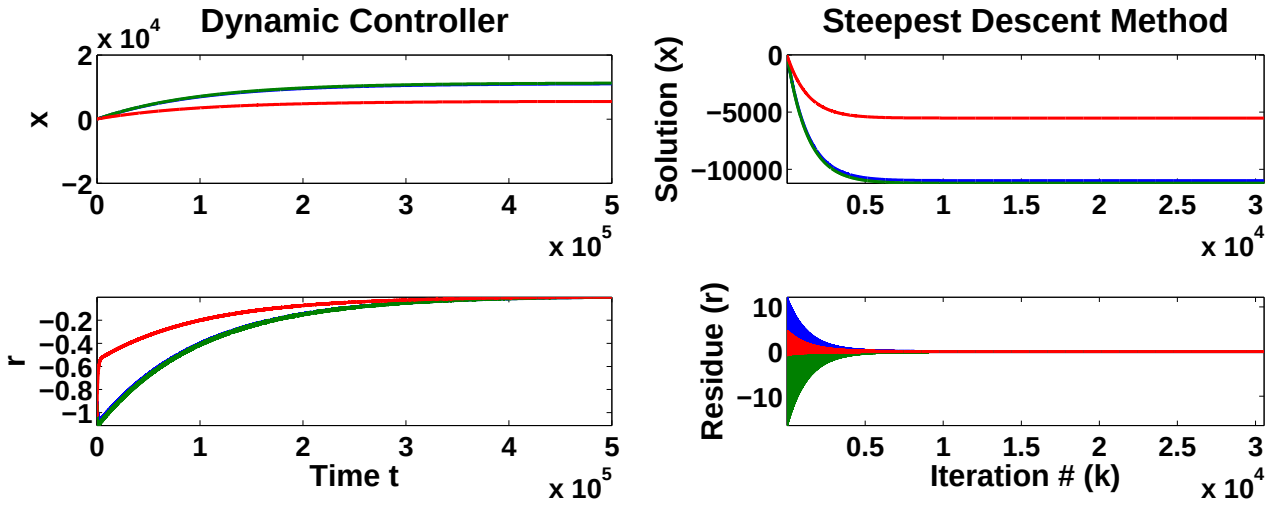


Figure 2.6: (Left) Trajectory x and residue r for A with condition number $\rho = 1 \times 10^5$, using the dynamic controller logic. (Right) Steepest Descent plots for the same example.

approximately 110 seconds of CPU time to converge with an error tolerance of 1×10^{-3}. Compared to tests on matrices with condition number 1, the steepest descent method took 93% and the dynamic controller took 88% lesser time than the respective convergence times for the poor condition-number cases.

The *ode23* solver in Matlab is used to solve the differential equations, during the implementation of both the static and dynamic controllers. We are yet to investigate the possibility of implementing a dedicated discretization routine, which will exploit the algorithm parameters such as the step size α_k at each step. Improvisations on the choice of P and a self developed discretization scheme have scope of improving the convergence rate further.

2.5 Optimal Control Formulation Approach

In this section we discuss the design of root-solving algorithm by formulating appropriate optimal control problems The procedure involves formulating the appropriate cost functions and solving for the control law u that will satisfy the following regulation objective,

$$\min_{u \in U} g_1(r(N)) \quad + \quad \sum_{i=0}^{N} g_2(x, u) \tag{2.33}$$

$$\text{such that } x(k+1) \quad = \quad x(k) + u(k), \quad r(0) = r_0. \tag{2.34}$$

The residual at the k^{th} step is defined by $r(k) = f(x(k))$, U is the allowable set of control laws, and the functions g_1 and g_2 are designed to accomplish performance objectives. For example, choosing $g(r(N))$ to be N and $g_2 = 0$ is equivalent to the minimum time optimal control problem [14]. Other algorithms can be derived by constraining the set U of allowable controls to impose specific desired constraints. Furthermore, in the unconstrained equivalent of this problem the Lagrange multipliers, which quantify the sensitivity of the cost function to constraints, can be modified to obtain the algorithms too. A major advantage of using

this algorithm is that it provides scope for generalization of the algorithms to nonlinear systems of equations that are not based on linearization tools and preserve the heuristics through the cost functions. We have been able to show that certain widely used algorithms, for positive definite, symmetric matrices, such as the steepest descent method and conjugate gradient method (CGM) can be deduced from this approach. This gives the motivation that extensions to such algorithms [15], to deal with asymmetric and indefinite cases, can be derived through this approach.

2.5.1 Deriving Existing Methods with Optimal Control Approach

In the following we present the derivation of the steepest descent and conjugate gradient methods using the optimal control formulation approach.For this, we consider a linear system of equations of the form $Ax - b = 0$, where A is a symmetric positive definite matrix, and x, and b are vectors of suitable dimensions. Then, we pose a suitable constrained optimization problem,

$$\begin{aligned} &\min_{u \in U} g_1(r(N)) \\ \text{such that } r(k+1) &= r(k) + Au(k), \quad r(0) = Ax(0) - b, \end{aligned} \tag{2.35}$$

where $g_1(r(N))$ is a generic cost function that depends on the residual at the Nth step. Reformulation of (2.35) as an unconstrained minimization. Using the variational approach, we obtain the necessary conditions that the solution must satisfy. More specifically, we seek the optimal trajectories $(r^*(k), u^*(k), \lambda^*(k))$ such that the variation $dJ = 0$ along these trajectories where J is the unconstrained Lagrangian given by

$$J(r, u, \lambda) = g_1(r(N)) + \sum_{k=0}^{N-1} \lambda^T(k+1)[r(k) + Au(k) - r(k+1)], \tag{2.36}$$

where λ is the Lagrange multiplier. The following necessary conditions were obtained for maintaining dJ at zero.

$$\begin{aligned} r^*(k+1) &= r^*(k) + Au^*(k), \quad r^*(0) = Ax_0 - b \\ \lambda^*(k+1) &= \lambda^*(k) + v(k), \quad \text{where } v(k) \perp r^*(k) \\ \lambda^*(k+1) Au^*(k) &= 0. \end{aligned} \tag{2.37}$$

These necessary conditions are satisfied by both the steepest descent and the conjugate gradient methods. In fact imposing the orthogonality constraint,

$$r^*_{k+1} \perp r^*_k, \tag{2.38}$$

in addition to the necessary conditions (2.37) yields the steepest descent method. This can be achieved by choosing v_k in the direction of r_{k+1}.

Similarly, addition of a different orthogonality constraint to the necessary conditions (2.37) results in the conjugate gradient method. The constraint is to satisfy

$$r^*_{k+1} \perp span(r(0), \cdots, r^*_k) \tag{2.39}$$

Therefore enforcing the following condition,

$$\lambda^*_{k+1} = \lambda^*_k + \theta_k r_{k+1}, \text{ for } \theta_k \in \mathbb{R}, \tag{2.40}$$

where λ_k is chosen to be $\frac{d_k}{||r_k||^2}$ and taking $u_k = \frac{-\lambda_k}{\lambda_k^T A \lambda_k}$, guarantees that the necessary conditions are satisfied and the conjugate gradient method is obtained. This result promises extension of the conjugate gradient method to non-definite matrices as well as the nonlinear function case since the variational approach is applicable to either of these cases.

2.6 Scope for Analyzing Existing Algorithms

The dynamic system viewpoint of numerical algorithms renders a wide range of tools for the analysis of these algorithms. Many existing algorithms are sensitive to parametric imprecisions in the function $f(x)$, quantization errors and other noise n, and initial guess $x(0)$. These algorithms may converge quickly for some equations but may not converge for some other equations. These algorithms can be studied, especially for robustness using well-developed control tools. In particular, we propose the use of concepts based on passivity analysis to study the robustness of interacting nonlinear blocks. This includes the re-interpretation of a numerical algorithm as a Lure-system of interconnected systems for sector nonlinearities and dynamic blocks [10,16]. In the following section, we describe the analysis of the steepest descent method under the Lure-system setup as an example.

2.6.1 Example - Steepest Descent Method

It has been shown already in Figure 2.2 that the steepest descent can be viewed in terms of a control block diagram, in the discrete time setup. The controller block in the steepest gradient method for solving a symmetric linear system of equation $Ax = b$ is given by a nonlinear block that acts on the residual $r = b - Ax$ (or the regulation-error) given by $K = \phi(r) = r^T r/(r^T Ar)$. Therefore the steepest descent method can be thought as an interconnection of a linear block with a static nonlinearity $\phi(r)$. Moreover, since $\phi(r)$ is the inverse of the Rayleigh quotient of A, it satisfies, $\lambda_{\max}^{-1}(A) < \phi(r) < \lambda_{\min}^{-1}(A)$, that is $\phi(r)$ satisfies sector condition and the interconnection can be treated as the well-studied Lure system and analyzed for absolute stability [10]. Thus the steepest gradient method is amenable to robustness analysis as well as convergence analysis by tuning the associated Lyapunov function in the absolute stability analysis [10, 16].

Similarly, the controller block in the conjugate gradient method (CGM) [15] can be thought of as a dynamic controller [9], which lends itself to passivity analysis discussed

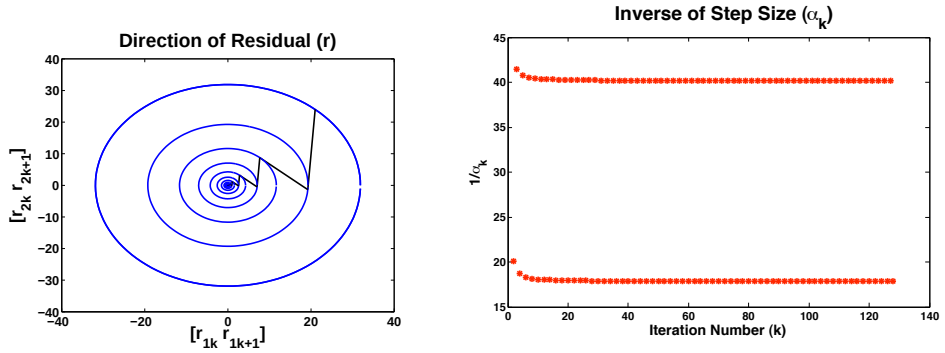
in [10]. In fact, most existing iterative algorithms for solving linear system of equations are explained in terms of Lanczos Iterations [17], and these iterations can be easily motivated as dynamic blocks, therefore lending themselves to robustness analysis via control techniques.

Chapter 3

An Approach to Determine Eigenvectors

3.1 Introduction

In this chapter, we present techniques based on well-developed control tools to determine eigenvectors of symmetric, positive-definite matrices. Prior knowledge of an eigenvector of linear systems before the application of gradient methods to solve for its roots is very useful. For instance, consider the steepest descent algorithm for solving linear system of equations $Ax = b$, Section 2.2. The step size α_k used in this method (2.6) can be derived through the minimization of a quadratic function $\Phi(x)$,

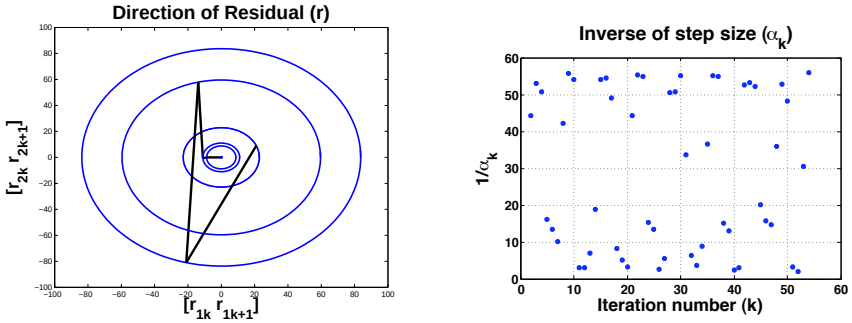


(a) (b)

Figure 3.1: (a) Progression of residue to the solution shown on the level sets of $\Phi(x)$ for a 2×2 A matrix case, for the L1 algorithm. (b) Inverse of the step size α_k at each step

$$\Phi(x) = x^T Ax - x^T b, \text{ for } x \neq 0 \tag{3.1}$$

The generic path taken by the steepest descent method to reach the solution, plotted on the elliptical level sets of $\Phi(x)$, is as shown in Figure 3.1(a). The steepest descent algorithm proceeds in such a way that the direction taken at each step is orthogonal to the direction at the previous step, i.e. $r_{k+1} \perp r_k$, for all k. Essentially, the residue vector follows a path where it toggles between two directions alternately. It must be noted that the eigenvectors of the system matrix A form the principal axes of the level sets shown in Figure 3.1(a). This implies that if the initial condition for the steepest descent method x_0 is chosen as an eigenvector of the system matrix A, then the convergence to solution will occur in a single step, with a suitably chosen step size α.



(a) (b)

Figure 3.2: (a) Progression of residue to the solution shown on the level sets of $\Phi(x)$ for a 2×2 A matrix case, for the L1D algorithm. (b) Inverse of the step size α_k at each step

In fact, there exist algorithms in literature such as the L1D method [18], which explores several directions in the m-dimensional space (m denotes the size of A) to find the directions close to the principle axes of the Φ level sets (Figure 3.2(a)). The L1D procedure reaches the solution in reduced number of iterations, as expected. The algorithm is achieved by delaying the step size parameter α_k to α_{k-1} at each step. This bypasses the orthogonality condition, $r_k \perp r_{k+1}$, present in the conventional steepest descent algorithm and approaches the eigenvector directions instead. Therefore, we seek to develop a method to determine the eigenvector direction of matrices, which can then be used as the initial condition for x in gradient root-solving algorithms.

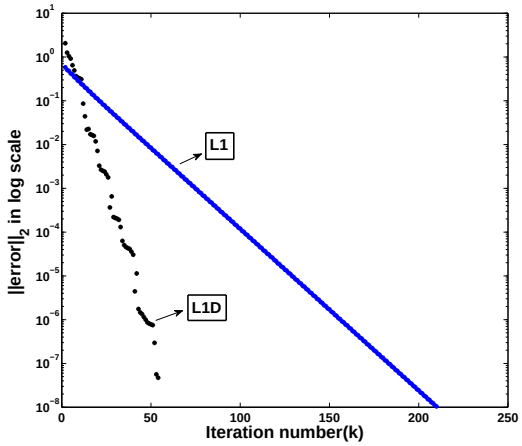


Figure 3.3: Comparison of error norm of L1 and L1D algorithms in log scale.

3.2 Determining Eigenvectors - Lyapunov Function Based Approach

In order to achieve our objective of determining an eigenvector of symmetric, positive-definite matrices we adopt, once again, a Lyapunov-function based approach that uses the stability concepts described in Chapter 2. In this direction, we design a non-negative energy function $V(x)$, where x is the state whose trajectory we want to converge to the eigenvector of matrix A. For this we seek control inputs u that make the time derivative $\dot{V}$ of the energy function negative. We use the dynamic system state equation $\dot{x} = u$.

We discuss some choices for the energy function and the corresponding control inputs below. We also provide demonstration of the methods on several examples.

3.2.1 Energy Function I

The first choice of the energy function $V(x)$ is based on the Cauchy's inequality. The fundamental Cauchy - Schwartz inequality [19] states that,

$$| < y, x > | \leq < x, x >^{1/2} < y, y >^{1/2}, \tag{3.2}$$

for all x, $y \in \mathbf{C^n}$ with equality if and only if x and y are dependent or the same when normalized.

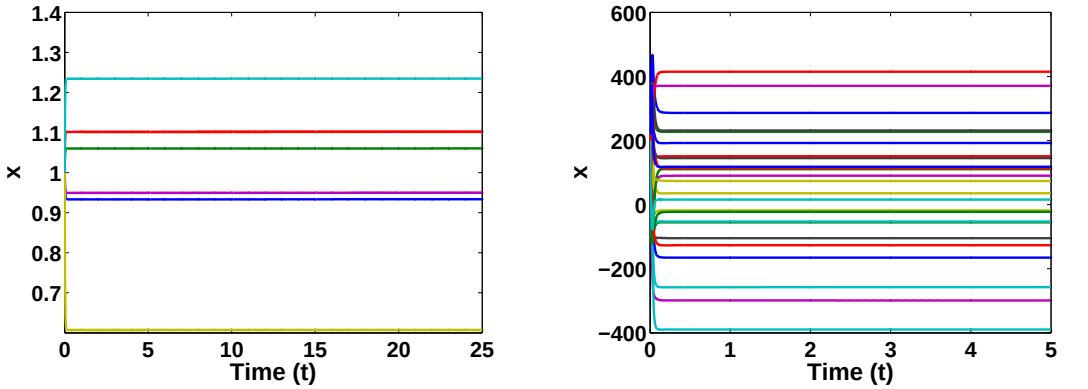
The energy function that we use is defined by,

$$V(x) = \frac{1}{2}\left[\frac{||Ax||}{||x||} - \frac{x^T Ax}{||x||^2}\right]^2, \tag{3.3}$$

and its derivative is calculated to be,

$$\dot{V} = \left[\frac{||Ax||}{||x||} - \frac{x^T Ax}{||x||^2}\right]\tilde{\Phi}u \tag{3.4}$$

$$\tilde{\Phi} = -\frac{||Ax||x^T}{||x||^3} + \frac{x^T A^2}{||x||||Ax||} + \frac{2x^T Axx^T}{||x||^4} - \frac{2x^T A}{||x||^2}. \tag{3.5}$$

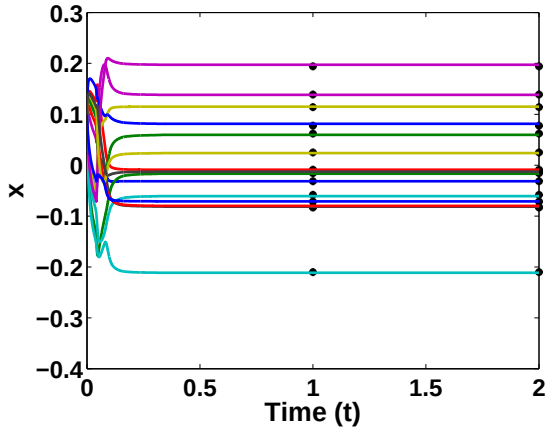


(a) $x_0 = ones(m, 1)$, $t_c = 1.8$ seconds (b) $x_0 = 200 \times ones(m, 1)$, $t_c = 1.3$ seconds

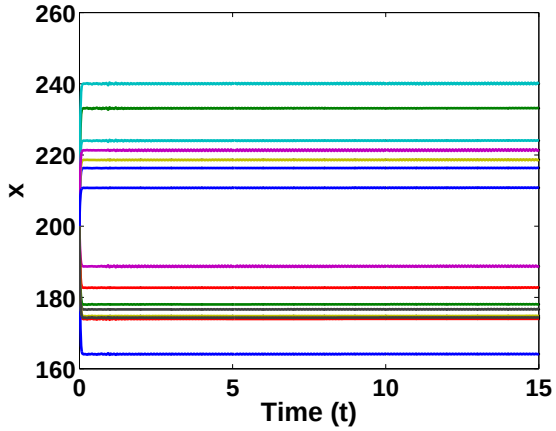
Figure 3.4: (a) $m = 6$, $\rho = 1.32 \times 10^4$.(b) $m = 25$, $\rho = 1.41 \times 10^4$.

Here $\tilde{\Phi}^\dagger$ denotes the pseudo-inverse [20] of the vector $\tilde{\Phi}$. From Cauchy's inequality, for a unit vector x, the two terms in (3.3) are equal only when x is an eigenvector of A. The control law

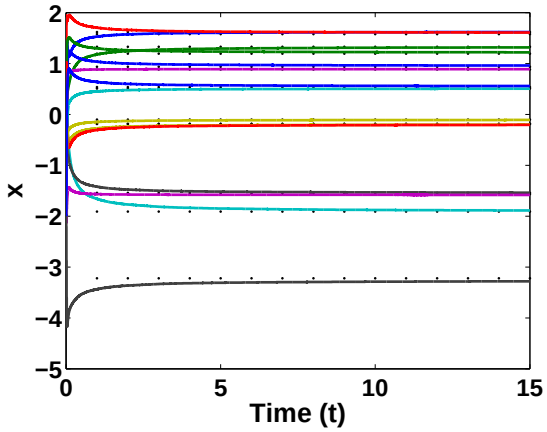
$$u = -k\left[\frac{||Ax||}{||x||} - \frac{x^T Ax}{||x||^2}\right]\tilde{\Phi}^\dagger \text{ where } k \text{ is a positive constant,} \tag{3.6}$$



(a) $x_0 = 0.1 \times ones(m, 1)$



(b) $x_0 = 200 \times ones(m, 1)$, $t_c = 5.2$ seconds



(c) $x_0 = (ones(m-5, 1); 2 \times ones(5, 1))$, $t_c = 65$ seconds

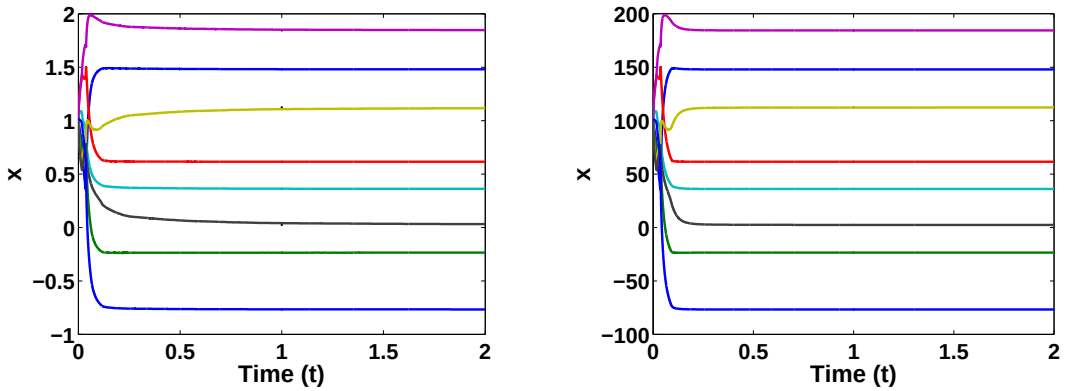
Figure 3.5: $m = 15$, $\rho = 1.71 \times 10^7$, correspond to the control law in (3.6).

is designed in such a way that it makes the time derivative (3.4) negative. Therefore by La Salle's theorem [10], the solution converges to the set $S := \{x : \dot{V}(x) = 0\}$. From (3.4) and (3.6) we get that,

$$\dot{V}(x) = -kV(x)\tilde{\Phi}\tilde{\Phi}^{\dagger}, \tag{3.7}$$

which shows that S is a subset of the union of sets $\{x : V(x) = 0\}$ and $\{x : \tilde{\Phi}\tilde{\Phi}^{\dagger}\}$. Thus it follows that $\dot{V}(x) \to 0$ when $V(x) \to 0$ or $\tilde{\Phi}\tilde{\Phi}^{\dagger} \to 0$. Using Cauchy's inequality we infer that the former occurs only when x is along an eigenvector direction of A and calculations show the same for the latter.Therefore the trajectories of the initial condition of x converges

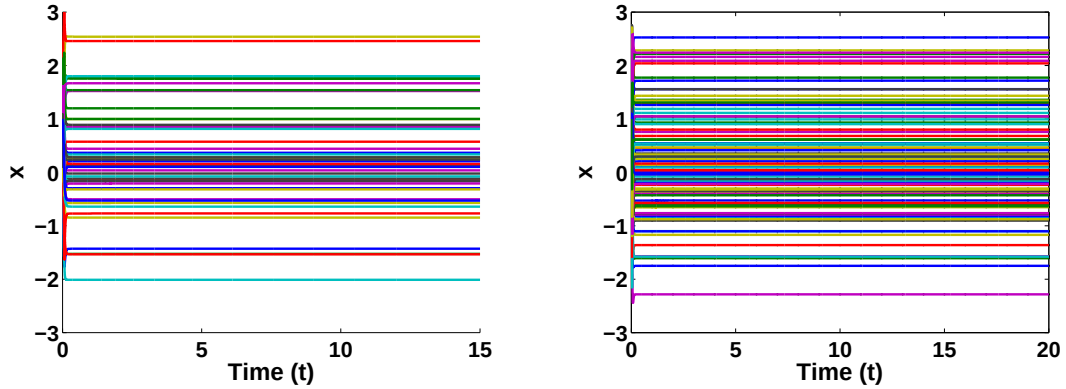
to some eigenvector of the system, and this has been verified on several examples. In these examples, the size and condition number of the system matrices, and the initial conditions for the algorithm have been varied. In all the figures, m denotes the size of the matrix, ρ



(a) $x_0 = ones(m, 1)$ (b) $x_0 = 100 \times ones(m, 1)$

Figure 3.6: $m = 8$, $\rho = 73.64$, control law in (3.6)

denotes the condition number, x_0 denotes the initial condition and $||e||$ gives the norm of the error, between the original eigenvector and the solution of the Lyapunov-based method. The convergence times (t_c) given refer to the CPU times taken by Matlab to arrive at the desired error norm.



(a) $m = 50$, $\rho = 2.2 \times 10^5$, $x_0 = ones(m, 1)$ (b) $m = 100$, $\rho = 1.6 \times 10^6$, $x_0 = ones(m, 1)$

Figure 3.7: Control law (3.6)

Observations: Figure 3.4 shows the state trajectories for matrices of size 6 and 25, using different initial conditions, converging to the eigenvectors corresponding to the maximum

eigenvalue λ_{max} of the two matrices respectively. The error norm $||e||$ for all the examples shown, is of the order of 1×10^{-9}. Figure 3.5 demonstrates the convergence to some eigenvector of the same matrix for different initial condition vectors. Considering an ascending ordering of eigenvalues, Figure 3.6 shows convergence to the eigenvector corresponding to λ_2 for a matrix of size 8. These figures shows that the method described in this section is consistent and also takes low convergence time. Irrespective of the initial conditions, the algorithm seeks the closest eigenvector direction.

3.2.2 Energy Function II

The Lyapunov function and the control design in this choice is such that the state trajectory converges to the eigenvector corresponding to the lowest eigenvalue λ_{min} of the system. We choose the Rayleigh's quotient of the system matrix A given by $\frac{x^T Ax}{x^T x}$ to be the Lyapunov function. The Rayleigh quotient always remains positive for a positive definite matrix and therefore makes a valid candidate. The Lyapunov function

$$V(x) = \frac{x^T Ax}{x^T x} \tag{3.8}$$

has the following derivative,

$$\dot{V}(x) = \tilde{\Theta} u \tag{3.9}$$

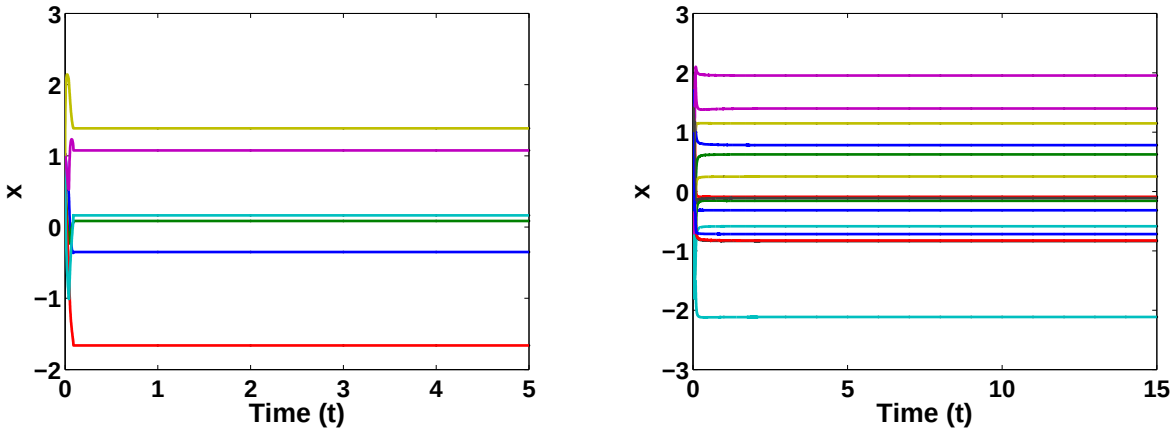
$$\tilde{\Theta} = \frac{2x^T A}{x^T x} - \frac{2x^T Axx^T}{x^T x^2}. \tag{3.10}$$

The control law that makes this derivative negative is computed to be,

$$u = -k\frac{x^T Ax}{x^T x}\tilde{\Theta}^{\dagger} \text{ where } k \text{ is a positive constant.} \tag{3.11}$$

By La Salle's theorem [10], we infer that the time derivative $\dot{V}(x)$ converges to zero. From the equations (3.9) and (3.11), we get,

$$\dot{V}(x) = -kV(x)\tilde{\Theta}\tilde{\Theta}^{\dagger}. \tag{3.12}$$

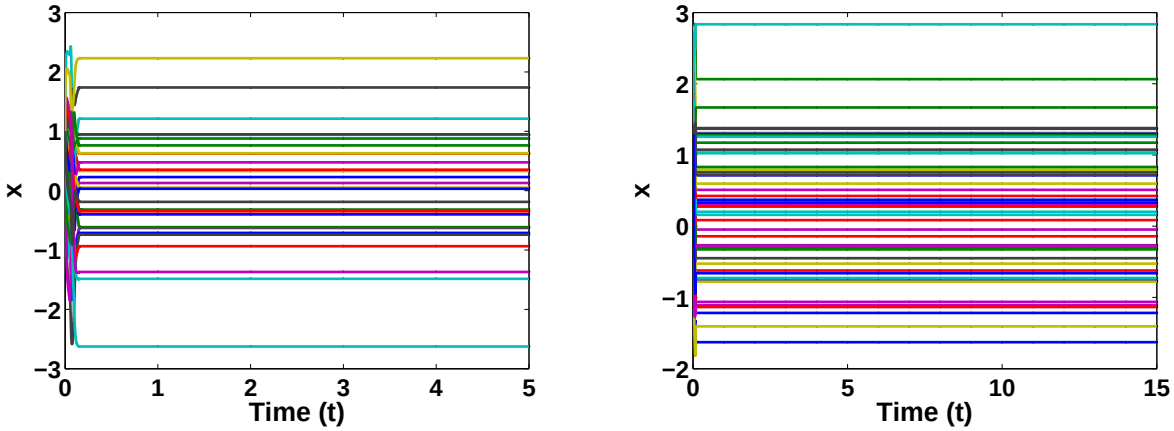


(a) $m = 6$, $\rho = 1.32 \times 10^4$. (b) $m = 15$, $\rho = 1.71 \times 10^7$

Figure 3.8: Control law (3.11)

It can be seen that $\dot{V}(x)$ can be zero only if $\tilde{\Theta}$ or its pseudo-inverse becomes zero. This is because $V(x)$ remains positive for all non-zero values of x. It can also be observed from (3.10) that $\tilde{\Theta}$ is zero if and only if x is in the direction of an eigenvector of A. Therefore, we conclude that $\dot{V}(x) \to 0$ implies the convergence of x to the eigenvector of A. Furthermore, it always converges to the eigenvector corresponding to the minimum eigenvalue $\lambda_{min}(A)$ because $V(x)$ is decreasing and bounded below by $\lambda_{min}(A)$.

Observations: The tests performed on example matrices establish that irrespective of the initial conditions used, this choice of Lyapunov function and control law in (3.11) drive the state to converge to the eigenvector corresponding to $\lambda_{min}(A)$ repeatedly. The maximum convergence times taken remain within $\pm 5\%$ of 6 seconds for an error norm $||e|| \approx 1 \times 10^{-9}$. This holds in spite of variations in the initial conditions. For symmetric positive-definite



(a) $m = 28$, $\rho = 1.55 \times 10^6$ (b) $m = 50$, $\rho = 2.2 \times 10^5$

Figure 3.9: Control law (3.11)

matrices of sizes between $m = 6$ and $m = 50$, converging trajectories of the initial state have been shown in Figures 3.8 and 3.9. The maximum value of condition number for which results have been presented is $\rho = 1.71 \times 10^7$.

3.3 Preconditioning

Preconditioning methods are applied to improve convergence of root solvers, however, these methods are problem specific, in the sense that they exploit the structure of $f(x)$ to design the pre-conditioner. For instance, in the preconditioned conjugate gradient method (PCGM), the solution to a linear system of equations $Ax = b$ is obtained by solving $PAx = Pb$, where the preconditioner P is designed such that the convergence properties of the latter system of equations is much better (in fact P is designed to guarantee clustering of eigenvalues of PA [15, 21]). This design of P requires exploiting the structure of A known a priori and depends mainly on the expertise and the experience of the person designing it. In the proposed framework, the output y in the Figure 3.10 is given by $y = Ax - b$. The controller block K can be viewed as consisting of two blocks K_1 and K_2 in series, where K_1 can be designed as a preconditioner and K_2 as a control block for the regulation objective. For

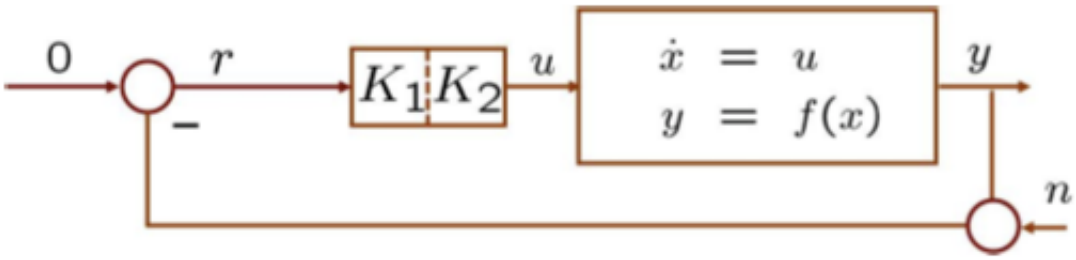


Figure 3.10: The controller K is considered to compose two parts K_1, which improves the numerical-conditioning of the equation and K_2 that achieves the regulation objective.

instance, K_1 can be chosen to be a constant matrix P or to transform $Ax = b$ to,

$$\begin{aligned} Az &= Pb \\ \text{where, } A &= PAP^T, \\ P^T z &= x. \end{aligned} \tag{3.13}$$

This is followed by posing appropriate design constraints on the controller block K, which can be effected by suitably modifying the Lyapunov-based function. As an example, if K_1 transforms A to P^TAP, then the energy function would involve terms that impose P^TAP to be an orthogonal matrix. Therefore, the energy function in the Lyapunov based method or the objective function in the optimal control approach (Chapter 2) incorporates the relevant additional conditions over regulation.

3.4 Inferences

We have so far explained a scheme that can determine an eigenvector of symmetric positive definite matrices. Once the eigenvector corresponding to $\lambda_{min}(A)$ has been determined, our aim is to use it as the initial condition in gradient based numerical algorithms to achieve convergence in a single step. Although this is theoretically proven, it is observed through simulations that the gradient methods are not robust to even low numerical aberrations in the initial condition vector. This obstacle can be overcome by the preconditioning of the A

matrix. Preconditioning is equivalent geometrically, to making the quadratic cost function level sets closer to a sphere than an ellipsoid, thereby making the numerical imprecisions less pronounced.

Chapter 4

Conclusions and Future Work

In this thesis, we have demonstrated and validated a systematic framework that uses control theoretic tools to generate and study numerical algorithms. Although a framework of this sort can be generalized to several numerical algorithms, our focus in this thesis has been on the root solving algorithms. To start with, we have suggested a perspective of viewing numerical algorithms as dynamical systems, which facilitates the use of tools and ideas from control theory. Subsequently, we have developed an algorithm based on Lyapunov-stability theory to derive some of the existing algorithms. In addition to this, we have also shown the capability to synthesize new root solving algorithms through this method. This has been demonstrated by designing static as well as dynamic control laws using the aforementioned method, that can solve for roots of linear systems. Furthermore, we have presented several examples comparing the performance of the algorithms developed. The algorithms that resulted from the dynamic control laws performed better than the static control based methods. The dynamic control based methods are comparable to the convergence rate of the steepest descent method and the appropriate comparisons have been presented. In all the schemes explained, it must be noted that the possibility of improving performance using a self designed discretization scheme, for solving the differential equations, has not yet been explored and we propose to pursue this in the future. Using a customized discretization algorithm can improve the performance of the new algorithms manifold, over the performance they exhibit using the discretization scheme built into the Matlab ODE solvers.

We have explored the use of another powerful control theoretic tool, optimal control, for our application. In this thesis, we have elucidated the use of optimal control theory to

emulate the conjugate gradient method, which gives us the motivation that this approach can be used to design novel algorithms. We aim to use the optimal control method to formulate a generalized conjugate gradient that can be applied to system matrices that are not symmetric or positive definite. Such extensions that have been suggested in the literature so far suffer drawbacks such as slow convergence time and large storage space requirements [17]. There is scope to avoid these shortcomings by adding appropriate constraints to the cost function.

An important setback in the methods that have been detailed in this thesis is that all of these focus on centralized control logic. In real world applications, the numerical computations involve matrices that are manifold larger than the matrices considered for our work. We propose to make our algorithms scalable to large systems by means of distributed control. This we believe, can be effected through parallelization. Under parallelization, the large system would be divided into several subsystems and the root solving algorithms would be applied to the subsystems. Simultaneously, the coupling and interaction between subsystems will be dealt with. Our goal is to construct a structured and distributive control framework, which treats the large problem as several robust control subproblems and will understand the coupling terms as disturbances or noise and attenuate them.

Furthermore, with the use of tools from Lyapunov stability theory, we have developed methods to determine an eigenvector of symmetric positive definite systems. Specifically, we have devised and demonstrated a Lyapunov based method to find the eigenvector corresponding to the lowest eigenvalue of a matrix. The method works with a very low computation time even for poorly conditioned matrix cases. In support of this claim, we have presented trajectory plots and computation time for sevaral examples, demonstrating the convergence of trajectories to an eigenvector of the system. For the positive definite case, a possible extension would be to devise a routine that can determine the remaining eigenvectors of the matrix, given the vector corresponding to the least eigenvalue, using concepts like projection on orthogonal subspaces. The Lyapunov based methods for eigenvector determination can

also be extended to asymmetric matrix cases by modifying the energy functions accordingly.

Preconditioning of the linear system improves the convergence rates of root-solvers. Preconditioners can be implemented by either adding extra conditions to the regulation requirement in the Lyapunov based approach or by appropriately altering the objective function in the equivalent optimal control problem. We have gathered preliminary evidence through implementations on small diagonal matrices. We propose to design feedback controllers that can regulate as well as precondition linear systems, thereby automating the process of preconditioning.

References

[1] Y. Tsypkin, *Adaptation and Learning in Automatic Systems.* Academic Press, New York, 1973.

[2] ——, *Foundations of Theory of Learning Systems.* Academic Press, New York, 1971.

[3] M. Krasnoselskii, J. Lifshitz, and A. Sobolev, *Positive Linear Systems: the Method of Positive Operators.* Heldermann Verlag, Berlin, 1989.

[4] A. Stuart and A. Humphries, *Dynamical Systems and Numerical Analysis.* Cambridge University Press, 1998.

[5] U.Helmke and J. Moore, *Optimization and Dynamical Systems.* Springer Verlag, London, 1994.

[6] R. Brockett, "Least squares matching problem," *Linear Algebra Applications*, vol. 122/123/124, pp. 710–777, 1989.

[7] ——, "Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems," *Linear Algebra Applications*, vol. 146, pp. 76–91, 1991.

[8] H. Pronzato, L. Wynn, and A. Zhigljavsky, *Applications of Dynamical Systems in Search and Optimization.* Chapman and Hall, 2000.

[9] A. Bhaya and E. Kaszkurewicz, *Control Perspectives on Numerical Algorithms and Matrix Problems.* SIAM, 2006.

[10] H. Khalil, *Nonlinear Systems.* Macmilan Library Reference, 1993.

[11] J. LaSalle, "Some extensions of liapunov's second method," *IRE Transactions on Circuit Theory*, vol. 7, no. 4, pp. 520–527, December 1960.

[12] T. Gronwall, "Note on the derivative with respect to a parameter of the solutions of a system of differential equations," *The Annals of Mathematics, Second Series*, vol. 20, no. 4, pp. 292–296, July 1919.

[13] Y. Lin and E. Sontag, "A universal formula for stabilization with bounded controls," *Systems and Control Letters*, vol. 16, no. 6, pp. 393–397, June 1991.

[14] M. Athans and P. Falb, *Optimal Control - An Introduction to the Theory and its Applications*. Dover Publications, 2006.

[15] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, ser. Texts in Applied Mathematics. Springer, 2000, vol. 37.

[16] M. Vidyasagar, *Nonlinear Systems Analysis*, 2nd ed. SIAM, 2002.

[17] G.H.Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[18] J. Ryder, "Optimal iteration schemes suitable for general non-linear boundary element modelling applications," in *7th International Conference on Computer Methods and Advances in Geomechanics*, 1991, pp. 1079–1084.

[19] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 2007.

[20] E. Moore, "On the reciprocal of the general algebraic matrix," *Bulletin of the American Mathematical Society*, vol. 26, pp. 394–395, 1920.

[21] D. Luenberger, *Linear and Nonlinear Programming*, kluwer academic publishers group ed. Prentice Hall, 2003.