MATHEMATICAL APPROACHES TO MODELING, UNDERSTANDING, AND CONTROLLING THE ACUTE INFLAMMATORY RESPONSE TO PATHOGEN AND ENDOTOXIN

by

Judy D. Day

B.S., Mount Union College, 2000

Submitted to the Graduate Faculty of
the Department of Mathematics in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2007

UNIVERSITY OF PITTSBURGH

MATHEMATICS DEPARTMENT

This dissertation was presented

by

Judy D. Day

It was defended on

June 15, 2007

and approved by

Dr. Jonathan Rubin, Department of Mathematics

Dr. Carson Chow, Laboratory for Biological Modeling, NIDDK, National Institutes of Health

Dr. Yoram Vodovotz, Department of Surgery, UPMC

Dr. Beatrice Riviere, Department of Mathematics

Dr. Gilles Clermont, Department of Critical Care, UPMC

Dissertation Director: Dr. Jonathan Rubin, Department of Mathematics

MATHEMATICAL APPROACHES TO MODELING, UNDERSTANDING, AND CONTROLLING THE ACUTE INFLAMMATORY RESPONSE TO PATHOGEN AND ENDOTOXIN

Judy D. Day, PhD

University of Pittsburgh, 2007

This thesis includes work dealing with topics related to the modeling, understanding, and controlling the acute inflammatory response. After the introductory chapter, the second chapter discusses a small (four equation) ordinary differential equation (ODE) model of the acute inflammatory response to endotoxin stimuli. Many scenarios of endotoxin tolerance are reproduced and explained in the context of inflammation. The third chapter explores the numerical aspects of coding an algorithm produced by Bernd Krauskopf and Hinke Osinga [63] for generating 2D (un)stable manifolds for 3D ordinary differential equation systems. The fourth chapter returns to the topic of endotoxin tolerance, but now in an abstract mathematical setting. The fifth chapter presents an exposition regarding the application of nonlinear model predictive control to the four equation ODE model (now with pathogen instead of endotoxin) to explore strategies to modulate the inflammatory response during severe infection.

TABLE OF CONTENTS

PREFACE xii

1.0 INTRODUCTION 1

1.1 Using Mathematical and Engineering Tools to Study the Acute Inflammatory Response 1

1.1.1 Background 1

1.1.2 Prior Research and Current Directions 3

1.2 Interesting Biology-Inspired Mathematical Problems 7

1.2.1 Manifolds 8

1.2.2 Endotoxin Tolerance from a Purely Mathematical Perspective 8

1.3 Organization of Thesis 9

2.0 MODELING ACUTE INFLAMMATION AND ENDOTOXIN TOLERANCE 11

2.1 Introduction 11

2.2 A mathematical model of the acute inflammatory response to endotoxin . . 13

2.3 Model simulations of experimental scenarios Endotoxin tolerance scenario . 17

2.3.1 Endotoxin tolerance scenarios 17

2.3.2 Potentiation scenarios sub-lethal and lethal doses 27

2.4 The importance of the dynamics of the late proinflammatory and anti- inflammatory mediators to tolerance 31

2.5 Insight from the model's responses to endotoxin administration 34

2.6 Discussion 41

2.7 Supplementary materials 45

3.0 AN IMPLEMENTATION OF AN ALGORITHM FOR GENERATING 2-D (UN)STABLE MANIFOLDS FOR 3-D ODE SYSTEMS . . . 51
3.1 Introduction . . . 51
3.2 The Algorithm . . . 53
3.3 Implementation Using MatLab . . . 60
3.4 Limitations and challenges . . . 107
4.0 MATHEMATICAL EXPLORATION OF TOLERANCE . . . 110
4.1 Introduction . . . 110
4.2 Preliminary Definitions . . . 112
4.3 Theorems extending the window of tolerance . . . 114
4.4 First Observations . . . 120
4.5 Tolerance in General 2D Linear ODE systems . . . 128
4.5.1 Setup . . . 128
4.5.2 Case 1: . . . 128
4.5.3 Case 2: . . . 130
4.5.3.1 Case 2a: . . . 131
4.5.3.2 Case 2b: . . . 132
4.5.4 Case 3: . . . 135
4.5.4.1 Case 3a: . . . 136
4.5.4.2 Case 3b: . . . 136
4.5.4.3 Case 3c: . . . 137
4.5.5 Eigenvector Configurations and Regions of Tolerance . . . 139
4.5.5.1 Eigenvector Configuration (a): . . . 140
4.5.5.2 Eigenvector Configuration (b): . . . 145
4.5.5.3 Eigenvector Configuration (c) . . . 150
4.5.5.4 Eigenvector Configuration (d) . . . 155
4.5.6 Examples . . . 160
4.6 Tolerance in 2D General Nonlinear ODE Systems . . . 167
4.6.1 Isoclines . . . 169
4.6.2 Numerical Approaches for Finding Tolerance . . . 190

4.6.2.1 Tolerance Algorithm 190
4.6.2.2 Isoclines 193

5.0 USING NONLINEAR MODEL PREDICTIVE CONTROL TO FIND OPTIMAL THERAPY STRATEGIES TO MODULATE INFLAMMATION 197

5.1 Introduction 197
5.2 Methods 199
5.3 Research and Development Phase 204
5.3.1 Configuration 1: No Mismatch; 1 Therapy 204
5.3.2 Configuration 2: No Mismatch; 2 Therapies 208
5.3.3 Configuration 3: Patient-Model Mismatch; 2 Therapies 209
5.3.4 Extraction Therapy Considered 215
5.3.5 Configuration 4: Patient-Model Mismatch; 2 Therapies; Pathogen Update; C_A therapy cap 217
5.4 Results 222
5.4.1 Alternative Therapies for Multi-Patient Simulations 222
5.4.2 Results: Configuration 1 224
5.4.3 Results: Configuration 2 232
5.4.4 Results: Configuration 3 234
5.4.5 Results: Configuration 4 239
5.4.6 Preliminary Results: Current/Future Configurations 243
5.5 Discussion 245

6.0 CONCLUSION 247

APPENDIX. GLOSSARY OF ABBREVIATIONS 250

BIBLIOGRAPHY 252

LIST OF TABLES

1 Model parameter names and values used in simulations 15

2 Scenario 1 (adapted from the experiments of Sly et al., 2004) 19

3 Scenarios 2a - 2c (adapted from the experiment of Wysocka et al., 2001) . . . 19

4 Scenarios 3a - 3c (adapted from the experiments of Rayhane et al., 1999) . . 20

5 Scenario 4 (adapted from the experiments of Balkhy et al., 1999) 20

6 Scenario 5 (adapted from the experiments of Berg et al., 1995) 20

7 Endotoxin administration parameter values and figure references for in silico simulations of Scenarios 1-8 21

8 Explanation of model parameters used in simulations 50

9 Model parameter ranges for Configuration 3 211

10 Model parameter ranges for Configuration 4 219

11 Results for Configuration 3 237

12 Uber Standard Therapy Results for Configuration 3 239

13 Results for Configuration 3 with Direct Pathogen Update 240

14 Results for Configuration 4 241

15 Results for Work in Progress 244

LIST OF FIGURES

1 Basic endotoxin administration scenarios. 18

2 Numerical results of simulations following scenario 1 in Table 2. 23

3 Numerical results of simulations following scenarios 2a–2c in Table 3. 24

4 Numerical results of simulations following scenarios 3a–3c in Table 4. 25

5 Numerical results of simulations following scenario 4 in Table 5. 26

6 Numerical results of simulations based on scenario 5 in Table 6. 28

7 Numerical results of simulations for sub-lethal potentiation scenario. 29

8 Numerical results of simulations for lethal potentiation scenario. 30

9 Instantaneous versus continuous endotoxin administration. 32

10 Endotoxin tolerance (based on scenario 1) illustrated with the N*-CA phase plane. 37

11 Potentiation (based on scenario 6) illustrated with the N*-CA phase plane. . 38

12 Rescue scenario in the N*-D-CA phase space. 39

13 Dependence of tolerance on preconditioning dose timing and magnitude. . . . 42

14 Initial discrete circle around the Saddle Point 55

15 The half plane perpendicular to the previous circle at point, r 56

16 Example trajectories illustrating the shooting method for finding the next point on the plane in the mesh of the stable manifold 56

17 The first two bands of the stable manifold, connected via a triangular mesh . 57

18 A view of the stable manifold of the origin in the Lorenz system 58

19 A view of the stable manifold of the origin in the Lorenz system 59

20 A closeup of the mesh of the Lorenz manifold 60

21 The original trajectory and the curve of all possible shifted points. 112
22 Illustration for Proposition 1 114
23 Illustration for Theorem 2 115
24 Illustration of the case when the original and competing trajectories converge monotonically in the first component and lie on the same solution curve. . . 122
25 Illustration for Proposition 10 124
26 Regions in the first quadrant for all relevant eigenvector configurations 140
27 Eigenvector configurations 141
28 Eigenvector configuration (a) with the first quadrant divided into three regions. 143
29 Eigenvector configuration (a) with an arbitrary initial condition labeled in Region 1a. 143
30 Eigenvector configuration (b) with the first quadrant divided into three regions. 146
31 Eigenvector configuration (b) with an arbitrary initial condition labeled in Region 1b. 147
32 Eigenvector configuration (b) with an arbitrary initial condition labeled in Region 2b. 148
33 Eigenvector configuration (b) with an arbitrary initial condition labeled in Region 3b. 149
34 Eigenvector configuration (c). 152
35 Eigenvector configuration (c) with an arbitrary initial condition labeled in Region 1c. 152
36 Eigenvector configuration (c) with an arbitrary initial condition labeled in Region 2c. 153
37 Eigenvector configuration (c) with an arbitrary initial condition labeled in Region 3c. 154
38 Eigenvector configuration (d). 157
39 Eigenvector configuration (d) with an arbitrary initial condition labeled in Region 1d. 157
40 Eigenvector configuration (d) with an arbitrary initial condition labeled in Region 2d. 158

41 Linear Example 1 160
42 Linear Example 2 161
43 Linear Example 3 162
44 Linear Example 4 163
45 Linear Example 5 164
46 Linear Example 6 165
47 Linear Example 7 166
48 The isoclines for the linear system given in Example 8. 173
49 Isoclines and shifted points in Example 8. 175
50 Regions I-IV defining the relationship between the coefficients of the analytic solutions of original trajectory and the competing trajectory in Example 8. . 177
51 The lines that form the regions I-IV, overlayed on the content shown in Figure 49. 178
52 Isoclines for Example 9. 180
53 Isoclines and shifted points in Example 9. 181
54 Numerical evidence of tolerance in Example 9. 183
55 Important features of Example 10. 184
56 Isoclines for Example 11. 185
57 Isoclines and shifted points for Example 11. 187
58 Isoclines and trajectories for Example 11. 189
59 Numerical results from the application of the tolerance algorithm to Example 11. 189
60 Comparison of numerical results and the results acquired from the isocline method and the linear method for Example 8. 194
61 A sample simulation from a simulation incorporating four therapies. 216
62 The scatterplot showing the distribution of patient outcomes with respect to pathogen growth rate and the inital value of pathogen. 218
63 An example of a standard therapy dose profile. 223
64 An example of a simulation under Configuration 1 with an aseptic outcome for the placebo case. 226

65 Example of Configuration 1, where the septic response in Figure 67 is modulated with therapy from the NMPC algorithm. 227

66 The simulation shown in Figure 65 is rerun with a heavy weight on Damage. 228

67 An example of a simulation under Configurations 1 and 2 with a septic outcome for the placebo case. 229

68 The NMPC algorithm is run on he septic scenario under Configuration 1 shown in Figure 67. 230

69 Th simulation run in Figure 68 for the septic scenario under Configuration 1 is rerun with a significant weight increase on Damage. 231

70 An example of a simulation under Configuration 2, where the septic response presented in Figure 67 has been modulated with therapy fromt the NMPC algorithm. 233

71 40 hour intervention time for NMPC simulation run on septic scneario of Figure 67. 235

PREFACE

I would like to extend my thanks to the following people who have played a direct role in the completion of my thesis:

- **Jonathan Rubin** - my primary advisor, who stepped in to that role after a job transition that took my initial advisor away from Pittsburgh. Thank you for working together to make the switch over a smooth process, keeping me connected to the projects in which I was already involved. Thank you as well for helping to continue my support as a GSR through the years and for your advice and opinions on matters with which a graduate student has limited experience. Lastly, I often smile thinking of some of our meetings, where afterward I knew that I had probably talked too fast and too much, possibly leaving your head spinning from the rush of many trains of thoughts that just flew by. Your patience to endure my loquaciousness is, I'm sure, a quality befitting a saint! (Although, perhaps you knew what you were getting yourself into after overhearing me teach a calculus recitation that first year.)
- **Carson C. Chow** - my initial advisor and co-advisor, who first opened the door for me into the world of bio math and encouraged me to step through, supporting my early research generously with equipment and supplies whenever needed. Thank you for extending the option of and entrusting me with the wonderful opportunity of a GSR appointment in the very early stages of my graduate career. I have always been grateful for that privilege and I attribute it to one of the things that enabled me to continue along this path without crashing and burning.
- **Gilles Clermont** - a member of my thesis committee, colleague, and friend, whose undying optimism always chose to speak about things that were not as though they were (e.g."Dr. Day"). I have enjoyed collaborating with you and look forward to future

opportunities. You have also been responsible for my continued support as a GSR and I have been thankful for this continued support. In addition, you and Anne-Marie have been a great encouragement to me. I will always cherish the excursions after conferences that I was able to make with you and A-M in Germany. You both have been so generous and provided me with such great opportunities. Thank you!

- **Yoram Vodovotz** - a member of my thesis committee, one of my most enthusiastic colleagues, and a great conversationalist, who has continued to keep me connected to the big picture concerning our research and whose optimism has kept me hopeful about the possibilities. Thank you for your encouragement and willingness to offer opportunities to me to develop my professional path and for your advice along the way. I look forward to working toward those possibilities about which you have remained fervent.
- **Beatrice Riviere** - a member of my thesis committee, whose encouraging words during the stressful finishing stages lifted some of the weight off of my shoulders. I appreciate your willingness to serve on my committee and thank you for bringing a constructive, positive presence to the table.

I would also like to thank my family and friends for their support, encouragement, and love throughout the years:

- My parents, who have patiently waited a long time to see me finally finish school 'for good.'
- My in-laws, who have abundantly poured their love into my life for the 6 wonderful years that I have known them so far.
- My sister, Jennifer, who is one of my closest friends since we stopped borrowing each other's clothes and grew up enough to realize what a gift it is to have a sister
- Gabrielle and Ray Kardohely - faithful friends from day one!
- My brother, Eric, who let me play with his Legos and matchbox cars when we were kids, which probably helped develop some of the skills that enabled me to get this far
- My sister-in-law, Jenni, who is a good friend and who leaves the most amusing messages on our answering machine

- My brother-in-law, Jacob, whose G-mail chats were always a most welcome interruption
- Angela Reynolds, who has worn many hats for me: good friend, colleague, office mate, collaborator, GSO copresident, walking partner along with a dozen others
- Jyotsna Didwadkar, who has been a joy to get to know and whose vocabulary I highly envy
- Stephanie Hoogendoorn, who has showed me a smile and given time to entertain my ramblings from the very first day we met
- Chris and Bojana Jones, who are great, fun friends and who have greatly empathized with me during all the final stages of the creation and submission of this thesis
- Jerry Day, who I'm taking with me when I leave

1.0 INTRODUCTION

1.1 USING MATHEMATICAL AND ENGINEERING TOOLS TO STUDY THE ACUTE INFLAMMATORY RESPONSE

The use of tools from mathematics and engineering is becoming more widespread for gaining insight into biological phenomena and making advances in the medical industry by assisting with practical solutions to clinical challenges in the treatment of patients. The "mathematical biology" community is no longer simply made up of mathematicians looking for interesting mathematical problems arising from biological phenomena. Crossdisciplinary research groups, consisting of mathematicians and engineers as well as clinicians and experimentalists are no longer the exception, with many realizing that research is most effective when there are different yet complementary areas of expertise involved. This is certainly the case with respect to the research that has focussed on understanding the dynamics and complexities of the acute inflammatory response.

1.1.1 Background

The initial response of the body to acute biological stress such as bacterial infection or tissue trauma is an acute inflammatory response. This response involves a cascade of events mediated by a large array of cells and molecules that locate invading pathogens or damaged tissue, alert and recruit other cells and effector molecules, eliminate the offending agents, and restore the body to equilibrium. [55] Ideally, the inflammatory response works effectively in this manner; however, there are times when inflammation can rise out of control, leaving the body susceptible to massive tissue damage, eventually leading to multi-organ failure and death. [70, 56, 103, 107]

Inflammation, while necessary for successful eradication of harmful agents, can cause more inflammation via a positive feedback loop. The cells and molecules that initially respond can cause damage, by either crowding an area or as a result of the release of molecules such as nitric oxide and superoxide intended to destroy foreign entities. [83, 57, 7] Consequently, damaged cells send signals upon death that alert the body of danger, since offending agents are typically the cause of damaged tissue. This, in turn, brings about the recruitment of more inflammatory mediators that attempt to help remedy the situation, causing a necessary yet, potentially, dangerous positive feedback loop. [53, 74]

There are anti-inflammatory mediators present to help control inflammation; however in cases of severe inflammation, these may be rendered ineffective. In addition, anti-inflammatory mediators might be initially suppressing inflammation and working against the purpose of eliminating the offender, allowing it, instead, to grow uncontrolled and destroy more tissue. [14] Ideally, an optimal strategy exists between pro- and anti- inflammatory mediators for eliminating the offending agent, while not accruing excessive tissue damage. This ideal, however, is a delicate balancing act. [93, 94]

Laboratory experiments have shed enormous light on various signaling pathways and mediators responsible for inflammation. [85, 90] However, the highly nonlinear, systemic nature of the problem has made it extremely difficult to gain understanding of the process as a whole from experimentation alone, much less generate effective strategies to correct immune dysfunction. As noted in the paper by Vodovotz et al, the limitations of animal models have reduced results to highly specific information available at only a few time points, making it difficult to interpret the results with respect to the global problem. [112] Therapies that have been found to be successful in such experiments have mostly been unable to improve survival in large clinical trials. [15, 71]

One reason for a lack of effective therapies in combatting uncontrolled inflammation is due to the fact that most were designed to target one specific mediator of inflammation. However, mediators that may be the cause of inflammation during one phase of the response might be completely irrelevant at a later point in time. In addition, the inflammatory response to an insult is dependent on a number of patient factors, including age, genetic predisposition, and gender. Clinical manifestations of an insult might be very similar in different individuals, yet treatment might help one and not the other. During the course of an inflammatory response to an insult, a single patient at different times can experience being hyper-inflamed or immunosuppressed. Hence, the timing of events in the inflammatory cascade is crucial to understanding when and how a therapy will be effective. In addition, the need for therapies tailored for individual patients, as well as therapies that target multiple mediators, is apparent. [13, 26]

In 2004, the U.S. Food and Drug Administration (FDA) reported that "A new product development tool kit–containing powerful new scientific and technical methods such as animal or computer-based predictive models, biomarkers for safety and effectiveness, and new clinical evaluation techniques–is needed to improve predictability and efficiency along the critical path from laboratory concept to commercial product." [39] The use of mathematical models, which can be subject to a countless number of numerical experiments, has allowed for unhindered exploration of the acute inflammatory response in general, as well as specific problems that can arise in a response. In addition, the cost of using these tools are miniscule compared to the cost of wet lab experiments. This is not to assert that traditional laboratory experimentation is obsolete and of little value, but, rather, to emphasize again the importance of using as many available resources as possible. Mathematical models are not perfect and need verification and validation from classical experimental studies.

It has been affirmed and reaffirmed that a systems approach to understanding inflammation is imperative. [16, 39, 109] Highlights from some of the prior and current research using mathematical models for exploring the intricacies of the acute inflammatory response are now discussed.

1.1.2 Prior Research and Current Directions

In general, mathematical modeling of biological systems, especially with ordinary differential equations, is not novel. Mathematical models of the immune system have appeared in literature in some form for more than a decade from the date of this writing (see, for example, [75]) and there are a plethora of models describing the pathogenesis of any number of diseases, such as HIV, tuberculosis, and cancer. [61, 92, 88] There are also plenty of books, some dating from the 1970s about mathematical modeling in biology and the subsequent analysis of them. [80, 81, 34, 99] Even so, with each new biological discovery and with the wealth of data that has been acquired from experiments, there is still plenty of room for new and improved models. In particular, only recently have clinical researchers begun to understand the complex ubiquitous role that inflammation plays in the pathogenesis of many different diseases. The push to understand this systemic process has emerged as a major focus of research groups worldwide. In fact, the *Society for Complexity in Acute Illness* (*http://www.scai-med.org/*) was formed specifically to bring together "clinicians, bench scientists, and modelers" from "hospitals, research institutions, and companies" to bridge the gap between the discoveries made under the microscope and the patient in the intensive care unit.

The primary approaches to modeling inflammation have consisted of modeling with ordinary differential equations (ODE), partial differential equations (PDE), and agent based modeling (ABM). This dissertation joins a substantial body of work that has sought insight and answers to inflammation related problems. In 2004, a review of some ODE and ABM models of inflammation was made by Vodovotz et al. [113]. In this review, the strengths and weaknesses of both types of models are given, along with some examples. [4, 5, 21, 22] ODE models, while conducive to rigorous mathematical modeling, can become very complex as the number of equations increases, making them difficult to calibrate and validate with experimental data. ABM models, which lend themselves to being easily constructed and interpreted, especially for nonmathematicians, can be computationally expensive to simulate with the number of agents needed to represent the system. It is also difficult to analyze the mechanisms of such a system. As a result, validation can be difficult to carry out on such models. However, in both cases progress has been made toward finding ways to improve the accuracy and prediction power of these models.

A more recent and comprehensive review by Vodovotz [110] is an update discussing modifications and improvements to the models previously discussed as well as new ODE and ABM models that deal with various aspects of inflammation. The models described in [20, 112, 114] are large scale ODE models consisting of 15-31 differential equations, describing the interactions of many of the cells, cytokines, and other molecules that play a role in the inflammatory cascade. These models may have several different instigators of inflammation, such as pathogen (bacteria), bacterial Lipopolysaccharide (LPS or endotoxin), trauma, or hemorrhagic shock. Because of the high detail of these models, they can be used for qualitative as well as quantitative insights. The many parameters that exist due to the large number of equations are mostly determined via methods that fit the model dynamics to existing animal data. Typically, the model is fit to a set or two of data for particular scenarios (for instance, an initial load of 3mg/kg or 6mg/kg of endotoxin) and then used to predict a separate set of data for a different scenario (an initial load of 12 mg/kg endotoxin) as validation of the model's accuracy and prediction capability.

The models presented in [77, 69] are ABM models of inflammation and wound healing used in the setting of diabetic foot ulcer pathology and biomechanical stresses related to phonotrauma. These models qualitatively reproduce behaviors and make predictions consistent with the literature and recent experimental findings. Another model under construction by Reynolds et al. explores the effect of inflammation on gas exchange in the lung, using a combination of partial differential equations and ordinary differential equations. In this model, tissue is treated as a separate spatial compartment. This allows for the exploration of the effects of inflammation on the diffusion of gas molecules in the context of an infection that first started either in the blood or in tissue. To simplify the computational aspects of the model, the equations representing the gas molecules were reduced to ordinary differential equations. [97]

The models presented in [65, 98, 28] are small scale ODE models containing 3, 4 and 4 differential equations, respectively. These models qualitatively, rather than quantitatively, reproduce biologically observable behavior of the inflammatory response to either pathogen [65, 98] or endotoxin [28]. The variables or mediators that are modeled in these examples represent general characteristics of several actual mediators rather than specific cells or molecules. For instance, one variable might be labeled the "early pro-inflammatory" mediator, which could be representative of a host of cells and molecules that have pro-inflammatory effects early in the course of an inflammatory response to an insult. Even though generalizations like this are made, these models have not only qualitatively reproduced the differing clinical and experimental outcomes that are seen with mild to serious infections, but have added insight into the nature and behavior of inflammation for various insults.

In [65], a three equation model was presented which included an equation for the inflammatory instigator, pathogen, and two equations for inflammatory mediators, one directly incited by pathogen and the other indirectly incited by the first pro-inflammatory mediator in a positive feedback loop. Based simply on varying the initial magnitude of the pathogen insult, three different outcomes were possible in the model:

- healthy: all mediators resolve to baseline levels
- aseptic: pathogen is eliminated, but the other mediators remain elevated
- septic: All mediators as well as pathogen remain elevated

Analysis of the model led to suggestions of possible therapeutic interventions to correct an aseptic or septic outcome and the insight that "the clinical condition of sepsis can arise from several distinct physiological states, each of which requires a different treatment approach."

In [98], a four equation model was constructed based on the idea founded in [65] of using general variables to represent characteristics of the inflammatory response. However, not only was a fourth equation representing an anti-inflammatory mediator added, but a model was constructed anew using an approach involving the creation of simple subsystems between the mediators. For instance, the subsystem consisting of pathogen and the early pro-inflammatory mediator was constructed to ensure the desired bistability of an excitable system (i.e. health and death states would be both possible based on initial magnitude of the insult). This model also qualitatively reproduced the different clinically observable outcomes mentioned above based on the magnitude and strength of the pathogen insult.

Moreover, with the addition of an anti-inflammatory mediator, the authors were able to highlight "the importance of dynamic antiinflammation in promoting resolution of infection and homeostasis." In addition, a simple therapeutic intervention involving the increase or decrease of the anti-inflammatory levels in the model during the course of the infection was explored. The authors showed that in some cases an addition of anti-inflammation would be beneficial, while in other cases it could be problematic and even detrimental to recovery. Also shown, was that the ability to remove anti-inflammation could be beneficial in cases where a pathogen was particularly virulent and the patient would need maximum host defense to overcome the infection. Consequently, it is obvious that there is a need for a systematic approach to finding the proper type, timing, and dosage amount of therapy. This is the subject of Chapter 5 of this dissertation where nonlinear model predictive control, a tool typically used in engineering applications, is utilized to find appropriate ways to administer different types of therapy, both pro- and anti-inflammatory.

In [28], the model from [98] is slightly modified to have endotoxin as the inflammatory response instigator instead of pathogen. As will be discussed in Chapter 2, endotoxin is a highly conserved, highly immunogenic, constituent molecule of the outer cell wall of Gram-negative bacteria. When bacteria are lysed by immune effector cells and molecules, surges of endotoxin may be released into the host, intensifying the inflammatory response and causing further activation of immune effector cells [2, 54]. It has been observed that in some instances repeated doses of endotoxin result in a considerably less vigorous immune response, a phenomenon referred to as endotoxin tolerance [10]. In fact, the induction of tolerance can greatly blunt the effect of a dose of endotoxin that would be lethal to a naïve animal. There is also the more intuitive observation that repeated doses of endotoxin result in a more vigorous immune response (potentiation). In this work, we suggest via the mathematical model that endotoxin tolerance is a direct result of the dynamic interactions between components of innate immunity, rather than a specific, distinct phenomenon.

1.2 INTERESTING BIOLOGY-INSPIRED MATHEMATICAL PROBLEMS

Although mathematical biology has become much more application focussed, there are still interesting mathematical problems that arise in the study of biological systems, which may not necessarily have direct application to the biological problem from which they originated. In other words, although the subsequent research may add some insight into the biology or assist in visualization and understanding, the results might be much more interesting and significant from a mathematical perspective. Two such problems originated from the research discussed in Chapter 2 with respect to endotoxin tolerance, both of which fit nicely into dynamical systems theory.

1.2.1 Manifolds

In Chapter 2, an illustration is presented showing a type of threshold or "dividing line" between two different steady states of the system, representing two very different outcome: life and death, or healthy and unhealthy. This divider is formally known as the separatrix and it consists of the stable manifold of the saddle point in a particular phase space. Initial conditions that begin on one side of this structure will evolve to the healthy state, while those beginning on the other side approach the unhealthy fixed point. The fascinating aspect of this structure is that the future trajectory of any initial condition is completely determined by which side of the manifold the initial condition lies.

If one considers this as a representation of what happens to a patient in the absence of any therapy, the fate of the patient can be determined solely from the initial starting point. Unfortunately, in practice, it is virtually impossible to know a patient's "starting conditions" since the manifestations of an infection do not always appear immediately, not to mention that measurements of crucial entities such as pathogen concentration and growth rate are impossible to acquire. Hence, this tool, while instructive for illustration purposes, holds little direct application to helping predict patient outcome.

However, the ability to numerically generate this structure is an extremely interesting dynamical systems problem and one that many have successfully solved. [30, 31, 32, 33, 44, 46, 58, 63] Unfortunately, there is no software or code currently available. Hence, the focus of Chapter 3 is with respect to the *implementation* of one of these algorithms, due to Krauskopf and Osinga. [63] Although the actual algorithm is not a part of the original content of this thesis, the code implementing the various steps is, since there exist several key computational challenges that arise for some of the steps in the algorithm that are not explicitly discussed in the original authors' paper.

1.2.2 Endotoxin Tolerance from a Purely Mathematical Perspective

Endotoxin tolerance as well as its counterpart, potentiation, are well established and thoroughly explored issues in the biological and experimental literature. (See [10, 27, 102, 119]) As mentioned previously, Chapter 2 discusses a model that reproduces many scenarios that exhibit endotoxin tolerance. One of the model mediators represents the tolerance variable, or the component that experiences a blunting (or potentiating) affect to repeated doses of endotoxin. Tolerance can be seen when time courses of this mediator for single dose simulations (original) versus repeated dose simulations (competing) are compared. When overlapped on a graph, the competing time course which initially starts at a higher value in this component, will at some time point manage to descend to a lower value in this component, compared to the original single dose time course. In essence, it can be thought of as a type of race.

This idea, brought up during the course of researching endotoxin tolerance with this ODE model, made it evident that the tolerance behavior may not be unique to this one system of equations. In fact, the problem deals with transient behavior of a dynamical system, something not very well developed in the theory. In order to pursue this idea in a purely dynamical systems context, the concepts of tolerance and potentiation are formalized mathematically.

From this, statements and theorems are made regarding the existence of tolerance in two-dimensional linear and non-linear systems. The 2D linear case has been characterized completely. The 2D nonlinear case is much more complicated than the linear case and requires substantial creativity because analytical solutions are not generally available for nonlinear systems and tools for studying transients are not well developed. Nevertheless, significant progress has been made toward pinpointing when tolerance can be exhibited in a system. These ideas and results are presented in Chapter 4.

1.3 ORGANIZATION OF THESIS

The chapters of this thesis are organized as follows:

- Chapter 1 is dedicated as this introduction.
- Chapter 2 presents the ODE model of the acute inflammatory response to endotoxin and the results and insights from reproducing various experimental scenarios of endotoxin tolerance.

- Chapter 3 discusses the MatLab computer program used to generate the two-dimensional stable manifold shown in the second to last figure of Chapter 2.
- Chapter 4 introduces the theoretical work done with respect to mathematically formalizing the idea of the tolerance phenomenon that was discussed in an experimental setting in Chapter 2.
- Chapter 5 explores the application of nonlinear model predictive control to finding therapeutic strategies to assist with immunomodulation of the acute inflammatory response during the course of an infection.
- Chapter 6 gives a brief summary of the results acquired, the challenges faced, and the ideas for future extensions and improvements.
- Glossary of Abbreviations

2.0 MODELING ACUTE INFLAMMATION AND ENDOTOXIN TOLERANCE

2.1 INTRODUCTION

The initial response of the body to acute biological stress such as bacterial infection or tissue trauma is an *acute inflammatory response*. This response involves a cascade of events mediated by a large array of cells and molecules that locate invading pathogens or damaged tissue, alert and recruit other cells and effector molecules, eliminate the offending agents, and restore the body to equilibrium. Bacterial lipopolysaccharide (LPS; *endotoxin*) is a highly conserved, highly immunogenic, constituent molecule of the outer cell wall of Gram-negative bacteria. When bacteria are lysed by immune effector cells and molecules, surges of endotoxin may be released into the host, intensifying the inflammatory response and causing further activation of immune effector cells [2, 54]. In fact, the administration of antibiotics can lead to pulses of endotoxin release from Gram-negative bacteria as the antibiotics kill the invading bacteria, confirming the clinical importance of this subject matter[35]. Since direct endotoxin administration in animals and humans can induce an acute inflammatory response that reproduces many of the features of an actual bacterial infection, such as fever, it stands as a valid experimental model for investigating the inflammatory response [23, 79, 91].

High doses of endotoxin can be lethal, even though this bacterial byproduct does not proliferate as a Gram-negative bacteria would [104]. It has been observed, however, that in some instances repeated doses of endotoxin result in a considerably less vigorous immune response, a phenomenon referred to as endotoxin tolerance [10]. In fact, the induction of tolerance can greatly blunt the effect of a dose of endotoxin that would be lethal to a naive animal. A variety of studies have followed up on Beeson's initial reports of endotoxin tolerance (for a historical perspective see Cross [27]; Schade [102]; West and Heagy [119]). Experimentally, it is now possible to assess the activation status of inflammatory cells or the levels of signaling proteins, such as cytokines, in organs or the blood as direct measures of inflammation [82, 84]. The cytokine *Tumor Necrosis Factor-α* (*TNF*) in blood serum, for instance, has become a prominent marker of inflammation [55, 101]. Thus, observing that the concentration of this cytokine is lower than levels normally observed after endotoxin administration suggests that inflammation is being suppressed.

Interestingly, the inverse phenomenon, called potentiation, has also been observed. In the extreme, an otherwise non-lethal dose of endotoxin rapidly following another non-lethal dose can result in death [18]. We hypothesized that a simple mathematical model of the acute inflammatory response could reconcile tolerance and potentiation, on the premise that the observed outcomes result from dynamic interactions between components of innate immunity. Accordingly, we adapted a recently developed computational model of the inflammatory response [98] and simulated various scenarios involving repeated endotoxin administration. We use actual experimental mouse scenarios to guide in silico experiments that recreate these scenarios qualitatively, including the phenomena of endotoxin tolerance and potentiation.

In our simulations, we find that both the timing and magnitude of endotoxin doses, relative to each other and to the dynamical interplay between pro- and anti-inflammatory mediators, are central in discriminating between the seemingly disparate phenomena of endotoxin tolerance and potentiation. Our results, derived from a mathematical model not constructed specifically to address the issue of preconditioning, support the perspective that endotoxin tolerance and related phenomena could be better explained and understood as "inflammatory-stimuli-induced" effects rather than specific, distinct phenomena [18]. This perspective is also supported by studies showing that various inflammatory stimuli (e.g. trauma, hemorrhage, cytokines) can act either to tolerize or to prime the host for subsequent homologous or heterologous stimuli [17, 19, 59, 60, 67, 76, 115, 125]. The intent here is not to carry out a detailed mathematical analysis of our model. Rather, we hope to argue convincingly that endotoxin tolerance, potentiation, and other phenomena related to repeated endotoxin administration are best viewed and understood via the acute inflammatory response [23, 122] and to demonstrate this with a mathematical model of that response.

2.2 A MATHEMATICAL MODEL OF THE ACUTE INFLAMMATORY RESPONSE TO ENDOTOXIN

To examine repeated endotoxin administration in the context of the acute inflammatory response, we use a mathematical model that incorporates the effects of key aspects of the immune system's response to an insult (Eqs. 2.1-2.4). The detailed derivation of this model, based on previous experimental findings, and a term-by-term explanation of its components are outlined by Reynolds et al. [98]. The model we use replaces the pathogen equation of Reynolds et al. with an endotoxin equation. These changes introduce several different parameters that replace or add to those used in Reynolds et al. These include μ_{pe} (1/h), k_{npe} (mg/kg/h), λ_i (mg/kg), t_i (h), and δ (h) which are described in Table 1. However, all other equations and parameter values have been maintained to agree with those presented in Reynolds et al. A substantial number of these parameters were obtained from existing experimental literature (Table 1). For more information on parameter acquisition and estimation, please see Section 2.7: Supplementary Materials.

This model consists of a system of ordinary differential equations containing two pro-inflammatory mediators, N^* and D, as well as an anti-inflammatory mediator, C_A. N^* is biologically comparable to phagocytic immune cells or early, typically pro-inflammatory cytokines, such as *TNF* and *Interleukin-1* (IL-1). The other pro-inflammatory variable, D, not only serves as a marker for tissue damage/dysfunction, but also as a positive feedback into the earlier pro- and anti-inflammatory arms of the system, as damaged (e.g. injured or necrotic) tissue would [74]. The anti-inflammatory mediator, C_A, acts on a slower time scale than N^*. For instance, C_A behaves more like the cytokine *Transforming Growth Factor-β1* (TGF-β1) rather than *Interleukin-10* (IL-10). However, it could also represent other typically anti-inflammatory mediators such as *cortisol*. In Section 2.4, we discuss the importance of dynamically modeling D and the necessity for the anti-inflammatory mediator to posses certain qualitative properties for tolerance to occur in the model.

Units for N^*, C_A and D are not given explicitly because there is no single biological entity or marker that these variables represent and thus there are no specific units that can quantify these variables empirically. Hence, we use "N^*-units," "C_A-units," and "D-units" because we cannot be any more precise about them. Although C_A (Anti-inflammatory Mediator) has characteristics of IL-10 and TGF-β, it would be inappropriate to assign real units to this variable and quantitatively compare it to actual data from these or other anti-inflammatory mediators.

The immune response instigator, pathogen endotoxin or P_E (mg/kg), serves as the initial stimulus that recruits N^* with a rate of k_{npe} which has units mg/kg/h. This begins the inflammatory cascade. P_E decays exponentially with rate μ_{pe}, having units per hour, with no other mediators affecting its decay. In addition, multiple intravenous injections (i.v.) of endotoxin can be emulated with Heaviside step functions in the P_E equation. The parameters λ_i and t_i in the Heaviside functions represent the endotoxin dosage load for the i^{th} dose given at time t_i hours, respectively, for $i = 1, 2, ...n$, the number of doses. If we want a total of λ mg/kg to be given over a duration of time, δ, then λ/δ(mg/kg/h)given for δ hours will accomplish this. The parameter δ is set to 0.01 h, which matches the time step of our numerical integration, when we wish to emulate a pulse, or a quick on–off, instantaneous injection. For instance, if $\delta = 0.01$, the administration of a load amount of 3 mg/kg given at time t hours would stop at $t + \delta = t + 0.01$ h, thereby essentially giving the whole load all at once. Larger values of δ will result in longer infusion times. For example, in scenario 8 we set δ equal to 24, thereby giving $3/24 = 0.125$ (mg/kg/h) continuously over the span of 24 h. This also gives a total of 3 mg/kg but over a longer span of time than the instantaneous injection. Although we model i.v. type injections, studies have shown that endotoxin administration given either intravenously or intraperitoneally invokes a similar inflammatory response [23]. Table 1 gives the parameter values that were established for this model, which is represented by Eqs. 2.1-2.4.

$$\frac{dP_E}{dt} = -\mu_{pe}P_E + \sum_{i=1}^{n}\frac{\lambda_i}{\delta}S(t_i, t_i+\delta), \tag{2.1}$$

$$\frac{dN^*}{dt} = \frac{s_{nr}R}{k_{nr}+R} - \mu_n N^*, \tag{2.2}$$

$$\frac{dD}{dt} = k_{dn}\frac{f(N^*)^6}{x_{dn}^6 + f(N^*)^6} - \mu_d D, \tag{2.3}$$

$$\frac{dC_A}{dt} = s_c + k_{cn}\frac{f(N^* + k_{cnd}D)}{1 + f(N^* + k_{cnd}D)} - \mu_c C_A \tag{2.4}$$

Table 1: Model parameter names and values used in simulations

Name	Range	Value Used	Description	Sources
μ_{pe}	0.6207–14.85	3/h	Decay rate of pathogen endotoxin (P_E)	[51, 117, 123]
λ_i	n/a	Various (mg/kg)	Amount of the i^{th} P_E dose administration	
δ	n/a	0.01 or 24 hr	Duration of PE injection: 0.01 corresponds to instantaneous delivery (1/100 of an hour) and 24 corresponds to constant delivery of a dose over 24 hours.	
t_i	n/a	Various (h)	Time at which the ith PE dose is given	
k_{npe}	Estimated	9/(mg/kg)/h	Activation of phagocytes by pathogen endotoxin (PE)	
k_{nn}	Estimated	$0.01/N^*$-units/h	Activation of phagocytes by already activated phagocytes (or the cytokines that they produce)	
s_{nr}	Estimated	$0.08/N_R$-units/h	Source of resting phagocytes	
μ_{nr}	0.069-0.12	0.12/h	Decay rate of resting phagocytes (macrophages and neutrophils)	[25]
μ_n	Less than μ_{nr}	0.05/h	Decay rate of activated phagocytes (macrophages and neutrophils)	[25]
k_{nd}	Less than k_{npe}	$0.02/D$-units/h	Activation of phagocytes by tissue damage (D)	[6]
k_{dn}	Estimated	$0.35/D$-units/h	Max rate of damage production by activated phagocytes (and/or associated cytokines/free radicals)	
x_{dn}	Estimated	0.06 N^*-units	Determines level of activated phagocytes (N*) needed to bring damage production up to half its maximum level	
μ_d	0.0174 (min)	0.02/h	Decay rate of damage; combination of repair, resolution, and regeneration of tissue HMGB-1 release by damage	[29, 116]
c_∞	Estimated	$0.28/C_A$-units/h	Threshold for effectiveness of the anti-inflammatory response	[50]
s_c	Estimated	$0.0125C_A$-units/h	Source of anti-inflammatory (CA) (IL-10, TGF-β1, cortisol);	
k_{cn}	Estimated	0.04 C_A-units/h	Maximum production rate of Anti-inflammatories	
k_{cnd}	Estimated	48 N^*-units/D-units	Controls relative effectiveness of activated phagocytes versus damage in producing anti-inflammatories	
μ_c	0.15-2.19	0.1/h	Decay rate of the anti-inflammatory mediator	[8, 12, 40, 49]

where n is the number of doses in the experiment and the other functions in 2.1-2.4 are given by

$$
\begin{aligned}
R &= \frac{(k_{npe}P_E + k_{nd}D + k_{nn}N^*)}{1+(C_A/c_\infty)^2}, \\
f(x) &= \frac{x}{1+(C_A/c_\infty)^2}, \\
S(t_{on}, t_{off}) &= H(t-t_{on}) - H(t-t_{off}) \\
&= \begin{cases} 0 \text{ if } t < t_{on} \\ 1 \text{ if } t \geq t_{on} \end{cases} - \begin{cases} 0 \text{ if } t < t_{off} \\ 1 \text{ if } t \geq t_{off} \end{cases}
\end{aligned}
$$

Using the parameter values given in Table 1, this system has three possible equilibrium states in the regime that we are interested in, namely where all solutions are nonnegative. Two of the three fixed points are stable and the remaining one is a saddle whose stable manifold separates the phase space of interest into two regions, each containing one of the stable fixed points. One of the stable states is specified by the background levels of the variables, $(P_E, N^*, D, CA) = (0, 0, 0, C_{A0})$. These low levels are characteristic of the state in which the system is at baseline, prior to any perturbation. Thus, when the mediators settle to this state we correspondingly interpret the outcome as healthy. The other stable equilibrium is classified as an unhealthy state in light of the fact that the values of the variables at this state are above background levels, except for P_E, which always decays asymptotically to zero. When the mediators are pulled to this state it indicates that the response has not properly resolved and, consequently, the outcome is unhealthy or inflamed.

The observations we make in our simulations have biological interpretations related to the characteristics of the acute inflammatory response. When we emulate an administration of endotoxin, the variables of the model react much like the mediators of the inflammatory response in the body in the presence of endotoxin, with their levels rising in the presence of this pro-inflammatory stimulus. After we induce this inflammatory response in our model with an injection of P_E, the system either settles to the healthy state or rises to the unhealthy state. If the dosage of P_E is large enough, it can elicit such a response that the system remains inflamed and is unable to return to its background levels. We equate such an outcome with persistent inflammation, which is an unhealthy endpoint. Given these features of our system, we interpret the existence of endotoxin tolerance in our model as a reduction in the response of N^* to a low dose of P_E after the system is preconditioned with an initial low P_E dose. Likewise, if a preconditioning dose of P_E prevents the system from ending up at the unhealthy state when an otherwise unhealthy dose is given, we infer this as the ability of the model to display protection from mortality. Using the model Eqs. 2.1-2.4, with parameter values from Tables 1 and 7, we are able to qualitatively reproduce the results of various published scenarios of repeated endotoxin administration, which we now discuss.

2.3 MODEL SIMULATIONS OF EXPERIMENTAL SCENARIOS ENDOTOXIN TOLERANCE SCENARIO

For our in silico simulations, we emulate the scenarios below using the dynamical systems analysis software XPPAUT [36]. Eqs. 2.1-2.4 are integrated numerically using the Runge–Kutta algorithm with step size 0.01 for 200 time units (hours), taking into account the simulated i.v. injections of P_E at the specified times. Thus, the design of our in silico endotoxin simulations can closely resemble actual endotoxin experimental scenarios, which originally were carried out with mice. The XPPAUT code for this model is included with the Supplementary Materials in section 2.7.

We start with the reproduction of proper responses to survivable and lethal endotoxin doses, simulated by simply varying the load (λ_1 mg/kg) of P_E at time zero ($t_1 = 0$ h). Regarding endotoxin administration and mortality, it is generally accepted that doses at or above 17 mg/kg cause a high mortality rate in mice [24]. Figs. 1a and b show the results of the model simulations carried out with low (Fig. 1a) and high (Fig. 1b) P_E doses. Having established these basic responses, we now consider experiments involving repeated endotoxin administration, most of which are based on experimental data in the literature.

2.3.1 Endotoxin tolerance scenarios

Published studies report that endotoxin tolerance can be induced in various ways, generally involving the administration of low, repeated doses of endotoxin over periods of time ranging from one day to a week [9, 11, 96, 105, 120]. Blood serum is collected at some time after the

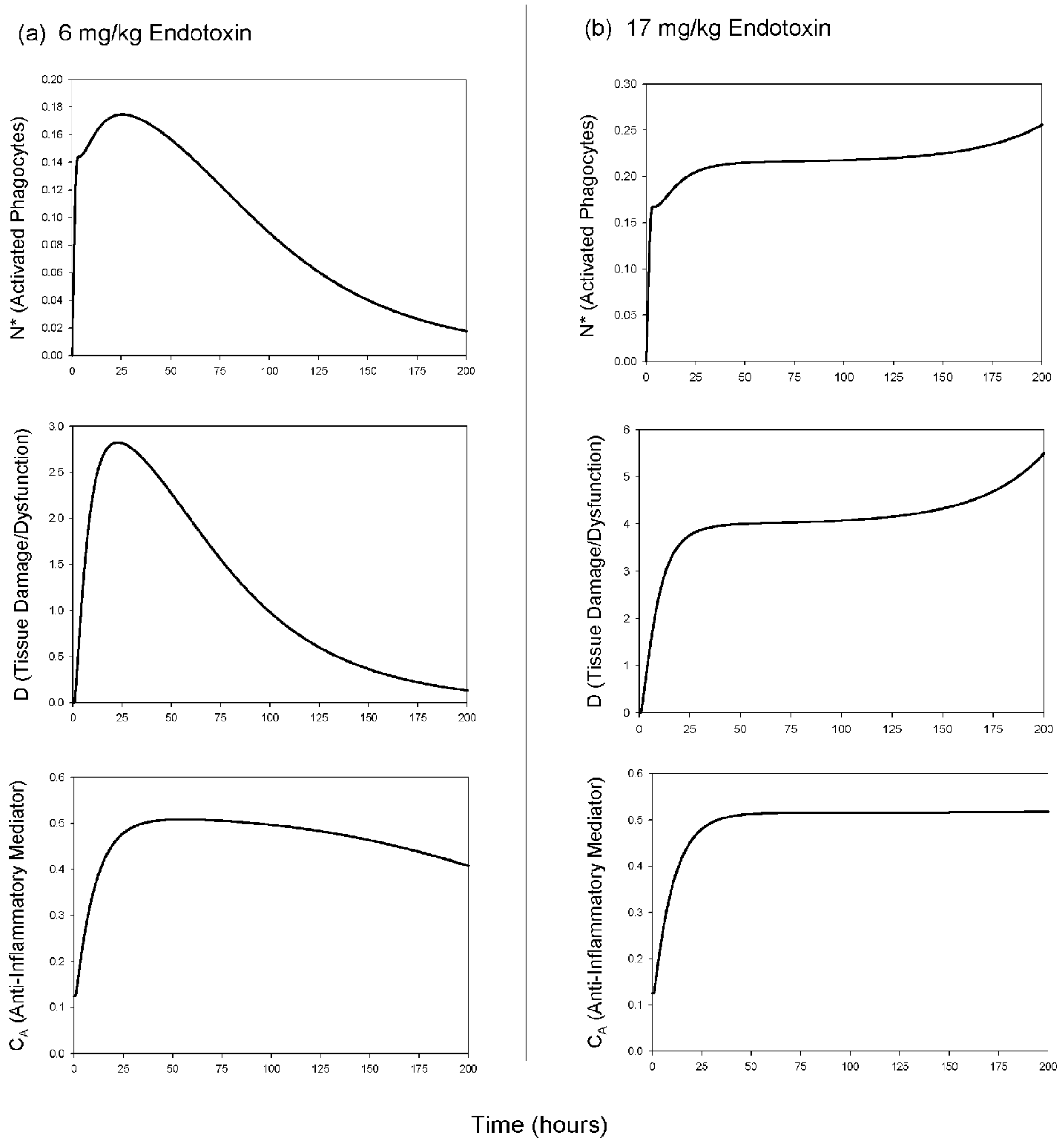


Figure 1: Basic endotoxin administration scenarios. The values of the parameters λ_1, t_1, and δ are set to simulate a one dose instantaneous ($\delta = .01$) administration of P_E at time zero ($t_1 = 0$). (a) Doses less than $\lambda_1 = 17$ mg/kg of P_E cause a response, but all mediators eventually settle back to baseline in a healthy resolution. Here we show a simulation done with a dose of $\lambda_1 = 6$ mg/kg of P_E. (b) Doses greater than or equal to $\lambda_1 = 17$ mg/kg of P_E cause all mediators to remain elevated, indicating an unhealthy outcome. The simulation results shown are carried out with a dose of $\lambda_1 = 17$ mg/kg of P_E. Time courses for N^*, D, and C_A are shown for both scenarios.

Table 2: Scenario 1 (adapted from the experiments of Sly et al., 2004)

Sly (2004)	**0 hours**	**24 hours**	**Experimental Results**
Non-Preconditioned	*Saline*	*10 mg/kg*	*600 pg/ml TNF @ 27 hours*
Preconditioned	*1 mg/kg*	*10 mg/kg*	*TNF levels very low @27 hours*

Table 3: Scenarios 2a - 2c (adapted from the experiment of Wysocka et al., 2001)

Wysocka (2001)	**0 hours**	**26 hours**	**Experimental Results**
Non-Preconditioned	*Saline*	*100 mcg*	*100 ng/ml TNF @ 27 hours*
Preconditioned 2a	*1 mcg*	*100 mcg*	*$<$ 20 ng/ml TNF @ 27 hours*
Preconditioned 2b	*5 mcg*	*100 mcg*	*$<$ 20 ng/ml TNF @ 27 hours*
Preconditioned 2c	*20 mcg*	*100 mcg*	*$<$ 20 ng/ml TNF @ 27 hours*

Mouse weight is estimated at 20 grams: 1 mcg/mouse = 0.05 mg/kg,
5 mcg/mouse = 0.25 mg/kg,
20 mcg/mouse = 1.0 mg/kg, and
100 mcg/mouse = 5.0 mg/kg

last (challenge) endotoxin dose, and inflammatory analytes (generally *TNF*) are measured. In all the above cited experiments, a reduced amount of *TNF* is seen in the group receiving more than one dose of endotoxin (preconditioned) as compared to the amount of *TNF* found in the serum of mice receiving only a single dose of endotoxin (non-preconditioned).

Scenarios 1–5 closely follow various experimental scenarios of repeated endotoxin administration as they are outlined in the literature. Tables 2-6 summarize the designs and results of these scenarios which are reproduced in our model simulations with respect to a qualitative reduction in our pro-inflammatory mediators, specifically N^*. Scenarios 6–8 are not explicitly found in the literature, yet we believe them to be relevant scenarios that merit consideration. The parameter values appearing in Table 1 are used for all the scenarios discussed in this section, with the exception of parameters that are used to set up i.v. PE administrations for the various simulations: λ_i,t_i, and δ. The values for these parameters as pertains to the different scenarios can be found in Table 7.

Scenarios 1-5 are based on those found in Tables 2-6. As an example, parameters for one simulation may be set as follows: $t_1 = 0$ hrs, $\lambda_1 = 0$ mg/kg, $t_2 = 24$ hrs, $\lambda_2 = 10$ mg/kg, and

Table 4: Scenarios 3a - 3c (adapted from the experiments of Rayhane et al., 1999)

Rayhane (1999)	**0 hours**	**24 hours**	**48 hours**	**72 hours**	**Experimental Results**
Non-Preconditioned 3a	*Saline*	*100 mcg*	*n/a*	*n/a*	*35 ng/ml TNF @ 25.5 h; 2.5 ng/ml TNF @ 27 h*
Preconditioned 3a	*2.5 mcg*	*100 mcg*	*n/a*	*n/a*	*3 ng/ml TNF @ 25.5 h; 2 ng/ml TNF @ 27 h*
Non-Preconditioned 3b	*Saline*	*Saline*	*100 mcg*	*n/a*	*35 ng/ml TNF @ 49.5 h; 2.5 ng/ml TNF @ 51 h*
Preconditioned 3b	*2.5 mcg*	*2.5 mcg*	*100 mcg*	*n/a*	*1 ng/ml TNF @ 49.5 h; .5 ng/ml TNF @ 51 h*
Non-Preconditioned 3c	*Saline*	*Saline*	*Saline*	*100 mcg*	*35 ng/ml TNF @ 73.5 h; 2.5 ng/ml TNF @ 75 h*
Preconditioned 3c	*2.5 mcg*	*2.5 mcg*	*2.5 mcg*	*100 mcg*	*1 ng/ml TNF @ 73.5 h; .5 ng/ml TNF @ 75 h*

Mouse weight is estimated at 20 grams: 2.5 mcg/mouse = 0.125 mg/kg and 100 mcg/mouse = 5.0 mg/kg

Table 5: Scenario 4 (adapted from the experiments of Balkhy et al., 1999)

Balkhy (1999)	**0 hours**	**24 hours**	**48 hours**	**72 hours**	**Experimental Results**
Non-Preconditioned	*Saline*	*Saline*	*n/a*	*300 mcg*	3- to 6-fold reduction in the peak serum TNF-α levels @ 73 hours
Preconditioned	*50 mcg*	*50 mcg*	*n/a*	*300 mcg*	

Mouse weight is estimated at 20 grams: 50 mcg/mouse = 2.5 mg/kg and 300 mcg/mouse = 15 mg/kg

Table 6: Scenario 5 (adapted from the experiments of Berg et al., 1995)

Berg (1995)	**0 hours**	**24 hours**	**Experimental Results**
Non-Preconditioned	*Saline*	*200 mcg*	No mice survived
Preconditioned	*25 mcg*	*200 mcg*	All mice survived

Mouse weight is estimated at 20 grams: 25 mcg/mouse = 1.25 mg/kg and 200 mcg/mouse = 10 mg/kg

Table 7: Endotoxin administration parameter values and figure references for in silico simulations of Scenarios 1-8. Scenarios 1-5 are based on those found in Tables 2 - 6. As an example, parameters for one simulation may be set as follows: $t_1 = 0$ hrs, $\lambda_1 = 0$ mg/kg, $t_2 = 24$ hrs, $\lambda_2 = 10$ mg/kg, and $\delta = .01$ hrs. This is analogous to giving a saline (non-preconditioned) dose ($\lambda_1 = 0$ mg/kg) to mice at time zero ($t_1 = 0$ hrs) and then giving a second dose ($\lambda_2 = 10$ mg/kg) of endotoxin at 24 hours ($t_2 = 24$ hrs) with both doses given as instantaneous injections ($\delta = 0.01$ hrs) at the specified times. The system is then integrated and one can look at time courses of the model variables. These parameters can be changed and the system integrated again to give another set of time course for comparison.

	λ_1	t_1	λ_2	t_2	λ_3	t_3	λ_4	t_4	δ	**Figure**
	mg/kg	*hrs*	*mg/kg*	*hrs*	*mg/kg*	*hrs*	*mg/kg*	*hrs*	*hrs*	**Ref**
Scenario 1										2a-2d
Non-Preconditioned	*0.0*	*0*	*10.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned	*1.0*	*0*	*10.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Scenario 2a-2c										3a-3c
Non-Preconditioned	*0.0*	*0*	*5.0*	*26*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned 2a	*0.05*	*0*	*5.0*	*26*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned 2b	*0.25*	*0*	*5.0*	*26*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned 2c	*1.0*	*0*	*5.0*	*26*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Scenarios 3a-3c										4a-4c
Non-Preconditioned 3a	*0.0*	*0*	*5.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned 3a	*0.125*	*0*	*5.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Non-Preconditioned 3b	*0.0*	*0*	*0.0*	*24*	*5.0*	*48*	*n/a*	*n/a*	*0.01*	
Preconditioned 3b	*0.125*	*0*	*0.125*	*24*	*5.0*	*48*	*n/a*	*n/a*	*0.01*	
Non-Preconditioned 3c	*0.0*	*0*	*0.0*	*24*	*0.0*	*48*	*5.0*	*72*	*0.01*	
Preconditioned 3c	*0.125*	*0*	*0.125*	*24*	*0.125*	*48*	*5.0*	*72*	*0.01*	
Scenario 4										5
Non-Preconditioned	*0.0*	*0*	*0.0*	*24*	*15.0*	*72*	*n/a*	*n/a*	*0.01*	
Preconditioned 4a	*2.5*	*0*	*2.5*	*24*	*15.0*	*72*	*n/a*	*n/a*	*0.01*	
Preconditioned 4b	*0.125*	*0*	*0.125*	*24*	*0.125*	*48*	*5.0*	*72*	*0.01*	
Scenario 5										6a-6b
Non-Preconditioned	*0.0*	*0*	*17.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned	*1.25*	*0*	*17.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Scenario 6										7a-7b
Non-Preconditioned	*0.0*	*0*	*6.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned	*3.0*	*0*	*6.0*	*24*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Scenario 7										8a-8b
Non-Preconditioned	*0.0*	*0*	*6.0*	*15*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned	*3.0*	*0*	*6.0*	*15*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Scenario 8										9a-9d
Non-Preconditioned	*3.0*	*0*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*0.01*	
Preconditioned	*3.0*	*0*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*n/a*	*24.0*	

$\delta = .01$ hrs. This is analogous to giving a saline (non-preconditioned) dose ($\lambda_1 = 0$ mg/kg) to mice at time zero ($t_1 = 0$ hrs) and then giving a second dose ($\lambda_2 = 10$ mg/kg) of endotoxin at 24 hours ($t_2 = 24$ hrs) with both doses given as instantaneous injections ($\delta = 0.01$ hrs) at the specified times. The system is then integrated and one can look at time courses of the model variables. These parameters can be changed and the system integrated again to give another set of time course for comparison.

Scenario 1 is based on the experiments of Sly and colleagues [105], summarized in Table 2. Fig. 2a, c and d show the time courses for the model variables obtained for this first scenario and Fig. 2b is a bar graph of selected time points from the numerical data shown in Fig. 2a. Scenario 2 follows the endotoxin tolerance experiments of Wysocka et al. [120], outlined in Table 3, where tolerance is induced with a variety of preconditioning doses ranging from 0.05 to 1 mg/kg. A qualitative reproduction of their results by our model can be seen in the time courses of Fig. 3. It is interesting to note that the preconditioning dose used in scenario 2b allows for the greatest reduction in N^* compared to doses used for scenarios 2a and 2c. This indicates that for a fixed preconditioning time interval, the size of the preconditioning dose can determine the magnitude of the reduction that is detected, with a nonmonotonic relationship between the two. We address this observation in more detail in Section 2.5.

Scenario 3 is based on the experiments done by Rayhane et al. [96]. Two of these experiments are more complicated than those of Sly et al. and Wysocka et al., since several preconditioning doses, rather than only one, are given before the challenging dose. In Table 4, the designs of the three separate tolerance experiments from Rayhane et al. are outlined along with a summary of their results. Fig. 4 shows our results.

The experiment of Balkhy and Heinzel [9], scenario 4 outlined in Table 5, is slightly different from the previous scenarios we have simulated, in that the final endotoxin dose (15 mg/kg) is given 48 h after the last preconditioning dose, instead of only 24 or 26 h after. Fig. 5 shows the results of our simulations for this scenario. We note that if we had simulated giving the challenge dose of 15 mg/kg earlier than 48 h after preconditioning (e.g. at 24 or 26 h after), this regimen would have led the system to the unhealthy state. This finding highlights the importance of timing as well as dosage size to tolerance outcomes.

As mentioned previously, endotoxin, when given above a certain threshold dose, can be

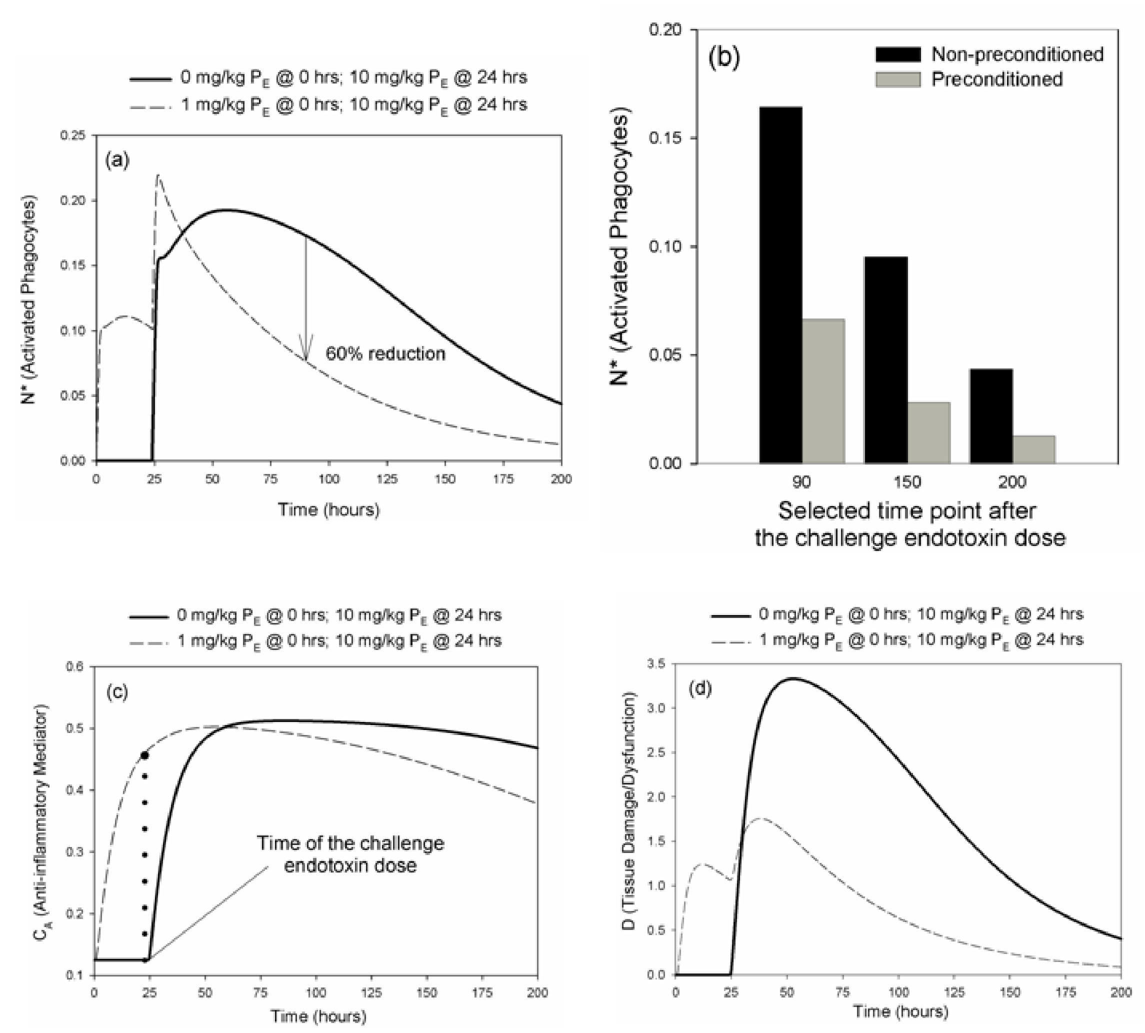


Figure 2: Numerical results of simulations following scenario 1 in Table 2, with administration parameters set as in Table 7 for Scenario 1. (a) Time courses of N^* for the non-preconditioned (solid) and preconditioned simulations (dashed), showing a maximum reduction of 60% as indicated by the downward arrow. In actual experiments, the data cannot usually be viewed as continuous time course curves. Instead, bar graphs are given showing the amount of certain analytes at a specified time after the challenge endotoxin administration, comparing the non-preconditioned group to the preconditioned group. To relate our in silico results to this convention, (b) shows a bar graph of the amount of N^* in both the non-preconditioned and preconditioned simulations at several time points after the challenge endotoxin administration, where a reduction in N^* is seen. (c)–(d) Time courses of C_A and D, respectively, for the non-preconditioned (solid curve) and preconditioned simulations (dashed curve). The dotted vertical line in (c) denotes the time the challenge P_E dose was given.

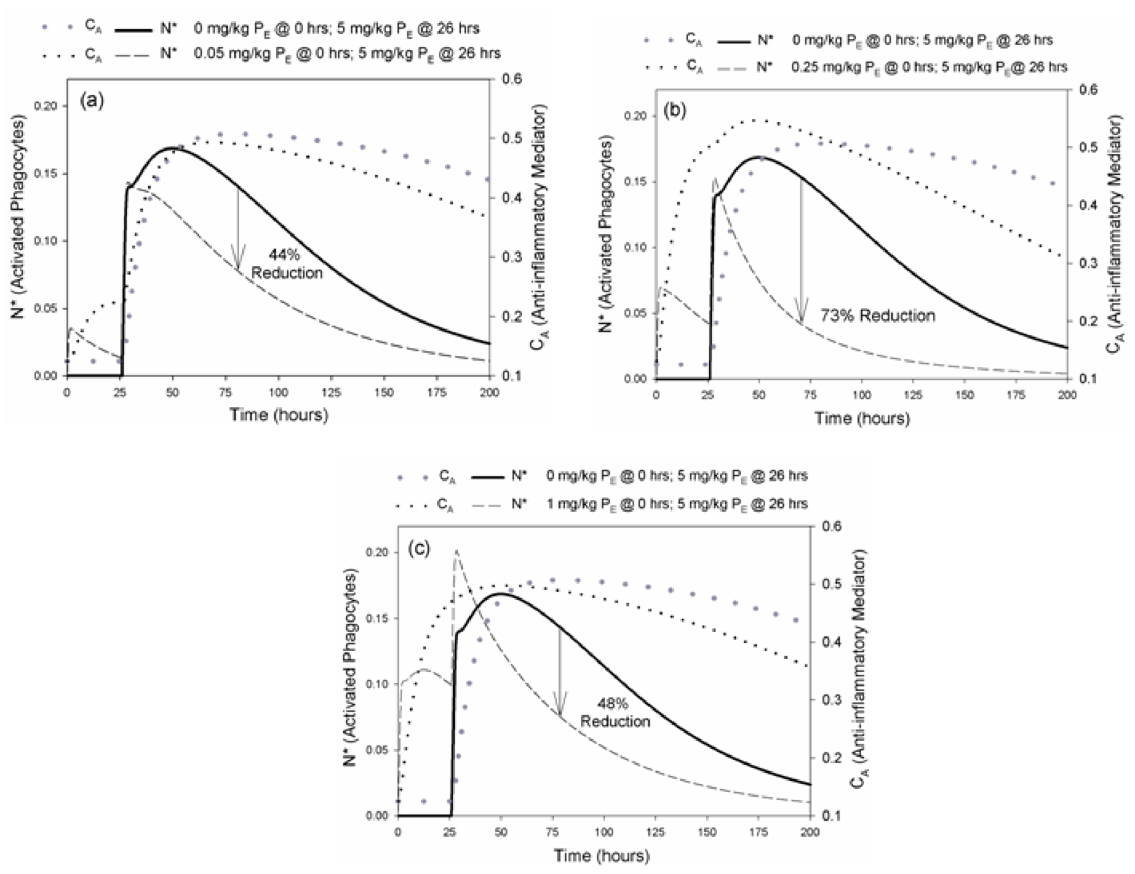


Figure 3: Numerical results of simulations following scenarios 2a–2c in Table 3, with dosage amounts converted from micrograms/mouse to milligrams/ kilogram in order to conform to the units of P_E (mg/kg) in our model. Administration parameters are set as in Table 7 for scenarios 2a–2c. Time courses of N^* for the non-preconditioned (solid) and preconditioned simulations (dashed) are shown for each scenario. Compared to the non-preconditioned simulation, there is a maximum reduction in N^* of 44% in (a) scenario 2a, 73% in (b) scenario 2b and 48% in (c) scenario 2c, indicated in each figure by the downward arrows. The time courses of C_A, for the non-preconditioned (large dots) and preconditioned (small dots) simulations, are also shown on each graph with a separate axis on the right of each graph.

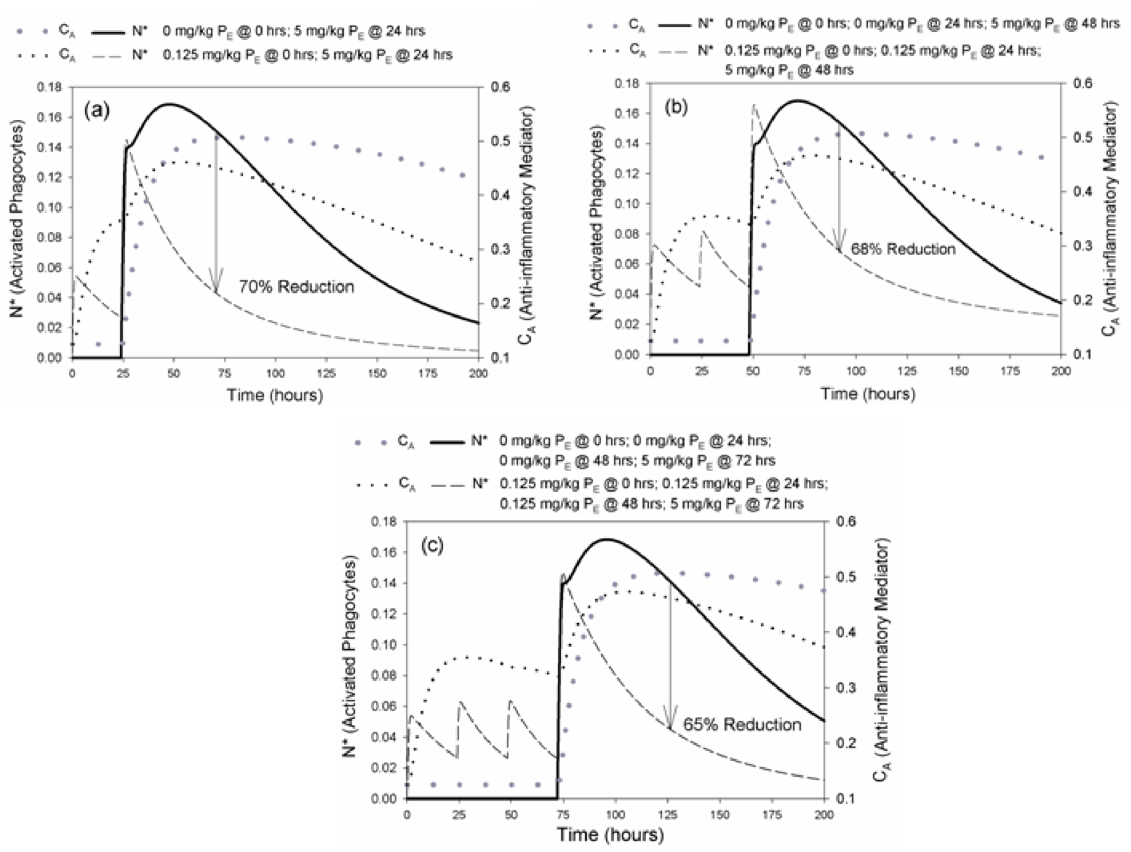


Figure 4: Numerical results of simulations following scenarios 3a–3c in Table 4, with dosage amounts converted from micrograms/mouse to milligrams/kilogram in order to conform to the units of P_E (mg/kg) in our model. Model parameters are set as in Table 7 for Scenarios 3a–3c. Time courses of N^* for the non-preconditioned (solid) and preconditioned simulations (dashed) are shown for each scenario. Measured against the non-preconditioned simulations, we see a reduction in N^* of 70% in (a) scenario 3a, 68% in (b) scenario 3b, and 65% in (c) scenario 3c, indicated in each figure by the downward arrows. The time courses of C_A, for the non-preconditioned (large dots) and preconditioned (small dots) simulations, are also shown on each graph with a separate axis on the right of each graph.

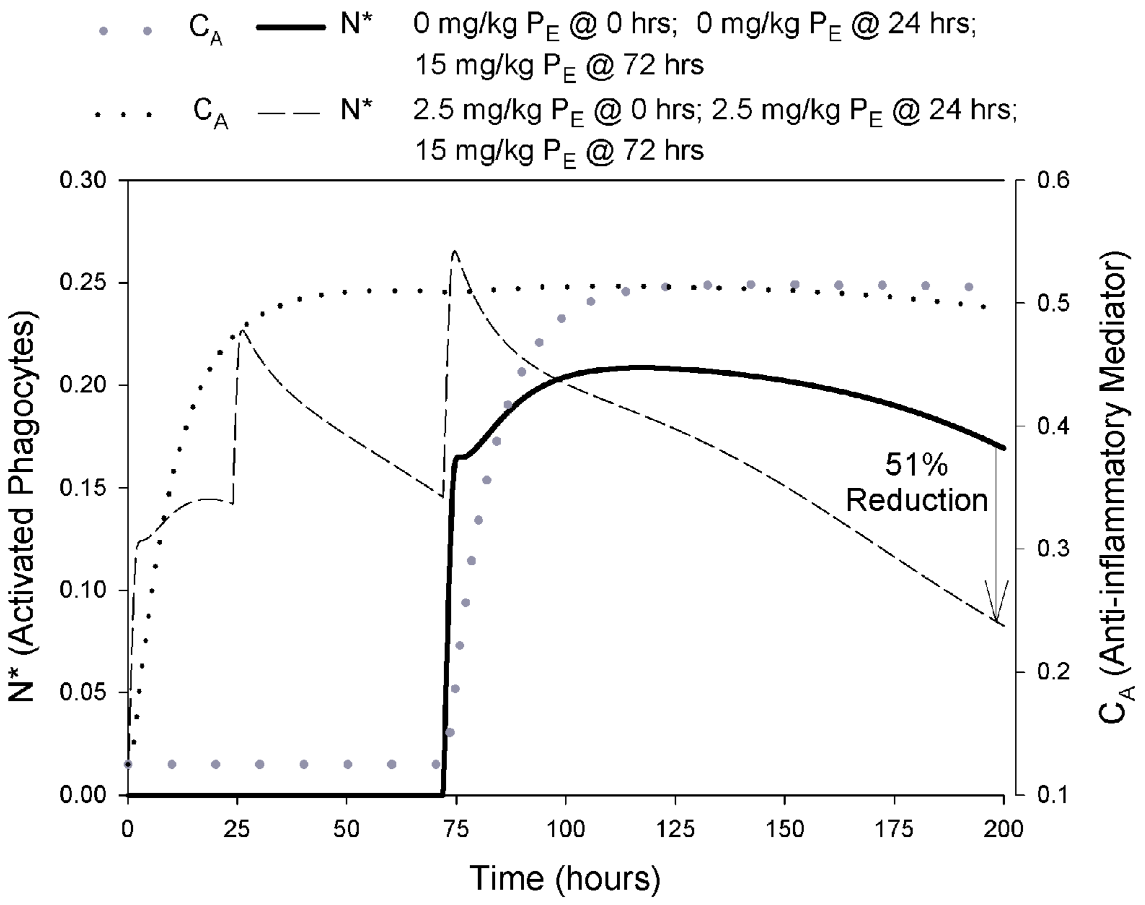


Figure 5: Numerical results of simulations following scenario 4 in Table 5, with model parameters set as in Table 7 for scenario 4. Time courses of N^* are shown for the non-preconditioned (solid) and the preconditioned simulations (dashed). Balkhy and Heinzel report a 3- to 6-fold reduction of serum TNF one hour after the challenge dose is given, compared to non-preconditioned results. Although our model does not capture an immediate reduction in our pro-inflammatory mediator, N^*, we do observe a significant reduction overall, as seen in this figure. The time courses of C_A, for the non-preconditioned (large dots) and preconditioned (small dots) simulations, are also shown with a separate axis on the right of the graph.

lethal for mice. This threshold can depend on specific experimental conditions as well as the strain of mouse used. However, experiments have shown that preconditioning mice with a low, survivable P_E dose can actually prevent animals from succumbing to a lethal challenge dose [11, 105, 122]. We conduct a model simulation of this effect using the experiment of Berg et al. [11] as a guideline (Scenario 5, Table 6). A dose of 10 mg/kg proved to be lethal in the mice that were used in Berg's experiment; however, based on our own studies, the lethal dose in our model is centered at 17 mg/kg [20]. Thus, our potentially lethal challenge dose in our simulations is λ_1 =17mg/kg of P_E. Figs. 6a and b show the simulation time courses for N^* and D, respectively, where we see that preconditioning enables a rescue from an otherwise lethal insult.

2.3.2 Potentiation scenarios sub-lethal and lethal doses

Experimentally, when the time between initial exposure to endotoxin and the secondary challenge is short relative to the magnitude of the endotoxin doses, an increase, rather than a reduction, of inflammation (i.e. TNF) is observed upon repeated endotoxin administrations. This phenomenon is referred to as potentiation [18]. As we will discuss in further detail later, both the timing of the administration of the doses as well as their magnitudes determine the final outcome of tolerance or potentiation. The scenarios introduced in this section demonstrate potentiation in several different forms. Scenarios 6 and 7 which are not explicitly based on experiments found in the literature demonstrate sub-lethal and lethal potentiation simulations, respectively. Figs. 7a and b show that in scenario 6 there is a clear elevation in the amount of N^* in the preconditioned simulation, compared to that of the non-preconditioned one, but the mediators eventually resolve to the healthy state. In scenario 7, Figs. 8a and b show that the non-preconditioned simulation results in a healthy outcome whereas the preconditioned one results in an unhealthy outcome. Comparing Scenarios 6 and 7 show that the timing and not just the amount of the second endotoxin dose determines whether or not the potentiation leads to an increase in N^* that eventually settles back to the healthy equilibrium, or to an increase that converges to the unhealthy state.

In order to experimentally simulate the kinetics of endotoxin release in animals during

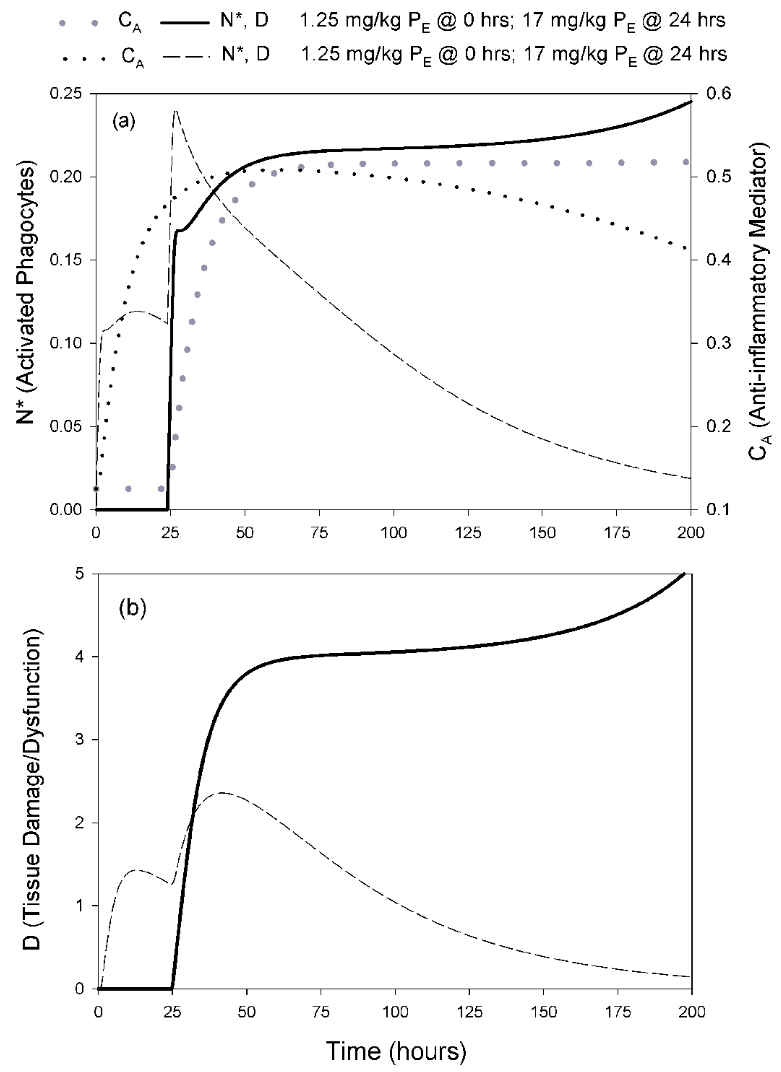
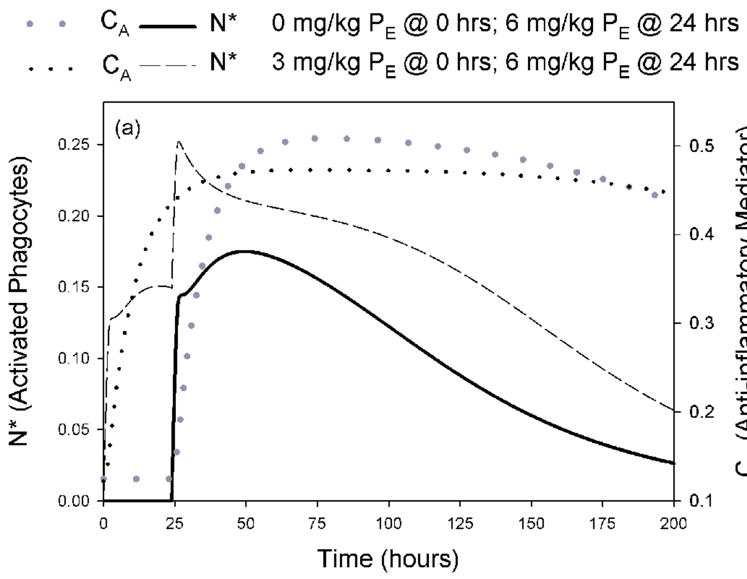


Figure 6: Numerical results of simulations based on scenario 5 in Table 6, with model parameters set as in Table 7 for scenario 5. This scenario demonstrates that our model qualitatively captures the result that a small preconditioning dose of endotoxin can prevent the negative outcome of an otherwise lethal dose. (a)–(b) Time courses of N^* and D, respectively, for the non-preconditioned (solid) and preconditioned simulations (dashed). The time courses of C_A,for the non-preconditioned (large dots) and preconditioned (small dots) simulations, are also shown on the N^* graph with a separate axis on the right of the graph. The non-preconditioned simulation clearly ends up at the unhealthy state, in which N^* and D remain high. However, the simulation that was preconditioned settles to the low healthy state, showing rescue from an otherwise lethal insult.



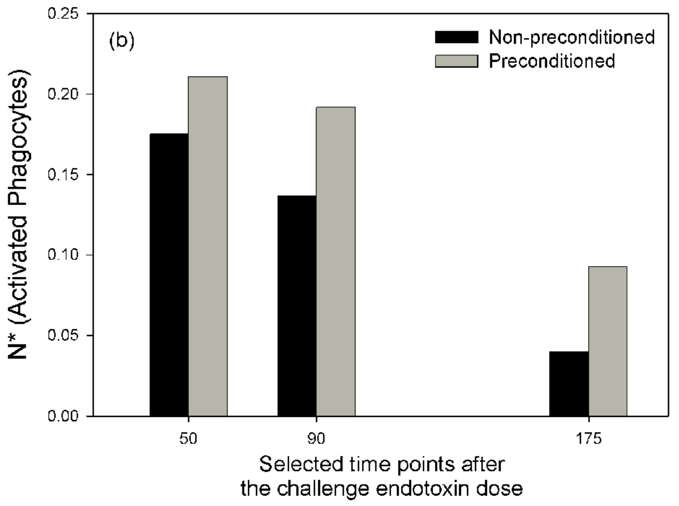


Figure 7: Numerical results of simulations for sub-lethal potentiation scenario. (a) Time courses of N^* for the non-preconditioned (solid) and preconditioned simulations (dashed), showing an increase in the amount of N^* with preconditioning compared to the non-preconditioned simulation. The time courses of C_A, for the non-preconditioned (large dots) and preconditioned (small dots) simulations, are also shown on the N^* graph with a separate axis on the right of the graph. (b) Bar graph of selected time points from (7a), showing the amount of increase in N^*.

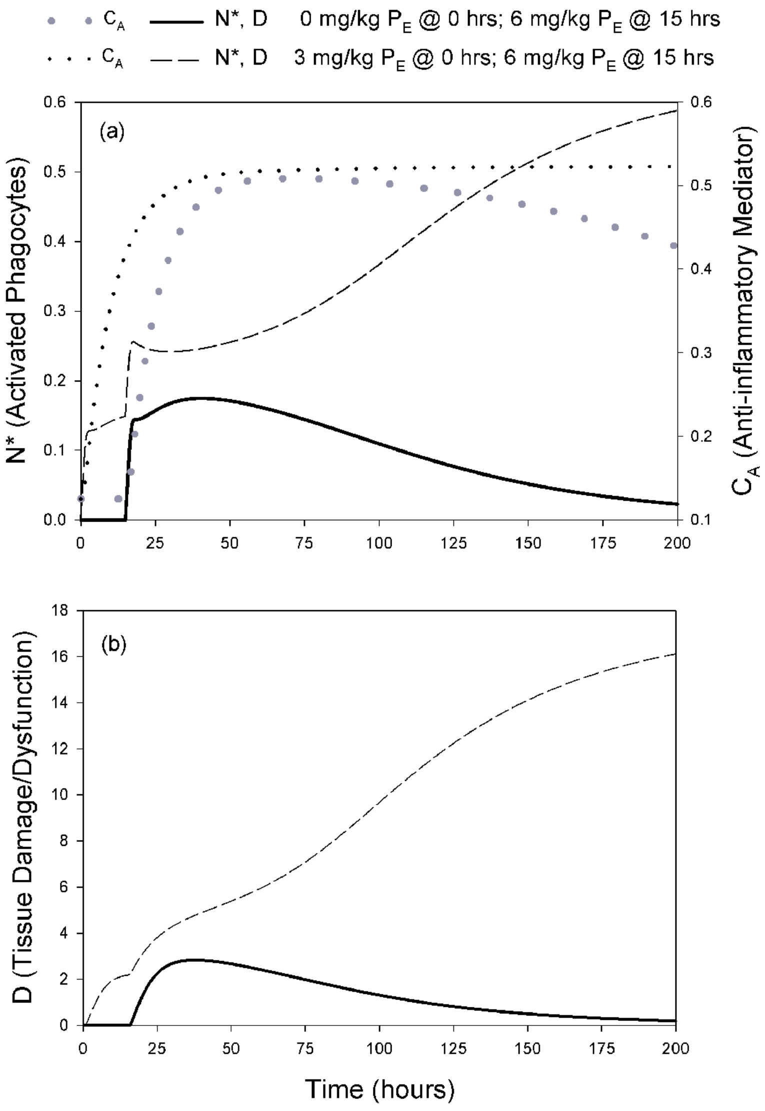


Figure 8: Numerical results of simulations for lethal potentiation scenario. (a)–(b) Time courses of N^* and D, respectively, for the non-preconditioned (solid) and preconditioned simulations (dashed). The time courses of, C_A, for the non-preconditioned (large dots) and preconditioned (small dots) simulations are also shown on the N^* graph with a separate axis on the right of the graph. Unlike the non-preconditioned simulation, the preconditioned simulation results in an unhealthy response.

sepsis, a continuous, low-dose infusion of endotoxin is administered [90]. Scenario 8 demonstrates that gradually administering a dose of 3 mg/kg P_E over 24 h forces the system to the unhealthy state, whereas the same dose given as an abrupt bolus does not. To approximate the instantaneous administration of 3 mg/kg P_E into the system, we set $t_1 = 0$, $\lambda_1 = 3$, and $\delta = 0{:}01$ (Fig. 9a). To simulate 3 mg/kg P_E given over 24 h at a constant rate, we set $\delta = 24$. In setting δ to a value of 24 we are simulating an endotoxin infusion that distributes a total of 3 mg/kg P_E gradually over 24 h (Fig. 9b). This is a fair comparison, because in both cases, in the absence of decay of P_E, the P_E level at the end of the infusion would be 3 mg/kg. Figs. 9c and d show that the constant administration of P_E, even though it is administered in very low amounts, causes the system to converge to the unhealthy state, whereas the instantaneous dose does not. These results imply that insults that elicit a strong initial pro-inflammatory response properly counter-balanced by an anti-inflammatory response are more likely to be tolerated by the host. In contrast, those stimuli that cause an initially weak but persistent response can be detrimental to the host.

2.4 THE IMPORTANCE OF THE DYNAMICS OF THE LATE PROINFLAMMATORY AND ANTI- INFLAMMATORY MEDIATORS TO TOLERANCE

A system of ordinary differential equations becomes complicated very rapidly as the number of equations increases. It can, therefore, be advantageous to attempt to reduce the number of equations to a manageable number by applying a *steady state assumption*. This strategy is most appropriately applied to variables that are transient, and is accomplished by setting their derivatives to zero; for example, if $x' = f(x, y)$, then we apply the steady state assumption to x by setting $x = X(y)$ such that $f(X(y), y) = 0$, if such an $X(y)$ exists. By making such a substitution, one is assuming that the relevant variable reaches its steady state quickly and does not deviate from it over time, although the particular value of its steady state may vary as the other quantities in the system evolve. Based on the form of the model in Section 2.2, it would be most convenient to reduce the number of equations in our model by applying the steady state assumption to D, although in fact it behaves as a slow-acting

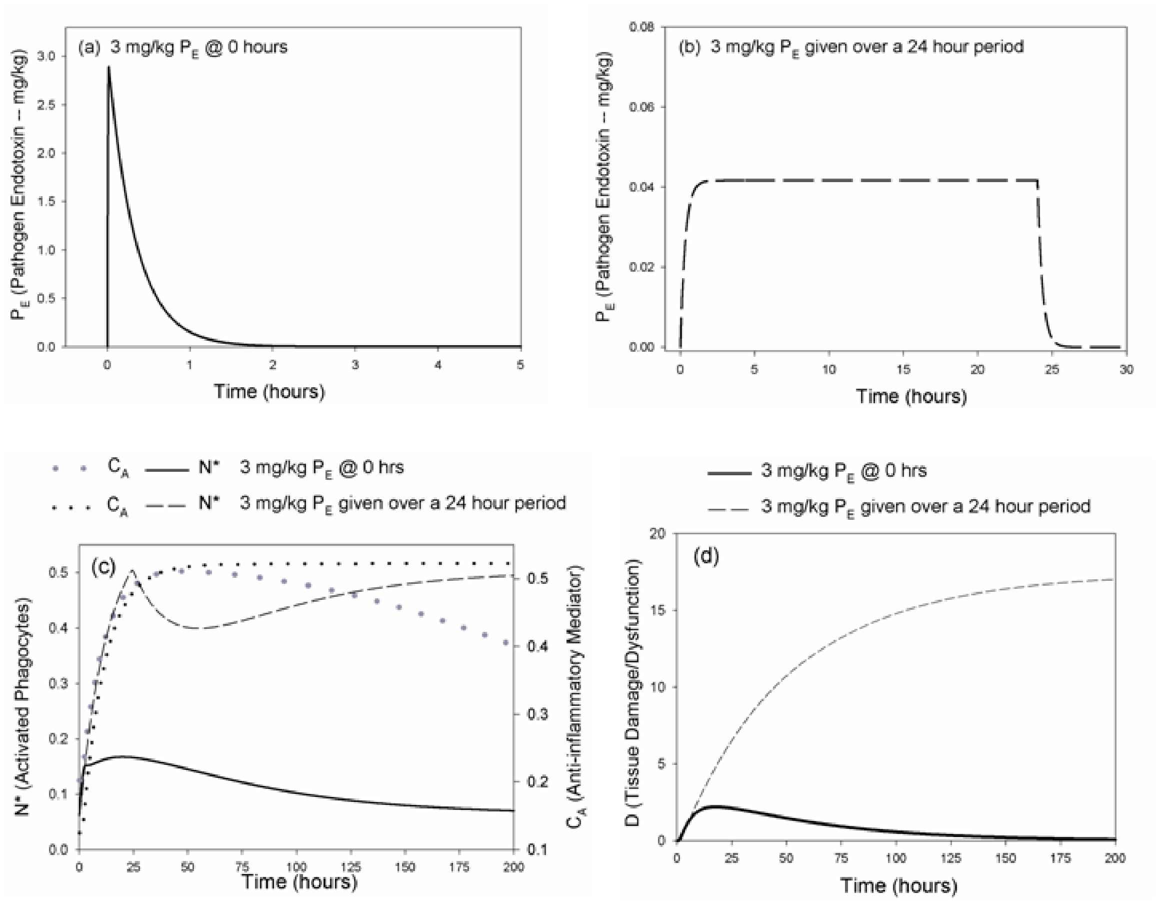


Figure 9: Instantaneous versus continuous P_E administration. (a) Time course of P_E for the simulation giving an instantaneous injection of 3 mg/kg P_E into the system. (b) Time course of P_E for the simulation giving 3 mg/kg P_E over 24 h at a constant rate. This is done by setting $\delta = 24$ in the P_E equation. In setting δ to a value of 24 we are simulating an endotoxin infusion that distributes a total of 3 mg/kg P_E over 24 h rather than an instantaneous introduction of that amount.(c)–(d) Time courses of N^* and D, respectively, for the instantaneous administration simulation (solid) and continuous administration imulation (dashed) time courses. The time courses of C_A, for the instantaneous administration (large dots) and continuous administration (small dots) simulations, are also shown on the N^* graph with a separate axis on the right of the graph.

pro-inflammatory mediator. As it turns out, under the steady state assumption on D, the model fails to reproduce the experimentally observed endotoxin tolerance results without parameter modifications that compromise the basic model performance or are outside of the physiologic range.

As mentioned, we define the existence of endotoxin tolerance in our model as a reduced N^* response to a low dose of P_E when the system is preconditioned with an initial low P_E dose. However, with D in steady state, we observe only potentiation of the N^* response regardless of when the second dose is administered. On the other hand, parameters can be changed to achieve tolerance, but these changes eliminate the possibility for the system to reach an unhealthy state, which is necessary in order for the model to retain basic biological fidelity. As previously mentioned, experiments have verified that a low preconditioning dose of endotoxin can rescue mice from a normally lethal endotoxin dose [96, 105, 122]. However, with D in steady state, a lethal P_E dose always leads to an unhealthy state even after a low preconditioning dose of P_E is given and, in fact, does so more prominently when the system is preconditioned.

Thus, dynamically modeling D allows for a number of outcomes that are not possible otherwise within the bounds of the biological constraints imposed by past experimental findings. The fact that D acts gradually and promotes the production of C_A allows the model to attain an extended C_A elevation, without compromising the existence of an unhealthy state in the system. This attribute of our model plays an important role in the reproduction of tolerance scenarios. We explored this further by looking at the effects that certain forms of altered C_A dynamics had on tolerance in our model (with dynamic D). First, appropriate model parameter values were adjusted so that C_A was only being produced by early immune responders (N^*) and so that it had an early peak and a relatively quick decay. The time course of C_A then closely resembled that of a fast acting anti-inflammatory cytokine, such as *IL-10*. In this scenario, the regimes of healthy and unhealthy still exist; however, tolerance does not occur. Indeed, preconditioning led to potentiation of the N^* response and sometimes caused the otherwise sub-lethal challenge dose to be lethal, much like what happened when D was assumed to be in steady state. Therefore, it appears that for tolerance to occur in our model, C_A cannot solely behave as an early antiinflammatory mediator, like *IL-10*.

On the other hand, another possible modification was to adjust model parameters so that C_A behaved as a later acting anti-inflammatory, accumulating on a time scale comparable to that of D. We found that significant changes in this direction drastically shrank the basin of attraction[1] of the healthy state. In some ways, modifying C_A in this way is comparable to considering *IL-10*-deficient (knockout) mice, and indeed a similar sensitivity to small endotoxin doses is seen experimentally in these animals [11, 120]. Tolerance effects have been seen in experiments with *IL-10* knockout mice. It is likely, however, that such knockout mice have a decreased susceptibility to pro-inflammatory stimuli or an increased upregulation of other anti-inflammatory mediators to compensate for the absence of *IL-10* early on in the immune response, which our model does not incorporate. Indeed, simulation of our model suggests that removal of early anti-inflammatory mediators without compensation would eliminate tolerance, since endotoxin doses small enough to be sub-lethal, given the decreased basin of attraction of the healthy state, fail to activate C_A sufficiently for tolerance to occur.

2.5 INSIGHT FROM THE MODEL'S RESPONSES TO ENDOTOXIN ADMINISTRATION

Looking at these preconditioning phenomena from the point of view of the dynamics of a mathematical model of the acute inflammatory response, we are able to offer insight into why these disparate results are seen experimentally. It is important to note that the development of this model only took into account empirical observations about the interactions of somewhat abstracted immune effectors. However, none of the endotoxin administration results that we have reproduced was built into the development of the equations. Rather, our findings emerge from the interactions of the dynamic variables and biological effects of repeated endotoxin administration. Thus, although *petitio principii* or "circular reasoning" is a potential pitfall of such reduced models, the model we present was not constructed to describe the specific paradigm of endotoxin tolerance.

The timing and magnitude of the endotoxin doses plays a crucial role in the types of

[1] The basin of attraction of a stable fixed point, x^*, of a dynamical system is the set of all initial conditions that dynamically approach x^*under the flow of the vector field as $t \to \infty$. [108, 118]

outcomes that are observed. In the model considered, the variable N^* is inhibited by C_A, the levels of which can remain elevated even after enough time has passed for N^* to start returning to its resting value. Using scenario 1 as an example, Fig. 2c demonstrates how the amount of C_A varies between the nonpreconditioned and preconditioned simulations at the time that the second PE dose is given (dotted vertical line). Comparing the amount of the anti-inflammatory mediator in the two simulations at this time point, we see that with preconditioning there are significantly higher levels of C_A than without preconditioning, which shows C_A levels that are still at baseline. In scenarios 1–4, which lead to endotoxin tolerance, the challenge endotoxin dose that follows the preconditioning regimen is given during a time when the system is precisely in this state of relatively low N^* and elevated C_A. The build-up of the anti-inflammatory mediator, induced by preconditioning, results in a reduction of the overall inflammation or build-up of N^*, incited by the challenge endotoxin dose. Fig. 2c also shows that even though a short time after the challenge dose the levels of the anti-inflammatory mediator for the non-preconditioned simulation have risen above the preconditioned simulation levels, this occurs too long after the final endotoxin stimulus to influence the relative levels of N^* across the two experiments.

It is important to note that, despite the inhibitory effects of C_A, the full model exhibits an attracting, unhealthy steady state that can be attained, for example, following the introduction of a single, sufficiently large dose of endotoxin. For the rescue phenomenon demonstrated in scenario 5 (Figs. 6a and b), we see that a preconditioning dose of endotoxin can prevent the system from reaching the unhealthy state upon subsequent exposure to an otherwise lethal endotoxin dose. Such a rescue is possible because the preconditioning changes the state in which the system lies when the lethal dose is encountered. Specifically, the anti-inflammatory mediator rises enough and the pro-inflammatory mediators are close enough to equilibrium after the preconditioning dose so that when the previously lethal endotoxin stimulus is given, the system lies in the basin of attraction of the healthy, baseline state, rather than that of the unhealthy state.

This behavior is similar to the tolerance observed in scenarios 1–4. The results from scenario 1 (Figs. 2a and c) are utilized in Figs. 10b and c to illustrate this by showing a projection of the system onto the N^*-C_A phase plane where trajectories for N^* and C_A can be seen with respect to their nullclines ($dN^*/dt = 0$ and $dC_A/dt = 0$).[2] For comparison, related time courses of N^* are shown in Fig. 10a. In addition to the elevation of C_A above equilibrium when the challenge endotoxin dose is administered, the proximity of N^* and D to their equilibrium levels is equally important for both tolerance and rescue to occur, since the (N^*, D) subsystem forms a positive feedback loop. As long as the level of N^* or D remains too high, the introduction of the lethal endotoxin dose will place the system in the basin of attraction of the unhealthy state. The potentiation phenomenon illustrated in scenario 6 (Fig. 7a) follows similarly, stemming from a second endotoxin dose that comes soon after the initial one. Figs. 11a–c use the same strategy demonstrated with Figs. 10a–c this time using the results of scenario 6 to illustrate potentiation from the viewpoint of a projection of the system to the N^*-C_A phase plane.

Likewise, Fig. 12a shows the N^*-D-C_A phase space to illustrate the rescue demonstrated in scenario 5 (Figs. 6a and b). The elevated amount of C_A in the system at the time of the challenge dose blunts the effect of the potentially lethal dose, enabling the trajectory of the preconditioned simulation toremain in the basin of attraction of the healthy fixed point. In addition, Fig. 12b shows a similar rescue scenario in the N^*-D-C_A phase space along with the 2-dimensional separatrix consisting of the stable manifold of the saddle point of the system.[3] The code for calculating the separatrix manifold was written in MATLAB, based on an algorithm presented by Krauskopf and Osinga [63]. The algorithm and its implementation are given in detail in Chapter 3. The separatrix forms the border between the basins of attraction of the healthy and unhealthy states. This surface is exact (and thus invariant; see Strogatz [108] for more details) only in the limit of $P_E = 0$. Nonetheless, since P_E decays quickly, this surface gives a reasonable estimate to the true separatrix location in (N^*-D-C_A) space for times that are not too close to endotoxin dose administration times. Thus, the location of a trajectory a short time after a challenge dose, relative to the separatrix, determines the long-term fate of the system.

[2]For a planar system $dx/dt = f(x, y), dy/dt = g(x, y)$, the nullclines are the two curves $f = 0$ and $g = 0$. Fixed points of the system occur precisely at intersections of the nullclines. In higher dimensions, nullclines are actually *nullsurfaces* and are much harder to visualize. In Figs. 10 and 11, we project onto a 2-dimenstional phase plane and are, therefore, looking at slices of the C_A and N^* nullsurfaces.

[3]We chose not to use exactly the same trajectories produced in Fig. 12a because they followed the manifold too closely and it was difficult to visualize what exactly was happening.

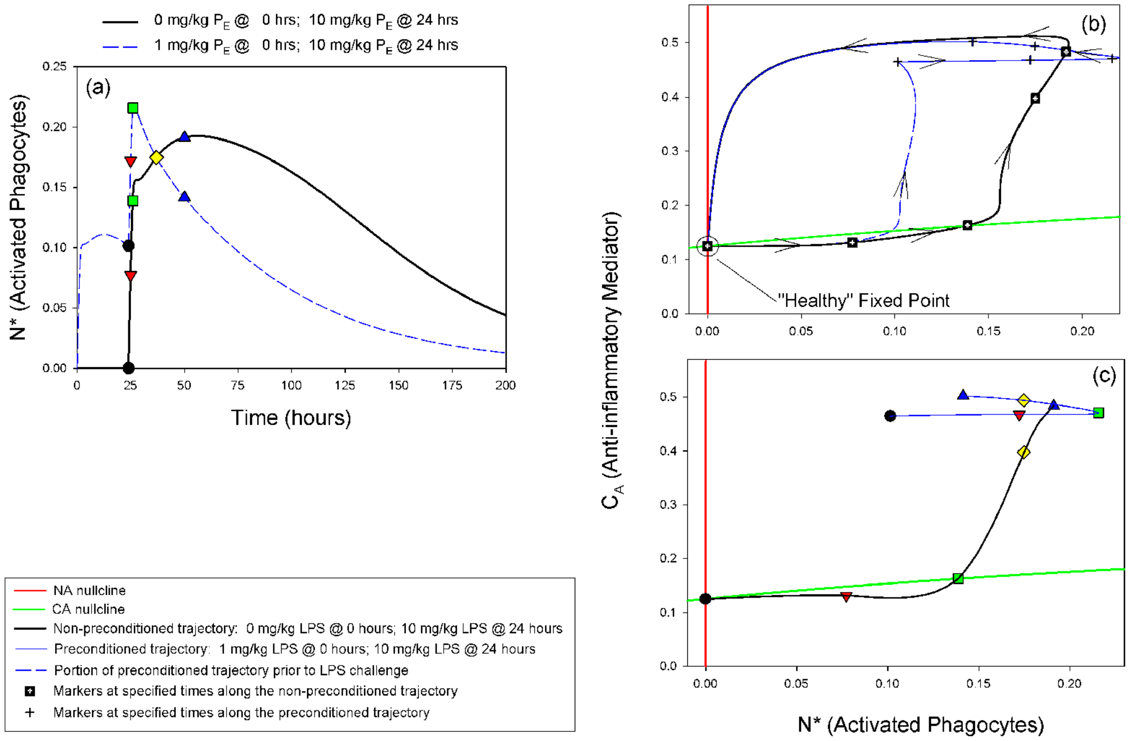


Figure 10: Endotoxin tolerance (based on scenario 1) illustrated with the $N^* - C_A$ phase plane. The specific markers represent the following: black circle = time point just prior to the administration of the challenge endotoxin dose, red upside down triangle = 1 h after the challenge dose, green square = 2 h after the challenge dose, yellow diamond = 13 h after challenge dose, and blue triangle = 26 h after challenge dose. In (a) the symbols described above are positioned on the N^* time courses of the non-preconditioned (solid) and preconditioned (dashed) simulations, where the preconditioned simulation time course falls below the non-preconditioned simulation time course just after the yellow diamond marker. In (b), the N^* nullcline (red; vertical line) and C_A nullcline (green; almost horizontal line) are shown along with two curves representing the trajectories of the nonpreconditioned (black; thick) and preconditioned (blue; thin) simulations of scenario 1. The arrows signify which direction the trajectories are flowing in the phase plane. Although both trajectories end at the healthy fixed point after running their courses, the preconditioned (blue; thin) trajectory actually approaches the fixed point faster, resulting in tolerance. Several points are marked on the non-preconditioned and preconditioned trajectory with ■ or +, respectively, denoting specific times prior to and after the time of the challenge endotoxin dose. These time points are shown again in (c) where they are color coded and connected to stress which ones belong on the non-preconditioned and preconditioned curves shown in (b). The black circle belonging to the curve of the non-preconditioned simulation shows that the trajectory is sitting at the healthy fixed point, where N^* and C_A are at their background levels. In comparison, the black circle belonging to the curve of the preconditioned simulation is sitting at a place in the phase plane where C_A is much greater than its baseline value. It is also a place where N^* is above its baseline, however, the trajectory is beginning to turn to the left toward the healthy fixed point and the challenge dose does not push the trajectory too far in the N^* direction. Comparing the other symbols in (c) on the two curves, the preconditioned trajectory is at a lower N^* level than the non-preconditioned curve at the same time just after the yellow diamond (compare the positions of the blue triangles with respect to N^*). This indicates that tolerance has occurred. Thus, the state the system is in at the time the challenge dose is given determines the outcome of the simulation.

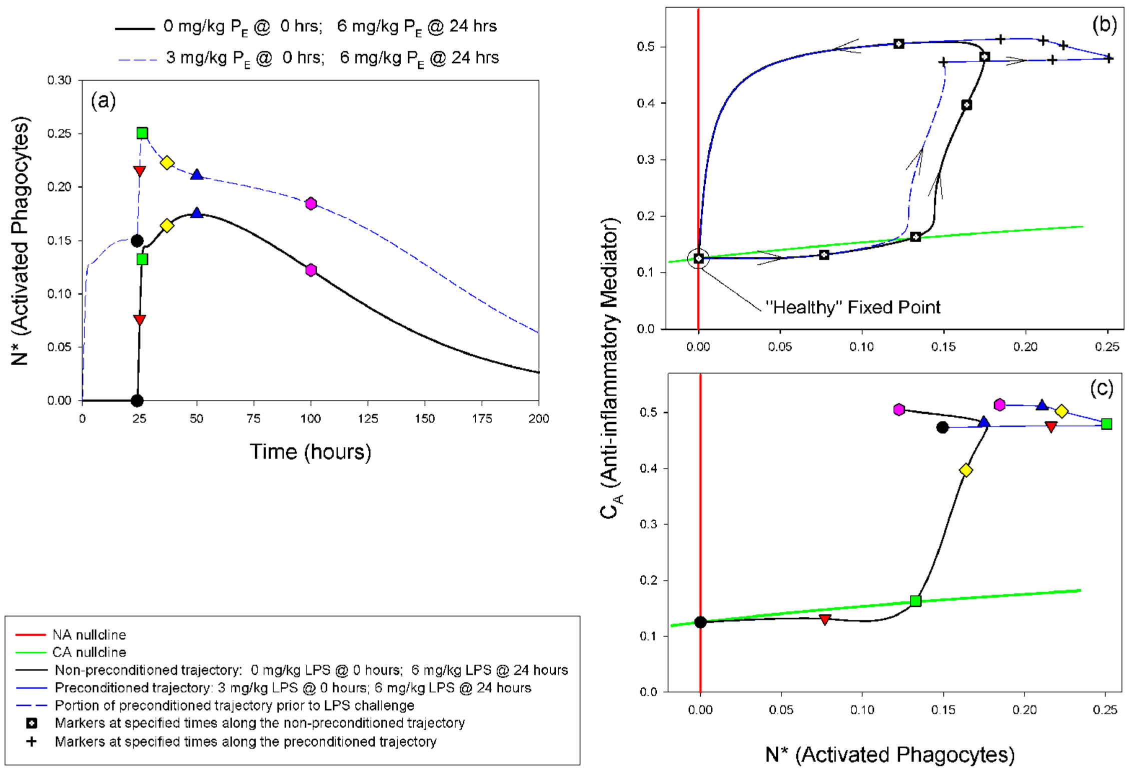
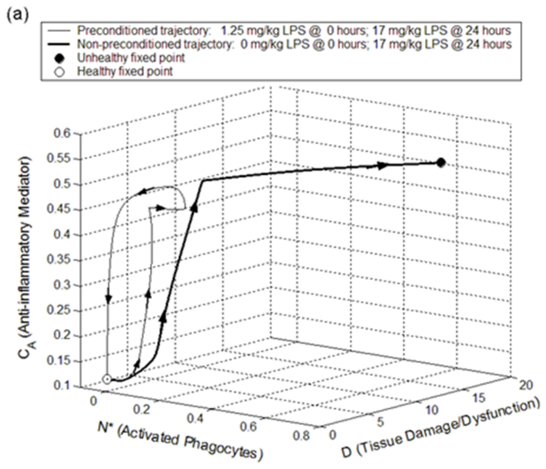


Figure 11: Potentiation (based on scenario 6) illustrated with the $N^* - CA$ phase plane. The N^* time courses in (a) of the non-preconditioned (solid) and preconditioned (dashed) simulations illustrate that at each of the time markers, the N^* level of the preconditioned simulation is significantly above the non-preconditioned simulation levels. The explanation given in the caption for Fig. 10 is very similar to this panel, except that instead of the preconditioned trajectory outrunning the preconditioned simulation trajectory, it now trails the non-preconditioned trajectory for all time, as seen in (b). In addition, when comparing the symbols in panel (c), the levels of N^* (x-axis) are always greater in the preconditioned simulation. Although the position of the preconditioned simulation trajectory just prior to challenge is a place of elevated CA, the level of N^* is quite high as well, and when the challenge is given, the new starting point of the preconditioned trajectory is pushed further to the right into high N^* territory. Consequently, the preconditioned trajectory cannot draw level with, much less pass, the non-preconditioned simulation.



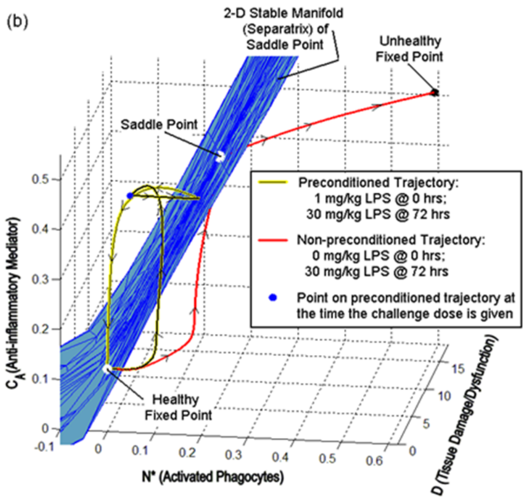


Figure 12: Rescue scenario in the $N^* - D - CA$ phase space. (a) Using our simulations for scenario 5, this figure illustrates the concept of protection or rescue in the $N^* - D - CA$ phase space. The non-preconditioned trajectory (bold) is pushed into the basin of attraction for the unhealthy fixed point by the injection of the 17 mg/kg P_E dose at 24 h. The preconditioned trajectory, however, remains in the basin of attraction of the healthy fixed point. This is because the amount of C_A in the system just prior to the challenge dose is significantly above baseline and the effect of the 17 mg/kg hit of P_E is, therefore, blunted. (b) Using a slightly different but similar simulation to scenario 5, in which preconditioning again leads to rescue, we now show a portion of the 2-dimensional separatrix consisting of the stable manifold of the saddle point of the system (see text for more details). The preconditioned trajectory (black on yellow) stays on the healthy side of the surface after the challenge dose, while the challenge dose pushes the non-preconditioned trajectory (red) to the unhealthy side of the surface.

In all of the above discussions, it is clear that timing is very important to achieve tolerance. Therefore, we further investigate the dependence of toleranceon the amount of time between the preconditioning and challenge dose as well as the magnitude of preconditioning by examining a range of preconditioning doses and times at which they are given. In scenario 1, it was shown that a preconditioning dose of 1 mg/kg endotoxin 24 h prior to the challenge endotoxin dose of 10 mg/kg endotoxin produced endotoxin tolerance, marked by a decrease in the level of the model variable, N^*, compared with the non-preconditioned simulation at a particular point in time, namely 66 h after challenge.

However, if we vary the amount of the preconditioning dose as well as the amount of time between the preconditioning dose and challenge dose, we see that there is a wide range of preconditioning doses and times at which they can be administered that also show a decrease in N^* at the time of comparison with the non-preconditioned simulation. Furthermore, the relationship between the size of the preconditioning dose and the time that it is given relative to the challenge dose is not obvious. In Fig. 13, we see that potentiation is evident for the range of preconditioning times that are close to the time the challenge dose is administered (approximately 0–15 h before challenge). Tolerance is observed when this interval is typically longer than 15 h. Interestingly, there is a brief interval (15–20 h before challenge) during which smaller preconditioning doses typically allow for more tolerance than larger doses do. For this case, the key point is that larger preconditioning doses elicit more inflammation than do smaller doses. Thus, for large doses, the inflammation is still high when the challenge dose is given, such that less tolerance is observed than for small doses. There is, however, a range of these preconditioning times (20–110 h before challenge) during which the relationship between the magnitude of the preconditioning dose and the amount of tolerance it elicits is not monotonic, due to a competition between the amount of inflammation and the amount of anti-inflam-mation invoked by the preconditioning dose. Finally, for the range of preconditioning times from 110 to 200 h before challenge, the larger preconditioning doses exhibit more tolerance than the smaller doses. This is due to the fact that the smaller anti-inflammatory responses elicited by small preconditioning doses have worn off by these times, whereas for the larger preconditioning doses, the large degree of inflammation invoked has subsided by these times while the associated strong, prolonged anti-inflammation persists.

Thus, early second stimulation with endotoxin leads to potentiation of inflammation and consequently enhances lethality. Alternatively, if the second stimulus comes too late, significant tolerance fails to be induced. In our simulations, the build-up of C_A after a sub-lethal preconditioning endotoxin dose is transient, and C_A eventually settles back to equilibrium, along with the other effectors in the model. Thus, the preconditioned system response to late challenge stimuli is similar to that seen in nonpreconditioned responses. In summary, we expect the existence of a window of possible challenge dose times and preconditioning magnitudes for which endotoxin tolerance is possible. In Chapter 4, we explore the nature of tolerance in a more general setting from a purely mathematical viewpoint.

2.6 DISCUSSION

The preconditioning phenomena of potentiation and tolerance characterize acute inflammation in both rodents and humans [23, 122]; in humans, the latter phenomenon is often referred to as "immune paralysis" or "immune exhaustion", in which leukocytes–derived from patients with severe inflammation as measured by circulating pro-inflammatory cytokines—often produce low levels of these same inflammatory agents [93, 94]. In this paper, we show that an experimentally calibrated but highly reduced computational model for the acute inflammatory response [98] (also see Supplementary Materials in section 2.7) incorporates sufficient dynamic complexity to qualitatively reproduce a suite of experimental results associated with multiple endotoxin administrations in mice. Our success in matching experimental endotoxin tolerance results offers support for the biological relevance of the reduced model. Moreover, our simulations illustrate how the outcomes of endotoxin administration experiments can emerge as a natural consequence of the interactions of different components of the acute inflammatory response and also highlight the importance of including a dynamic late proinflammatory component in the model. We find that the relative time scales of the onset and decay of pro- and antiinflammatory mediators are key determinants of outcomes in these experiments, as illustrated in the simulations and phase plane projections that we present. Finally, the results of our simulations involving potentiated responses (Fig. 7), low-

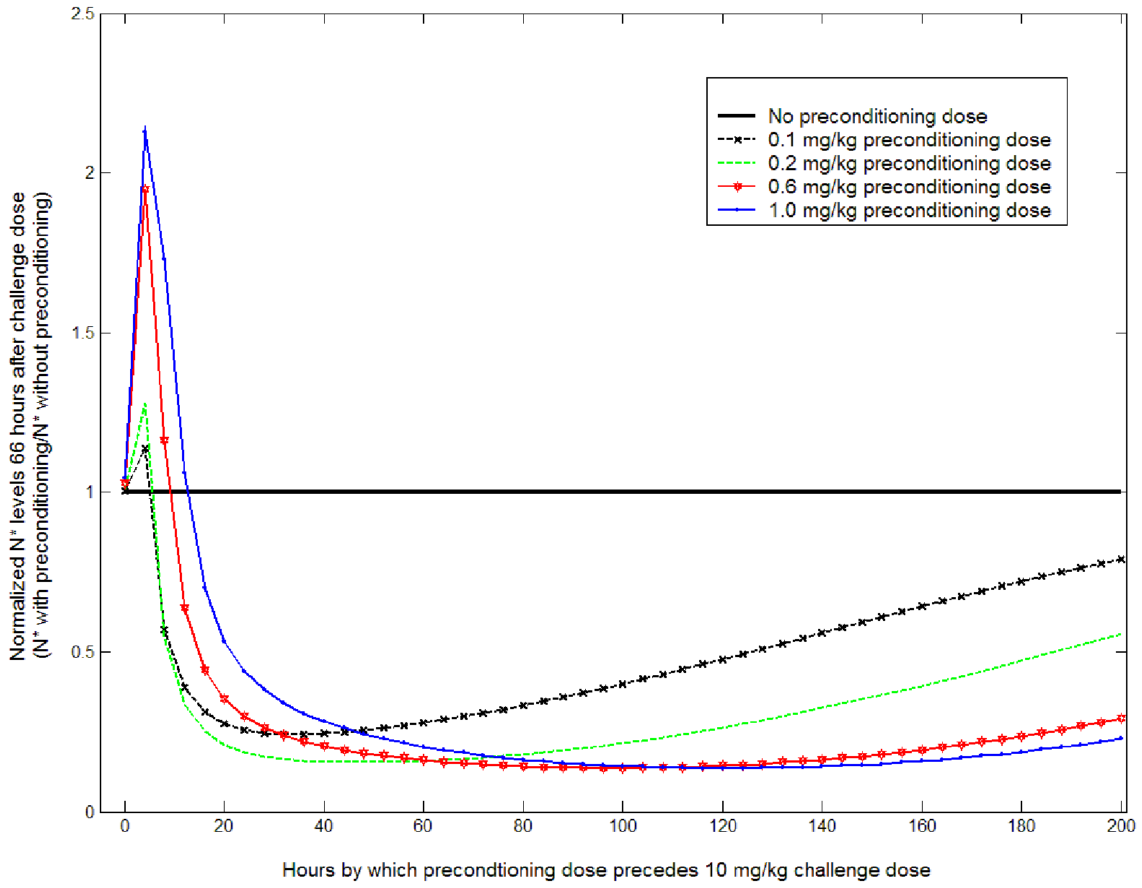


Figure 13: Dependence of tolerance on preconditioning dose timing and magnitude. The solid, horizontal line marks the normalized level of N^* of the non-preconditioned simulation at 66 h after a 10 mg/kg challenge dose is given. Each individual curve shows the normalized level of N^* (recorded at 66 h after challenge) for a particular preconditioning dosage amount as a function of the time at which the dose was given prior to the challenge dose (from 0 to 200 h prior to challenge with 10 mg/kg endotoxin). Points on the curves that fall below the black line indicate that tolerance has occurred: The N^* value for a preconditioned simulation at 66 h after challenge is lower than that of the non-preconditioned simulation (represented by the solid, horizontal line). Those which are above the solid line produce potentiation instead. (See Section 2.5for further details.)

dose protracted endotoxin infusion (Fig. 9), and variations in the timing and amplitude of preconditioning doses (Fig. 13) yield predictions that remain to be verified experimentally.

Mathematical approaches to understanding endotoxin tolerance have not been extensively represented in the literature. However, in the work of Mayer and colleagues [75] a simple two equation mathematical model of the immune response is presented and tolerancelike behavior is mentioned. Their model consists of immune cells (E) and target cells (T) which represent bacteria, for instance, and are inhibited by the immune cells. Although Mayer et al. do not consider endotoxin specifically and do not model anti-inflammatory effects, a form of tolerance is manifested in their model by a reduction in the growth of their target cells when a secondary infection is initiated, compared to the growth of the initial infection. This reduction is due to the fact that the concentration of the immune cells they model is elevated when the secondary infection is introduced. In fact, the secondary infection is initiated after the system has approached a steady state in which the immune cell concentration is high and the primary infection has been cleared. This simulation differs substantially from the situation we consider, in which a positive resolution corresponds to a return to the baseline rest state and endotoxin challenges come during transient excursions from this state induced by endotoxin preconditioning.

Much of the experimental literature regarding endotoxin tolerance focuses on the roles of *IL-10* and T*ransforming Growth Factor-$\beta 1$* (*TGF-$\beta 1$*), two potent anti-inflammatory cytokines [42, 68, 95, 124]. The possible roles that they each may have in endotoxin tolerance have been documented by Randow et al. [95] and others [18, 42, 105]. Our model suggests that the timing of doses for which tolerance will occur strongly depends on the time course of the antiinflammatory mediators. For the experiments that we have reproduced, it is necessary for an anti-inflammatory influence to arise early on in the response, as is seen with *IL-10*, but also to remain elevated longer than *IL-10*. This latter feature might be true of mediators like *TGF-$\beta 1$* or possibly *IL-6*, which has been shown to have antiinflammatory characteristics and is typically a cytokine produced relatively late in the course of an immune response [121].

It has been shown that preconditioning with *IL-10* protects mice from lethal endotoxin doses and also partially mimics endotoxin tolerance [3, 18, 48]. However, this finding does not contradict our findings on the importance of a prolonged anti-inflammatory response for tolerance, since *IL-10* preconditioning leads to a different time course of anti-inflammatory mediators than occurs with the intrinsic immune response to endotoxin preconditioning. Indeed, if *IL-10* preconditioning introduces a significant resence of *IL-10* in the host during the time that a rather toxic dose of endotoxin is administered, then it will suppress the pro-inflammatory response and thus enhance tolerance. Moreover, *IL-10* can either induce or activate *TGF-*β*1* [68, 111], thereby prolonging the overall anti-inflammatory effect.

As with other recently developed mathematical models of the acute inflammatory response [20, 21, 65] the model used here was calibrated to be consistent with relevant experimental literature. However, we do not claim that the model with the parameters we have chosen will be valid over a wide range of species, especially in regard to the differences in sensitivity to endotoxin. Mice can survive much higher doses of endotoxin than humans can, for instance. Not all the parameter ranges and estimates could be acquired from mouse data alone; however, whenever possible, we looked at literature and data regarding experimental work done in mice. In order to reproduce experiments carried out with other species such as humans, for example, it would be necessary to consult species specific data. See Supplementary Materials for more details regarding parameter choices.

Since our model is based on a simplified response system, it has certain limitations. For example, it is difficult to match specific biological mediators to the variables we have chosen, with the exception of endotoxin (P_E), and the model cannot predict quantitative measurements. However, we have tried to select parameters such that the time courses of our variable, N^*, are qualitatively similar to those suggested by experimental data to exist for early pro-inflammatory mediators like *TNF* and activated phagocytes. For example, our preliminary experimental data in rats suggest that the peak of activated neutrophils roughly matches that of circulating *TNF* [66]. There are other apparent differences between our results and those in the literature. For instance, the reductions we show in N^* are not seen at the peak of its production, whereas the literature suggests that the reduction in *TNF* production is seen at its peak (90–120 min after challenge) [101]. However, since our early pro-inflammatory mediator is not solely based on *TNF*, exact comparison with experimental *TNF* data is simply not feasible. In addition, we demonstrate the induction of endotoxin tolerance by modeling the basic binding interaction of P_E with immune effector cells but without any special alterations that affect the clearance of P_E. Despite these limitations, our results highlight specific ways in which endotoxin tolerance and related phenomena can emerge from the timing and the overall interplay between mediators of the acute inflammatory response and illustrate the utility of a reduced model for the computational testing of hypotheses and generation of predictions related to this response.

2.7 SUPPLEMENTARY MATERIALS

The standard parameter values for the reduced endotoxin model equations 2.1-2.4 are supplied in Table 8. These parameter values are selected to remain within the given ranges and constraints, which are based on experimental literature as well as on unpublished data. Details on the derivation of these ranges are given below. Parameters that could not be documented from existing data were estimated such that the subsystems presented in Reynolds et al. [98] behave in a biologically appropriate manner for all physiologically relevant levels of the anti-inflammatory mediator and so that the modified endotoxin model presented here also exhibits observed biological behaviors of immune mediators in the presence of endotoxin. Many of the comments below refer to the subsystems of the pathogen model thoroughly discussed in [98]. Although the comments may not be explicitly relevant to the endotoxin model discussed here, we include them since they explain how many of parameters which do appear in the endotoxin model were estimated.

Units for N^*, C_A, and D are not given explicitly because there is no single biological entity or marker that these variables represent and thus there are no specific units that can quantify these variables empirically. Hence, we use "N^*-units," "C_A-units," or "D-units" because we cannot be any more precise about them. Although C_A (Anti-inflammatory Mediator) has characteristics of *IL-10* and *TGF-β1*, it would be inappropriate to assign real units to this variable and quantitatively compare it to actual data from these or other anti-inflammatory mediators. Correspondingly, units of most parameters related to these variables are not in conventional form, but rather in terms of the associated variable.

Comments:

1. The range for the decay rate of pathogen endotoxin, μ_{pe}, was calculated from various half-lives of endotoxin given by Iversen et al., [51] on pg. 1160 and the control curve on Fig. 1 (for rabbits), Warner et al., [117] in the abstract (for rats), and Yoshida et al., [123] in the abstract (for rabbits).

2. The second term of the pathogen endotoxin equation 2.1 allows for multiple endotoxin doses to be given at different times during a simulation. The function S is the difference of two Heaviside step functions, which determines when an administration starts (t_{on}) and stops (t_{off}). The value assigned to t_i determines t_{on} and $t_i + \delta$ determines t_{off} for the i^{th} endotoxin dose. The parameter λ_i, is the amount of the i^{th} endotoxin dose to be given and δ governs the duration of time over which the dose is administered. The coefficient λ_i/δ thus gives the dose of endotoxin to be administered per hour during the time interval $[t_i\ ,\ t_i + \delta]$. The summation in Eq. 2.1 allows for numerous doses to be given during the simulation. Section 2.2 describes how the values for λ_i, δ, and t_i were chosen when emulating various experimental scenarios involving repeated endotoxin administration.

3. We estimated the rate of activation of N^* by P_E, k_{npe}, in such a way so that the following two stable states exist: healthy (levels of all mediators are low) and unhealthy (levels of all mediators, excluding P_E, are high). A lethal dose of endotoxin for mice was acquired from the literature and we used this information to calibrate the model to give proper responses to lethal and sublethal doses known for mice, specifically [24]. Thus, both of these states can be reached by simply varying the initial condition of P_E: for a low (single) initial condition of P_E (less then 17 mg/kg) the system resolves to the healthy state and for a high (single) initial condition of P_E (17 mg/kg or more) the system evolves to the unhealthy state, consistent with observed biological behaviors of immune mediators in the presence of endotoxin.

4. In the model presented by Reynolds, et al. [98], the activated phagocytes/pathogen (N^*/P) subsystem was fit such that for low pathogen growth rate (k_{pg}) health is the only stable state, and at a moderately high k_{pg} septic death exists and is stable. Parameters in this subsystem were first estimated so that these general dynamics occurred for a significant range of the physiologically possible levels of the anti-inflammatory mediator. They were then adjusted so that both the reduced model with pathogen dynamics and

the modified endotoxin model presented here exhibited observed biological behaviors of immune mediators in the presence of pathogen or endotoxin, respectively. See also comment 6.

5. In Reynolds et al. [98], the activated phagocytes/tissue damage (N^*/D) subsystem was initially fit such that for physiologically relevant levels of the anti-inflammatory mediator the system is bistable between health and aseptic death with a reasonable basin of attraction for the health state. Adjustments were then made so that both the reduced model with pathogen dynamics presented in Reynolds et al. and the altered endotoxin model presented here exhibited observed biological behaviors of immune mediators in the presence of pathogen or endotoxin, respectively. See also comment 6.
6. Once the anti-inflammatory mediator (C_A) was incorporated in the pathogen model of Reynolds et al. [98] as a dynamic variable, the parameters were adjusted so that the reduced pathogen model now has the following behavior (1) the model exhibits bistability between the health and aseptic death states for low k_{pg} with a plausible basin of attraction for the health state, (2) for moderate to high k_{pg} all three states (health, aseptic death, and septic death) are stable, and (3) as k_{pg} continues to increase, the health state and the aseptic death state lose stability.
7. The parameter, k_{nn}, corresponding to the rate of activation of resting phagocytes by those previously activated, was estimated to ensure $\mu_n > (s_{nr}k_{nn})/\mu_{nr}$ This inequality must hold for the health state to be stable in the pathogen model of Reynolds et al. [98].
8. In Reynolds et al. [98], snr, the source of resting phagocytes (N_R), was set to ensure a stable concentration of resting phagocytes in the health state. It was adjusted to balance μ_{nr}, the decay rate of resting phagocytes . These parameters are related since in the health state $N_R = s_{nr}/\mu_{nr}$.
9. The range for the decay rate of the resting phagocytes, μ_{nr}, was calculated from the half-lives (6-20 hours) of circulating neutrophils presented in Coxon et al. [25].
10. The half-life of activated phagocytes, μ_n, is longer than the half-life of resting phagocytes, μ_{nr}, due to delayed apoptosis in the activated population; therefore, $\mu_n < \mu_{nr}$ [25].
11. In Reynolds et al. [98], the peak of the activated phagocyte response elicited from pathogen, k_{np}, is greater than that triggered by damage, k_{nd}; therefore, $k_{nd} < k_{np}$. In

the modified endotoxin model presented here, we adopted the same restriction so that $k_{nd} < k_{npe}$.

12. The minimum for μ_d, which represents tissue repair, resolution, and regeneration, was estimated from data in Wang et al., [116]. We used the half-life of *HMG-1*, since it is a histone tethering protein leaked by damaged cells as a surrogate for the many danger molecules that perpetuate the inflammatory signal. Wang and colleagues give data for *HMG-1* levels during an inflammatory response to bacterial lipopolysaccharide (LPS). Therefore, we estimated the lower limit as the slope of the data shown in Fig. 1C of Wang et al. [116] during the decay phase of *HMG-1*. It would be unrealistic to set μ_d to a value higher than the time constant of a recognized marker of cellular injury.

13. The value for c_∞ was set such that corresponds to $\approx 75\%$ inhibition, i.e. , when C_A reaches maximum value in response to an insult. We set this to be approximately 75% because Fig. 6B in Isler et al. [50] shows that when the anti-inflammatory mediator, *IL-10*, is blocked with *anti-IL10* there is approximately a 75% increase in the production of *IL-12* (a pro-inflammatory cytokine produced by activated phagocytes).

14. Organisms have constitutive levels of anti-inflammatory effectors. Therefore, the source parameter, s_c, was chosen to balance the corresponding half-life, μ_c. These parameters are related because at the health state $C_A = s_c/\mu_c$.

15. Anti-inflammatory signals have downstream cellular effects not explicitly modeled herein, lasting longer than the effector cytokines or molecules producing it. Therefore, the value for μ_c was set at the lower limit of reported half-lives of anti-inflammatory effectors, which were estimated from pg. 130 of Bacon et al. [8], Table 1 on pg. 277 of Bocci [12], pg. 291 of Fuchs et al. [40], and the abstract of Huhn et al. [49].

16. This Hill coefficient for Eq. 2.3 was set to six so that the response of tissue damage to activated phagocytes is not hypersensitive. A lower Hill coefficient would not appropriately represent this. In other words, it is biologically plausible that low levels of activated phagocytes do not trigger significant amounts of damage that could lead to a positive feedback capable of sustaining aseptic death. Also, for values six and higher, there was not a significant difference in the sensitivity of damage to the activated phagocytes. Contrary to the common inference regarding the use of Hill coefficients in enzymatic kinetics,

we are not implying that a cooperativity-based mechanism is at work.

Table 8: Explanation of model parameters used in simulations

Name	Range	Value Used	Description	Comments
μ_{pe}	0.6207–14.85	3/h	Decay rate of pathogen endotoxin (P_E)	1, 4, 5
λ_i	n/a	Various (mg/kg)	Amount of the i^{th} P_E dose administration	2
δ	n/a	0.01 or 24 hr	Duration of PE injection: 0.01 corresponds to instantaneous delivery (1/100 of an hour) and 24 corresponds to constant delivery of a dose over 24 hours.	2
t_i	n/a	Various (h)	Time at which the ith PE dose is given	2
k_{npe}	Estimated	9/(mg/kg)/h	Activation of phagocytes by pathogen endotoxin (PE)	3, 4, 5
k_{nn}	Estimated	0.01/N^*-units/h	Activation of phagocytes by already activated phagocytes (or the cytokines that they produce)	4, 5, 7
s_{nr}	Estimated	0.08/N_R-units/h	Source of resting phagocytes	4, 5, 8
μ_{nr}	0.069-0.12	0.12/h	Decay rate of resting phagocytes (macrophages and neutrophils)	4, 5, 9
μ_n	Less than μ_{nr}	0.05/h	Decay rate of activated phagocytes (macrophages and neutrophils)	4, 5, 10
k_{nd}	Less than k_{npe}	0.02/D-units/h	Activation of phagocytes by tissue damage (D)	5, 11
k_{dn}	Estimated	0.35/D-units/h	Max rate of damage production by activated phagocytes (and/or associated cytokines/free radicals)	5
x_{dn}	Estimated	0.06 N^*-units	Determines level of activated phagocytes (N*) needed to bring damage production up to half its maximum level	5
μ_d	0.0174 (min)	0.02/h	Decay rate of damage; combination of repair, resolution, and regeneration of tissue HMGB-1 release by damage	5, 12
c_∞	Estimated	0.28/C_A-units/h	Threshold for effectiveness of the anti-inflammatory response	6, 13
s_c	Estimated	0.0125C_A-units/h	Source of anti-inflammatory (CA) (IL-10, TGF-β1, cortisol);	6,14
k_{cn}	Estimated	0.04 C_A-units/h	Maximum production rate of Anti-inflammatories	6
k_{cnd}	Estimated	48 N^*-units/D-units	Controls relative effectiveness of activated phagocytes versus damage in producing anti-inflammatories	6
μ_c	0.15-2.19	0.1/h	Decay rate of the anti-inflammatory mediator	6, 15
hill	Estimated	6	Hill coefficient in the Damage Equation	16

3.0 AN IMPLEMENTATION OF AN ALGORITHM FOR GENERATING 2-D (UN)STABLE MANIFOLDS FOR 3-D ODE SYSTEMS

3.1 INTRODUCTION

Stable and unstable manifolds of fixed points are important invariant geometric structures in the theory of dynamical systems. Intersections of stable and unstable manifolds can be the source of interesting, complex behavior. They can be valuable for visualizing the separatrix of a saddle point between two stable fixed points, showing how the phase space is divided into regions containing points that have common convergence sets as $time \to \infty$ or $time \to -\infty$. [43] Recall that in Chapter 2 we used the stable manifold to visualize how a challenge dose of endotoxin can cause a trajectory that started in the basin of attraction of the healthy fixed point to be bumped to the other side of the manifold and heading toward the unhealthy fixed point.

With the 4D ODE model describing the acute inflammatory response to endotoxin, there were two stable states in the chosen parameter regime: one that represented a healthy outcome and the other an unhealthy outcome, separated by an unstable saddle point. The separatrix forms the border between the basins of attraction of the healthy and unhealthy states. We wanted to visualize the threshold between these life and death outcomes; however, in systems of n-dimensions, this threshold is in the form of an (n-1)-dimensional (stable) manifold. Since the endotoxin tolerance system is four dimensions, it would be difficult to visualize the stable manifold of the saddle even if it were a two-dimensional structure since it would be embedded in a four dimensional space. However, because the endotoxin variable, P_E, decays quickly we are able to consider only the other three variables: N^*, D, and C_A. (i.e. set the endotoxin equation to zero). Then, it is possible to visualize the resulting two-dimensional stable manifold of the saddle point that exists in a three dimensional space, giving a reasonable idea of how the space is divided.

There are presently a number of algorithms that have been published in the literature that use various approaches to generate two dimensional (un)stable manifolds of equilibrium points. A thorough review of these methods, given by Krauskopf and colleagues in [64], not only explains each method but also points out the difficulties that each faces. Generally, all the methods approach the problem by 'growing' the manifold out from a small neighborhood around the fixed point. Specifically, though, they differ in the way this is accomplished and also in the way that the accuracy of the mesh is ensured. Furthermore, these methods can be used to compute manifolds of dimension higher than two. For obvious visualization reasons, however, demonstrations of the algorithms are usually done for two-dimensional manifolds in three-dimensional space, as is done in this chapter.

Simply parameterizing the manifold by a collection of circles generated by the image under the flow of the vector field of an initial small circle around the fixed point, while intuitive, does not provide a nice enough mesh. This can be caused by one eigendirection being stronger than the other. As a result, the subsequent curves produced by flowing the vector field out from the initial circle soon become deformed, creating an insufficient and incomplete mesh. Instead, the manifold needs to be parameterized by more nicely formed curves which are topologically equivalent to circles. These can be found if one considers the geodesic distance between two points, x and y, on the manifold. As Krauskopf and colleagues explain, the geodesic distance is "the arclength of the shortest path" in the manifold that connects x and y. Parameterizing the manifold by curves called *geodesics*, which contain points that are all at an equal geodesic distance from the fixed point, is the correct method. This parameterization only depends on the geometry of the manifold and not the underlying dynamics of the system. Furthermore, the smoothness of the manifold guarantees that the geodesic level sets are topologically equivalent to circles up to some maximum geodesic length from the fixed point. See [64] and [106] for more details.

The usual example case for testing the various algorithms is the computation of the stable manifold of the origin of the Lorenz system:

$$\left.\begin{array}{rcl} \dot{x} & = & \sigma(y-x), \\ \dot{y} & = & rx - y - xz, \\ \dot{z} & = & xy - bz \end{array}\right\} \tag{3.1}$$

with the following standard parameter values: $\sigma = 10, r = 28$, and $b = 2\frac{2}{3}$. Abraham and Shaw presented the first handwritten drawing of this manifold [1], while Guckenheimer and Worfolk first published a computer generated image [45]. Since then, methods due to Doedel, Dellnitz and Hohmann, Guckenheimer and Vladimirsky, Henderson, Johnston and colleagues, and Krauskopf and Osinga have graced the dynamical systems literature. [30, 31, 32, 33, 44, 46, 58, 63] However, what is lacking is the availability of a program that allows users to generate manifolds for their own systems, as we wished to do for the endotoxin tolerance model presented in Chapter 2. In the present chapter, the implementation of an algorithm to generate a 2D (un)stable manifold of a saddle equilibria for 3D ODE systems is discussed. The algorithm we implement is due to Bernd Krauskopf and Hinke Osinga in their 1999 article published in *Chaos [63]*. We first introduce the algorithm using the notation used by Krauskopf and Osinga and then present the details of our implementation of the algorithm, which was programmed using MatLab. [72]

3.2 THE ALGORITHM

In their 1999 article, Bernd Krauskopf and Hinke Osinga presented pseudo code for an algorithm to generate a 2D mesh of an unstable manifold. The paper, as well as the review paper [64] mentioned above, thoroughly explains how the algorithm works and demonstrates its capabilities with impressive pictures of the manifold for the Lorenz system; however, specific details were not given as to how some parts of the algorithm were computationally and practically carried out in order to generate these figures. Moreover, there was no code or software made available by which to utilize the method. Thus, we sought to convert the pseudo code given in their paper into a usable MatLab program. In Chapter 2, since the stable manifold was calculated for the endotoxin tolerance model, the implementation of this algorithm is presented in the context of the generation of a mesh for a 2D stable manifold of a saddle equilibrium with alternatives for a 2D unstable manifold in parenthesis. The explanations given in Krauskopf and Osinga's work is done in the context of the unstable manifold; however the differences are minor, mainly having to do with integration being carried out in forward or backward time.

First, we detail the input and output data.

- INPUT
  - f : the right-hand-side of the vector field: $\dot{x} = f(x)$, $x \in \mathbb{R}^3$, f sufficiently smooth
  - x_0 : the saddle point of f
  - C_r, δ : initial discrete circle with radius δ, approximating the local (un)stable manifold, $W^s_{loc}(x_0)$ ($W^u_{loc}(x_0)$), in the (un)stable eigenspace. The (un)stable manifold theorem guarantees that $W^s_{loc}(x_0)$ ($W^u_{loc}(x_0)$) exists in a small neighborhood around x_0. The (un)stable manifold is tangent to the (un)stable eigenspace of the linearization of $\dot{x} = f(x)$ at x_0. Thus, if δ is chosen sufficiently small, then C_r will be a good approximation to $W^s_{loc}(x_0)$ ($W^u_{loc}(x_0)$). This comprises the starting data of the mesh and is the foundation from which the next mesh points (i.e. next discrete circle) will be calculated.
  - Δ : initial guess of the euclidean distance between one discrete circle and the next concentric circle in the mesh. This value will potentially change depending on the accuracy of the mesh. In other words, Δ might need to be decreased so that the generated mesh is approximating the manifold accurately enough. Accuracy specifications are discussed later.
  - L_{arc} : total arclength to be computed. In the end, because the arclength of the manifold is approximated by summing all the various values of Δ used throughout the computation, the manifold is calculated up to an approximation of this specified arclength.
- OUTPUT
  - The overall output that the algorithm generates is a mesh on $W^s(x_0)$ ($W^u(x_0)$) with an approximated arclength from x_0 close to L_{arc}. Each discrete circle or band

that comprises the mesh is stored in a matrix structure so that the manifold can be generated from previously generated data (a much faster process!) or so that the manifold can be continued from a specific band.

Starting with the saddle point, the global (un)stable manifold $W^s(x_0)$ ($W^u(x_0)$) mesh is 'grown' out from the saddle in bands made up of successive discrete circles generated by the algorithm, as demonstrated in figures that follow. In reality, the first circle is the saddle point – a circle with radius zero. For a 2D (un)stable manifold of a saddle, there are two linearly independent eigenvectors associated with the negative (positive) eigenvalues of the linearization about the saddle from which the next substantial discrete circle, C_r, with center, x_0, is calculated. The points of this circle are then connected to the saddle point to form the first band of the manifold as shown in this first picture.

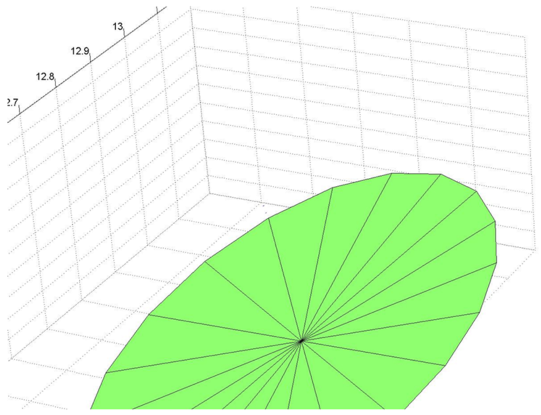


Figure 14: Initial discrete circle around the Saddle Point

Each subsequent circle is found by using the previous discrete circle. Denote C_r as the previously calculated discrete circle and let $r \in C_r$. The following steps are carried out in order to find each point on the next discrete circle lying on $W^s(x_0)$ ($W^u(x_0)$), all of which are then connected to the points from the previous discrete circle, forming the next band of the manifold:

- Calculate the half plane, $\mathcal{F}_r$, perpendicular to the previous circle at point, r. The goal is to find, for each r, the point, b_r, on $\mathcal{F}_r$ that is Δ away from r. This was accomplished via a shooting method described in more detail in the next section.

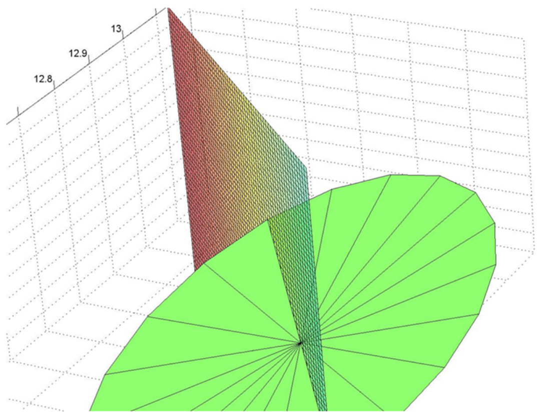


Figure 15: The half plane, $\mathcal{F}_r$, perpendicular to the previous circle at point, r

- Shooting from the point r as the first initial condition, find the intersection of the resulting trajectory with $\mathcal{F}_r$. For an (un)stable manifold, we integrate in backward (forward) time, in order to generate the trajectories. Then, we choose the next initial condition on C_r from which to shoot. This choice is based on the results of the previous shooting attempt. When an intersection point is Δ away from r, then this point is the desired b_r that we

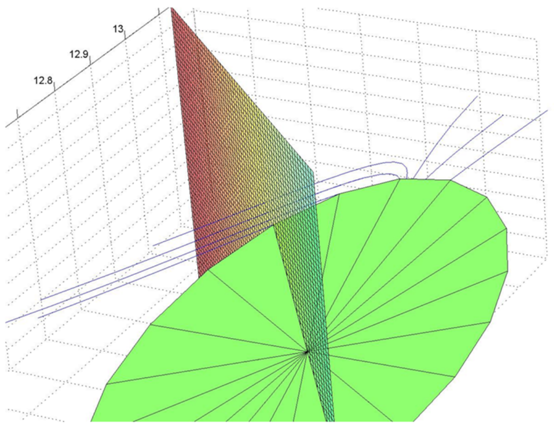


Figure 16: Example trajectories illustrating the shooting method for finding the next point on the plane $\mathcal{F}_r$ in the mesh of the stable manifold

wanted to find. Then, we move to the next point in C_r from which we repeat the process to find the next b_r associated with the next $r \in C_r$. This is done for each $r \in C_r$ until a new discrete circle is formed This process and its implementation are given in more detail in the next section.

- The previous process is done for each $r \in C_r$ until a new discrete circle, having x_0 as its center and being situated at a distance Δ from C_r, is formed. This new discrete circle, C_b, is connected to C_r, forming another band of the manifold.

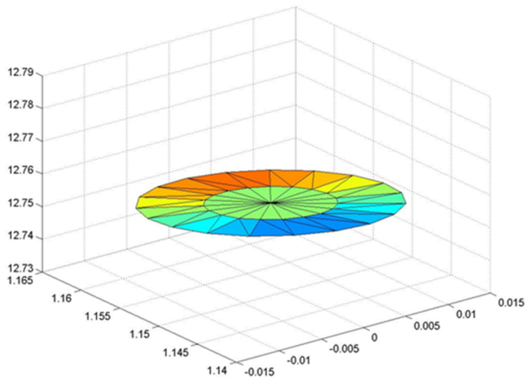


Figure 17: The first two bands of the stable manifold, connected via a triangular mesh

- The new discrete circle then becomes the next C_r from which another discrete circle is calculated. This is continued for the desired arclength. As a demonstration of the capability of the program, we generate the stable manifold of the origin for the Lorenz system 3.1.

The algorithm provides a method by which to calculate a mesh that is not affected by magnitude differences in the negative (positive) eigenvalues and the accuracy of which can be guaranteed. After each new mesh point is calculated, it is checked using an accuracy subroutine to make sure it is accurately following the (un)stable manifold. The accuracy check used by Krauskopf and Osinga is based on a method used in another of their papers and is comparable to a similar technique due to Dana Hobson. [47] The accuracy is determined by controlling the angle formed by three successive points in the mesh along with "the product of this angle and the distance between the last two of the three points."[63, 47]

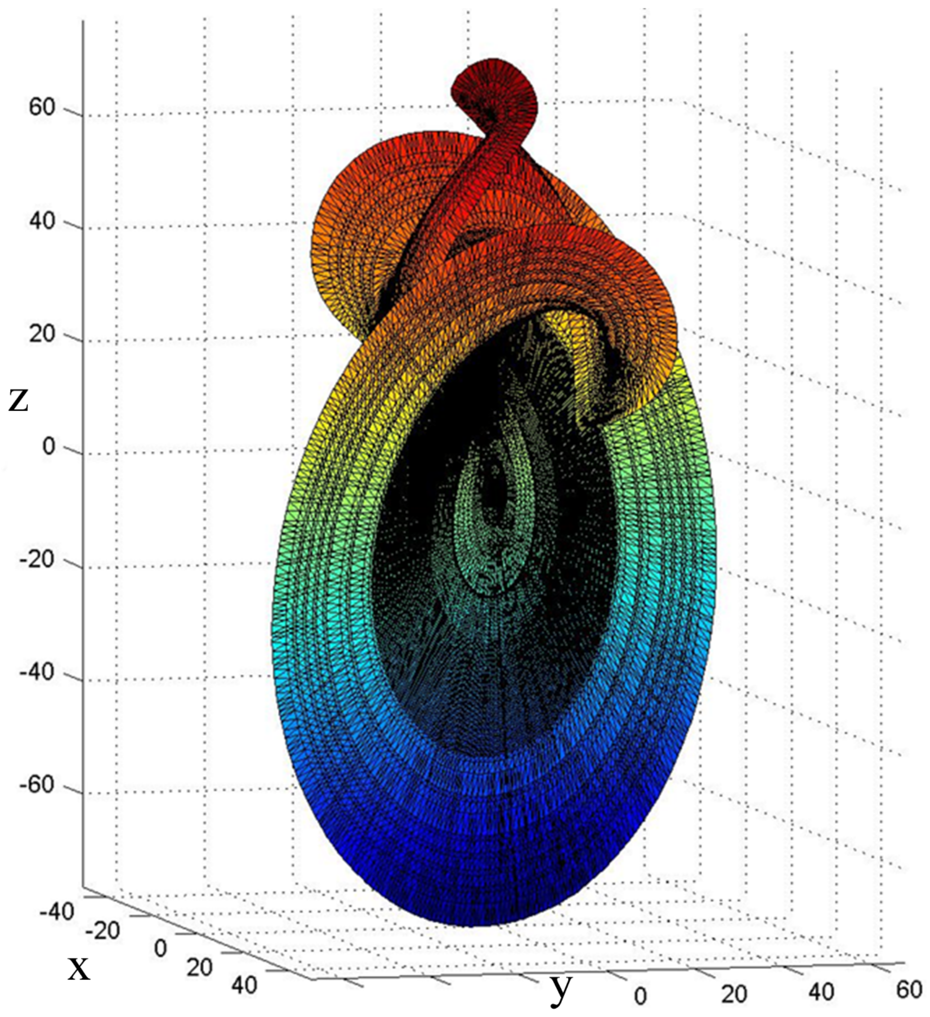


Figure 18: A view of the stable manifold of the origin in the Lorenz system calculated with our Matlab implementation of the algorithm given by Krauskopf and Osinga [63].

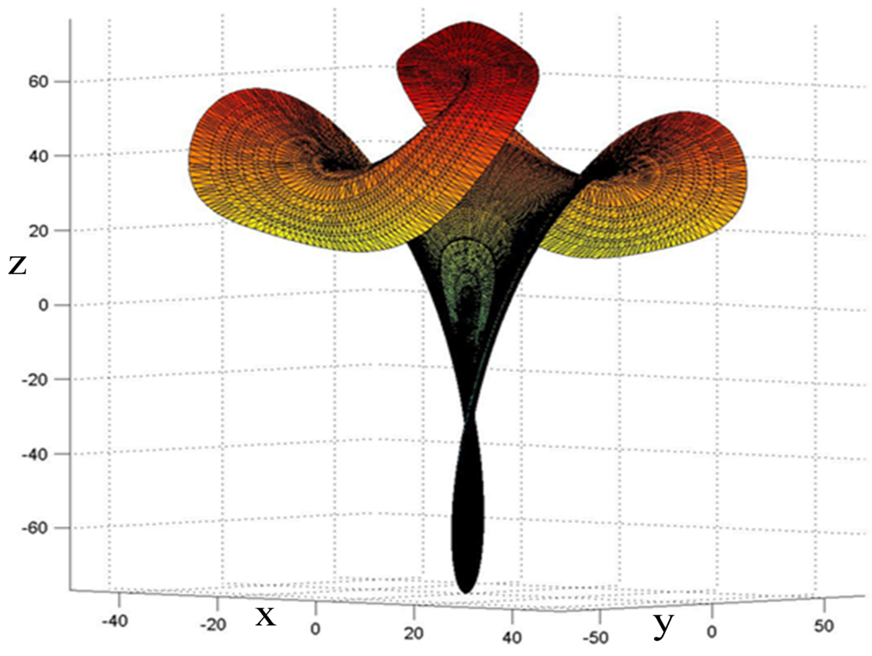


Figure 19: A view of the stable manifold of the origin in the Lorenz system calculated with our Matlab implementation of the algorithm given by Krauskopf and Osinga [63].

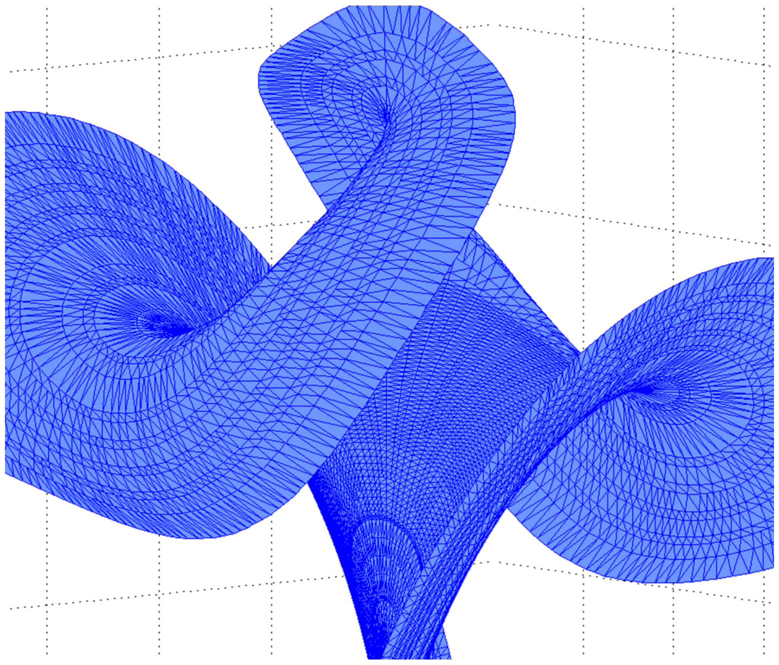


Figure 20: A closeup of the mesh of the Lorenz manifold that was calculated with the present implementation of the algorithm due to [63]. Additions and deletions of mesh points can be seen.

This helps determine the value of Δ, which might need to be decreased to maintain desired accuracy. On the other hand, Δ may be allowed to increase based on the accuracy estimates.

3.3 IMPLEMENTATION USING MATLAB

In this section, the MatLab m-file of our implementation of the algorithm in the previous section is given along with ample comments distributed throughout explaining the various steps. The algorithm grows the manifold, starting from the saddle point, by generating concentric discrete circles that form bands of a specified geodesic distance from the saddle point. Each circle of points must be computed before the next circle is generated. In addition, there are accuracy checks to ensure the mesh correctly represents the manifold. All lines of code are in a `different font` and numbered to the left. Comments about the code are in **bold** and lettered (a., b., c., etc) under the lines of the code to which they refer.

1. `function Manifold(basicsfile,rhsfilename, CircleNumber, SystemName)`

2. `odefunction = str2func(rhsfilename);`

3. `save('odefunctionname.mat','odefunction');`

   a. `basicsfile` is the name of the m-file that contains information about the 3D ODE systems, including eigenvectors and eigenvalues. The code for this file and for the `rhsfilename` file described next, is given after the entire manifold code is presented.

   b. `rhsfilename` is the name of the m-file containing the ODE system. This file is used for carrying out the numerical integration.

   c. `CircleNumber` refers to the number of the circle from which the algorithm should start. If this is the first time the algorithm is run, then this number would be zero. However, if x number of bands have already been generated, then `CircleNumber` can equal x and the algorithm runs from that set of points on, rather than from the beginning.

   d. `SystemName` is a user-supplied string to be used in the naming conventions for the various data files generated by the algorithm.

   e. In Line 2, the ODE filename is converted from string input into a function, so that, later on, it can be passed to other functions, especially the integration function, ode45. The ODE filename is also saved in a data file so that it can be loaded outside of this program if necessary.

```
4. if CircleNumber == 0
5.     [Saddle, eVects, eVals, negevals, negevects] = feval(basicsfile);
6.     [Cr, steps, SaddleMatrix]= initcir(Saddle, negevects,...
               rhsfilename, SystemName);
7.     Cp = SaddleMatrix;
```

   a. If the algorithm is to start from the saddle point (`CircleNumber ==0`) then the basic information to run the algorithm is obtained from the `basicsfile`. This includes the saddle point, eigenvalues and their eigenvectors and specifically, the negative eigenvalues and their eigenvectors which are used in the generation of the stable manifold. (Positive eigenvalues and their eigenvectors are used for unstable manifold generation.)

b. The `initcir` routine (Lines 45-a) is called to calculate the initial circle around the saddle: `Cr` will be a matrix whose columns are the points of the discrete circle around the Saddle. It is calculated using the eigenvectors associated with the negative eigenvalues (negevects). This is the initial discrete circle around the Saddle on the stable eigenspace and hence really close to being on the stable manifold of the saddle.

c. Here, `Cp`, which denotes the "previous" circle, is the saddle point, which is the "circle" prior to the initial discrete circle. In the `initcir` routine, the saddle is "transformed" into a matrix, each column of which contains the saddle point. This is done for computational reasons. It has dimensions `(3 x steps)` where `steps` is the number of points generated in the initcir routine for the initial circle

```
8.     DrawBand(:,:,1) = Cp;
9.     DrawBand(:,:,2) = Cr;
10.    CalcFromBand(:,:,1) = Cp;
11.    CalcFromBand(:,:,2) = Cr;
12.    drawwidth = lengthCr;
13.    drawwidthfn = [SystemName, 'DW_0.mat'];
14.    save(drawwidthfn, 'drawwidth')
15.    DrawBandFN = [SystemName, 'DrawCircle_0.img'];
16.    multibandwrite(DrawBand,DrawBandFN,'bsq','precision','double');
17.    width = length(Cr);
18.    widthfilename = [SystemName,'BW_0.mat'];
19.    save(widthfilename, 'width');
20.    CalcFromFN = [SystemName, 'CalcFrom_0.img'];
21.    multibandwrite(CalcFromBand,CalcFromFN,'bsq','precision','double');
```

a. Lines 8-21 above store the previous circle (saddle matrix here) and the current circle and save data in a specific file name for use in recreating the image later and also, for use in calculating from data from a specific circle. `Cr` and `Cp` will have the same number of points. Later on, points might be added to `Cr` to refine the mesh and maintain accuracy in calculating the next circle, `Cb`. This is the reason for saving data in separate files: one for rendering an image of the mesh and one from which to calculate the next circles. So, the number of points in `Cr` and the number of points in the updated `Cr` (`drawwidth` and `width`, respectively) are included in the data file. The `DrawBand` files contain the data points before any points are added to `Cr` and the `CalcFrom` files contain the data points of `Cr` in addition to any points added later to refine the mesh. This is later denoted `Updated_Cr`. Here, `Cr` and the updated `Cr` are the same; however, this is not always the case, so we set up the data storing structure from the beginning.

```
22.      CircleNumber = 1;
```

a. Increment the `CircleNumber` counter by one.

```
23. end  (if CircleNumber == 0)
24. BigDelta =.25;
25. GeoDistance= BigDelta;
26. TotalGeoDist=20;
```

a. An initial guess is made for `BigDelta`, the distance between circles.
b. `GeoDistance` is the current total geodesic distance of the manifold, initially set to the value of `BigDelta`; It is increased by the current value of `BigDelta` after each circle is calculated.
c. `TotalGeoDist` is the total geodesic distance of the manifold to be calculated.

```
27. while GeoDistance < TotalGeoDist
28.      filename=[SystemName,'CalcFrom_',int2str(CircleNumber-1),'.img' ];
29.      widthFN=[SystemName,'BW_',int2str(CircleNumber-1),'.mat'];
30.      load(widthFN);
31.      CalcFromBand=multibandread(filename,[3,width,2],...
                 'double',0,'bsq','ieee-le');
32.      Band(:,:,1) = CalcFromBand(:,:,1);
33.      Band(:,:,2) = CalcFromBand(:,:,2);
```

a. Lines 28-33 load the data from the last two calculated circles to be used in the calculation of the next circle.

```
34.      [Cb, BigDelta,Next_BigDelta] = nextcir(Band,odefunction,BigDelta);
```

35. `Band(:,:,3) = Cb;`

   a. Calculate the next circle at a distance `BigDelta` from the last circle with a call to the `nextcir` routine (Line 71).

   b. Store the newly created circle in the third page of the matrix structure which stores the circles, `Cp`, `Cr`, and now `Cb.`

36. `MinMeshDistance = .2;`

37. `MaxMeshDistance = 2;`

   a. A minium and maximum distance between adjacent mesh points in a discrete circle are specified. If the mesh points are too close to one another, then the algorithm will get confused when finding the next circle. If the mesh points are too far from one another, then there is a risk that the accuracy of the manifold will be compromised.

38.
```
[Updated_Cr,Updated_Cb]=MeshQuality(Band,MinMeshDistance,...
    MaxMeshDistance,BigDelta,odefunction,CircleNumber,SystemName);
```

   a. A call to the `MeshQuality` routine (Line 364) checks the quality of the newly calculated mesh points and draws the mesh when the quality is acceptable. New points might be added to the mesh or existing mesh points may be deleted.

39. `BigDelta = Next_BigDelta;`

40. `GeoDistance = GeoDistance + BigDelta;`

41. `Band = zeros(3,length(Updated_Cb),3);`

42. `CircleNumber = CircleNumber + 1;`

   a. Set `BigDelta` to the value, `Next_BigDelta`, determined by the nextcir routine. The new value of `BigDelta` might be larger or smaller than its previous value, depending on the results of the `Accuracy` routine called from within the `nextcir` routine. If one new point in the circle currently being calculated does not meet accuracy requirements (explained more later), then `BigDelta` is decreased and the new circle is re-calculated from the beginning. This must be done so that each point in the new circle is the same distance from the previous circle. The new value for `BigDelta` is stored in `Next_BigDelta` and returned when the `nextcir` routine exists properly.

   b. Increment the current total geodesic distance, `GeoDistance`, by `BigDelta`.

c. Reset the third page of the `Band` matrix structure to a (`3 x length(Updated_Cb)`) matrix of zeros. The `MeshQuality` routine called above returns the updated data sets for the circles, `Cr` and `Cb`, in the variables `Updated_Cr` and `Updated_Cb`. These are also stored in a file (with a filename specific to the circle number to **which** the `Cb` data corresponds) and recalled at the beginning of this `while` loop. The data from the file is read into the `CalcFromBand` variable on Line 31 above, and the first and second pages of `Band` are updated so that: (1) `Cr` is now `Cp` and stored in the first page, (2) `Cb` is now `Cr` and stored in the second page, and (3) the third page is now ready to store the data of the next circle to be computed.

d. Increment the number of the circle we are on currently. This variable is mainly used for creating filenames for data that are specific to a circle's number in the mesh, with circle 0 being the initial discrete circle around the fixed point.

43. `end` (end the `while` loop)

44. ————————————————————————

45. Now the `initcir` routine is examined in detail. It is a subfunction within the `Manifold.m` **file.**

46. `function [Cr, steps, SaddleMatrix] = initcir(Saddle, negevects,...`
    `odefilename, SystemName)`

    a. This routine calculates the initial circle around the saddle point, approximating the local (un)stable manifold, $W^s_{loc}(x_0)$ $(W^u_{loc}(x_0))$, in the (un)stable eigenspace.

47. `delta=1;`

48. `steps=20;`

49. `SaddleMatrix = zeros(3,steps+1);`

    a. `delta` is the radius of initial discrete circle; i.e. the distance from the saddle point.

    b. The value of `steps` is the number of discrete points in the initial circle, a distance of `delta` from the saddle.

    c. Make a (`steps x 3`)-matrix that has the saddle point (vector) as each of its columns so that the vector loop below this will work.

50. `for j=1:steps+1`

```
51.     SaddleMatrix(:,j)=Saddle(1,:)';
52. end
```

a. Transpose the saddle row vector, `Saddle(1,:)`, so that it is a column vector to match dimensions for the `SaddleMatrix`.

```
53. theta = 0:2*pi/(steps):2*pi;
54. Cr = delta*(negevects(:,1)/norm(negevects(:,1))*cos(theta)
        + negevects(:,2)/norm(negevects(:,2))*sin(theta)) + SaddleMatrix;
```

a. The initial circle, `Cr` is formed using the saddle point and the 2 negative (positive) eigenvalues, `negevals (posevals)`, which were acquired from calling the `basics` routine on Line 5, right before the call to this routine. Since this initial discrete circle around the saddle is on the stable eigenspace it is close enough to be considered "on" the stable manifold of the saddle. The formula for `Cr` is as follows:

$$\begin{aligned}\texttt{Cr} = \textbf{delta} \quad &\cdot \quad \left[\frac{negevects(:,1)}{norm(negevects(:,1))}\cos(theta)\right.\\ &+ \quad \left.\frac{negevects(:,2)}{norm(negevects(:,2))}\sin(theta)\right]\\ &+ \quad \textbf{SaddleMatrix}\end{aligned}$$

Using cosine and sine functions and defining `theta` as a vector containing values from 0 to 2π in increments of $\frac{2\pi}{steps}$, a set of discrete points all at a distance `delta` from the saddle point is generated: i.e. a discrete circle having the saddle point as its center and radius equal to `delta`. The variable `Cr` is a matrix whose columns store the points of this first discrete circle. Each column of `SaddleMatrix` contains the saddle vector and this matrix has the same number of columns as the value of the variable `steps`, which is the number of points being generated in this initial discrete circle. (Also, the distance between any two mesh points in `Cr` is the same.)

```
55. i=1:steps;
56. Cr=Cr(:,i);
57. SaddleMatrix = SaddleMatrix(:,i);
```

a. Here, we get rid of the last entry of `Cr` which is a repeat of the first, and re-size `SaddleMatrix` to agree with the dimension of `Cr`.

58. `for j = 1:steps`
59. `    fprintf('%2.0f -- ', j);`
60. `end`
61. `fprintf('\n');`

a. Simply print to the screen the number associated with each point around the circle.

62. `initBand(:,:,1) = zeros(3,steps);`
63. `initBand(:,:,2) = SaddleMatrix;`
64. `initBand(:,:,3) = Cr;`

a. Storing the last three circles in the 3-page matrix structure `initBand`. The first 'circle' is a matrix of zeros. The second circle is a matrix, the columns of which each contain the saddle point (vector). The third circle is the one just calculated around the saddle point.

65. `MinMeshDistance = .2;`
66. `MaxMeshDistance = 2;`

a. These serve the same purpose (for the initial circle) as the variables of the same name that are used in the main file.

67. `figure('Position', [10, 400, 400, 300], 'Units', 'inches')`

a. Set up a MatLab figure in which to draw the mesh.

68.
```
[SaddleMatrix, Cr] = MeshQuality(initBand,MinMeshDistance,...
            MaxMeshDistance, delta, odefilename, 0, SystemName);
```

a. Call the `MeshQuality` routine (Line 364) to ensure that the distance between mesh points is within the lower (`MinMeshDistance`) and upper (`MaxMeshDistance`) bounds. The `MeshQuality` routine is also a subroutine within the `Manifold.m` file and is explained later.

69. ---

70. The next routine to be explained is `nextcir.` It is also a subfunction within the `Manifold.m` file. Recall that **this** routine was called on Line 34 above.

71. `function [Cb,BigDelta,Next_BigDelta]=nextcir(Band,TheOdeFile,BigDelta)`

    a. The `nextcir` routine takes, as input: (1) the data points of the last 3 calculated circles (stored in `Band`), (2) the file name (`TheOdeFile`) of the ODE system, (3) and the value of `BigDelta` to calculate the next **circle,** `Cb`, which is a distance of `BigDelta` away from the last circle, `Cr.`

72. `AccuracyChecker = 0;`

73. `BDIncrease = 5;`

    a. Initialize the `AccuracyChecker` variable to 0. When the `Accuracy` routine is called, if the next point **in** the mesh that was found passed the accuracy test, then `AccuracyChecker` is set to 1 and the algorithm **can** start finding the next new point in the mesh. Otherwise, if the new point does not pass the accuracy test, `AccuracyChecker` is left at a value of 0 and the value of `BigDelta` is decreased. Then, the algorithm starts over on calculating the current circle since all the new points to be calculated must now be at a distance of (the new) `BigDelta` from corresponding points on the previous circle.

    b. The variable, `BDIncrease`, helps to keep track of whether or not the `Accuracy` checker can increase `BigDelta` when finished.

74. `while AccuracyChecker == 0`

    a. Continue the loop until the next point in the mesh is calculated and passes the accuracy checks.

```
75.     lengthCr = length(Band);
76.     Rotated_Band = Band(:,:,2);
77.     Cb = zeros(3, lengthCr);
```

    a. `Rotated_Band` contains the data points from the last calculated circle and is manipulated later on so that the point, `r`, we are currently considering in `Cr`, is always the first column of this matrix. Hence, the points from the last circle from which we are computing the next circle, are "rotated" around to the first column position in

`Rotated_Band`. This is done so that it is simple to identify the position of points relative to the current point, `r`. This helps greatly with indexing issues regarding the shooting method described in the next subroutine, `Shooting_For_b`.

b. `Cb` is the (`3 x lengthCr`)-matrix that will contain the points, `b`, of the next new circle in the mesh. Here it is initialized to hold zeros.

```
78.    for k = 1:lengthCr
79.        b = Shooting_For_b(Rotated_Band, BigDelta, TheOdeFile, k);
80.        if b == [0;0;0]
81.            disp('We didn"t converge...now what??');
82.            Cb = zeros(3, lengthCr);
83.            AccuracyChecker = 1;
84.            break;
85.        end
```

a. Call the `Shooting_For_b` **routine** (Line 109) to find the next point, `b`, in the mesh, corresponding to the current point, `r` in `Cr`. This for loop runs through all the points in `Cr`.

b. Unfortunately, there are times when the code in lines 80-85 does get executed. This happens when the algorithm is unsuccessful in finding a next point. The `break` statement exits out of this `for` loop and an error will result because `Cb` matrix was not completely calculated. (The `Cb` matrix is reset to zeros and this will cause the `MeshQuality` routine to have a heart attack and the program will exit.) Setting `AccuracyChecker` to 1 ensures that we exit out of the `while` loop. The reasons for why the algorithm may not find appropriate mesh points is discussed in the `Shooting_For_b` routine.)

```
86.        fprintf('%2.0f ...  ', k);
87.        Pkminus1 = Band(:,k,1);
88.        Pk = Band(:,k,2);
89.        Pkplus1 = b;
```

```
90.       [AccuracyChecker, Next_BigDelta, BDIncrease]
                    = Accuracy(Pkminus1, Pk, Pkplus1, BigDelta, BDIncrease);
```

a. If the code is successful, then the number, `k`, of the point that was found is printed, so that the user knows **the** status of the algorithm.

b. The `Accuracy` routine (Line 341) is called to ensure that the point that was just calculated passes the accuracy check. In order to do the accuracy check, the following need to be passed to the `Accuracy` routine:

   i. `Pkminus1`*:* the point in the k**th** position on the previously calculated circle, `Cp`. `Cp` is stored in `Band(:,k,1)`, the k**th** column of the 1**st** page of the `Band` matrix structure.

   ii. `Pk`**:** the point in the k**th** position on the current circle, `Cr`, from which we are calculating the next circle. `Cr` is stored in `Band(:,k,2)`, the k**th** column of the 2**nd** page of the `Band` matrix structure.

   iii. `Pkplus1`**:** the point in the k**th** position on the newly calculated circle. This is simply the point, `b`, that was just calculated during this k**th** step of the `for` loop.

c. These points are used to form an angle that measures how accurate the new mesh point is. This is explained more below in the `Accuracy` routine. The `Accuracy` routine returns (1) a value for the variable `AccuracyChecker`, (2) a value for the next value of `BigDelta` (`Next_BigDelta`), and (3) an updated value for `BDIncrease` which helps keep track of when `BigDelta` has been increased/decreased.

```
91.       if AccuracyChecker == 1
92.           disp('It''s a good point!');
93.           Cb(:,k) = b;
94.           Prev_Band = Rotated_Band;
95.           j=1:lengthCr-1;
96.           Rotated_Band(:,j) = Prev_Band(:,j+1);
97.           Rotated_Band(:,lengthCr) = Prev_Band(:,1);
98.       elseif AccuracyChecker == 0
99.           disp('We have to shrink BigDelta and reshoot.');
```

100. `            BigDelta = Next_BigDelta;`

101. `            BDIncrease = 5;`

102. `            break;`

103. `        end`

   a. If the `Accuracy` routine returns a value of 1 for the `AccuracyChecker` variable, then the calculated point, ***b***, is accepted as "a good point" and the k**th** column in the `Cb` matrix is assigned the value of ***b***. `Rotated_Band` is stored in a matrix called `Prev_Band` so that the columns of `Rotated_Band` can be "shifted" or "rotated" in order for the next point, `r`, of the `Cr` circle (which is currently stored in `Rotated_Band`) can be in the first column of `Rotated_Band`. As described previously, this helps with indexing issues in the `Shooting_For_b` routine.

   b. If the `Accuracy` routine returns a value of 0 for the `AccuracyChecker` variable, then the calculated point, `b`, is not accepted. Instead, the value of `BigDelta` is given a new smaller value (`Next_ BigDelta`), which was returned by the `Accuracy` routine. Then, the algorithm breaks out of the loop and restarts the circle over again with the new value of `BigDelta`. The variable `BDIncrease` is reset to some number not equal to zero. In the `Accuracy` routine if one of the angle conditions is larger than the allowed minimum value, `BDIncrease` is set to zero and this tells the accuracy routine that it's not a good idea to increase `BigDelta`. On the other hand, if the angle condition is less than (or equal) to the allowed minimum value, then `BDIncrease` would have its initial value of 5 ($\neq 0$), and this would tell the `Accuracy` routine that it is alright to increase `BigDelta` (double it). For information on the angle conditions, refer to the `Accuracy` routine explained below.

104. `    end` (end the `for` loop in Line 78)

105. `end` (end the `while` loop in Line 74)

106. `fprintf('\n');` (print-to-screen a line break)

107. ————————————————————————

108. The next routine to be explained is `Shooting_For_b`. It is also a subfunction within the `Manifold.m` file. Recall that this routine was called on Line 79 above. This routine is the heart of the algorithm, the part that finds new mesh points, `b`. This subroutine

takes care of the majority of the steps highlighted in the Algorithm section above, such as (1) defining the plane transversal to the last circle at `r`, the normal of which is defined by the points to the right and left of `r` in the circle, and (2) actually shooting to find the point on the plane that is on the manifold and at a distance `BigDelta` from `r`.

```
109. function [b]=Shooting_For_b(Rotated_Band,BigDelta,OdeFile,pointnumber)
110. clear j;
111. r = Rotated_Band(:,1);
112. rLeft = Rotated_Band(:,end);
113. rRight=Rotated_Band(:,2);
114. N = rRight - rLeft;
115. save('r_BigDelta_N.mat', 'r', 'BigDelta', 'N');
```

a. `j` is a counter used in the `while` loop below. It's value is cleared from memory each time this routine is called to ensure that the counter value starts at the value assigned during the loop.

b. As mentioned previously, the first column of `Rotated_Band` will always contain the current point, `r`, from which we are calculating the next mesh point, `b`. The `nextcir` routine guarantees this.

c. `rLeft` the point to the left of `r` in `Cr` will always be the last column of the `Rotated_Band` matrix. The `nextcir` routine guarantees this.

d. `rRight` the point to the right of `r` in `Cr` will always be the second column of the `Rotated_Band` matrix. Again, the `nextcir` routine guarantees this.

e. `N` is the normal vector of the plane, F_r, at `r` and is defined as the vector `rRight - rLeft` and thus points in the direction to the right of the plane. This is important in later calculations that determine on which side of the plane the endpoint of a trajectory is. Note, `rRight - rLeft` is only an approximation to the normal vector, since the actual vector is tangent to `r`. If the mesh points are close enough, then this approximation is good enough.

f. The variables, `r`, `BigDelta`, and `N` are all saved in the file `r_BigDelta_N.mat` to be recalled in other subfunctions where these variables are needed. Another option would be to make these global variables.

116. `y0=r;`

117. `epsilon=.01;`

118. `BigDeltaUpperBnd = (1+epsilon)*BigDelta;`

119. `BigDeltaLowerBnd = (1-epsilon)*BigDelta;`

   a. `y0` is the current initial condition of the ODE system from which the algorithm will attempt to find an appropriate intersection with the plane. The first `y0` is set to `r`. Throughout the course of the `while` loop below, `y0` will be set to many other values while the algorithm searches for the point, `b`.

   b. For practical numerical reasons, there are upper and lower bounds on `BigDelta` since it is unlikely that the point `b` will be exactly `BigDelta` from `r.` Hence, an ε-neighborhood is defined around `BigDelta`, using the variable `epsilon`. The upper bound is defined as `BigDeltaUpperBnd` and the lower bound is `BigDeltaLowerBnd.`

120. `tspan=[0 -.2]`

121. `StopValue=0;`

122. `Init_Step = .01;`

123. `StopThis = 0;`

   a. `tspan` is the interval of time for which to integrate the ODE system in the attempt to find a trajectory that crosses the place at a distance `BigDelta` from `r`. The correct amount of time needed will vary from system to system. Some experimentation might be necessary. It is also helpful to plot some of the trajectories for a particular set of shooting attempts for a particular `r`.

   b. In the following `while` loop, when the next mesh point, `b`, is found, `StopValue` is set to a value of 1 and the loop breaks. This means that we can move on with finding the next point in `Cb`. Initially, `StopValue` is 0, meaning that `b` has not yet been found.

   c. `Init_Step` provides the initial integration step size for the numerical ODE solver used below.

   d. The variable, `StopThis`, which is initialized to a value of zero, is set to ensure that the `while` loop below is not an infinite loop and that if an appropriate point, `b`, cannot be found after the specified number of iterations, then this loop exits and, in turn, the program exits. Now for the main `while` loop:

124. `while StopValue==0`

125. `options = odeset('Events',@FrzeroEvent, 'InitialStep',...`
     `Init_Step, 'Refine', RefiningNum);`

126. `[T,Y, TE, YE, IE] = ode45(OdeFile,tspan, y0, options);`

a. `options` defines some of the options to be passed to the numerical integration solver, ode45, specifically it lets the solver know that there is an events function, `FrzeroEvent`. The solver will then locate times and corresponding solutions where these functions are zero.

b. The above integration step is the most important part of this algorithm. The numerical integration solver, `ode45`, is "based on an explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair. It is a one-step solver - in computing $y(t_n)$, it needs only the solution at the immediately preceding time point, $y(t_n - 1)$ [72]." The line

$$[\mathtt{T}, \mathtt{Y}, \mathtt{TE}, \mathtt{YE}, \mathtt{IE}] = \mathtt{solver}(\mathtt{odefun}, \mathtt{tspan}, \mathtt{y0}, \mathtt{options})$$

integrates the ODE system, specified in `OdeFile`, for the allotted amount of time, `tspan`, starting from the initial condition, `y0`, using the options specified above. In addition, since the `options` included an event function, it also finds where certain functions, called event functions, are zero. The event function `FrzeroEvent` is described below after the end of this subroutine.[72] `T` is the column vector of times at which the solution, `Y`, of the ODE was calculated. The i^{th} entry of `T` corresponds to the solution point in the $i^{\mathbf{th}}$ row of `Y.` `TE` is a vector of times at which the events occur. The rows of `YE` contain the solution point at the corresponding time in `TE` that the event occurred, and `IE` is the index of the event function for which the event occurred.

c. There are 3 event functions: (See also the events function, `FrzeroEvent`, on Line 328 below this `Shooting_For_b` subroutine.)

i. Event Function 1: The function that calculates the distance between a solution point and the plane: If this function is zero, then this means that the solution lies on the plane and an intersection of the solution trajectory and the plane has occurred. However the integration does not stop because there may be other intersections or there may be other events, described next, that need to be recorded. If this is the $i^{\mathbf{th}}$ even to occur, then the $i^{\mathbf{th}}$ entry of `IE` will contain a 1.

ii. Event Function 2: The function that calculates whether the derivative of the solution is zero with respect to the direction of the normal vector, `N`, of the plane. Zeros of this function do not necessarily imply that the solution is on the plane. It simply means that the vector tangent to the solution is pointing in the same direction as the plane. This solution may, in fact, be far away from the plane. However, it might be the case where the point we are seeking is located at the point where the trajectory is tangent to the plane. A tangent intersection like this will not necessarily be caught by the first event function, so we have to check for them this way. Again, integration is not stopped when such solutions are found, so that other events, if any, can be detected. If this is the $i^{\mathbf{th}}$ even to occur, then the $i^{\mathbf{th}}$ entry of `IE` will contain a 2.

iii. Event Function 3: The function that calculates the distance between the solution and `r` and compares it to the value of `BigDelta`. If this function is zero, then the solution point is at a distance `BigDelta` from `r`. However, like with ii. above, this does not guarantee that this solution point is on the plane. It simply means that a trajectory has entered the `BigDelta` (+/- epsilon) ball around `r`. This check, however, is necessary because the trajectory originating at `r` sometimes shoots straight out into the plane. For some reason, however, the other event functions do not register these as intersections nor points of tangency, so these obvious intersections are found by first checking the distance between the solution and `r`. If the solution is at a distance of `BigDelta` away, this event will locate where/when that happens. Then, after the integration has finished, the distance of this solution to the plane is checked. If it is "on" the plane, i.e. within a very small neighborhood of the plane, then this point is the point, `b`, we were seeking. Again, integration is not stopped when such solutions are found, so that other events, if any, can be detected. If this is the $i^{\mathbf{th}}$ event to occur, then the $i^{\mathbf{th}}$ entry of `IE` will contain a 3.

```
127.    for i=1:length(IE)
128.        if IE(i)==3
129.            YE1 = YE(i,:)';
130.            rydistance = norm(YE1-r);
131.            FRdist = dot(N/norm(N), (YE1 - r));
132.            if (abs(FRdist)<.00001
                    && BigDeltaLowerBnd <= rydistance
                    && rydistance <= BigDeltaUpperBnd);
133.                b = YE1;
134.                StopValue = 1;
135.                break;
136.            end
137.        end
138.    end
139.    if StopValue==1;
140.        break;
141.    end
```

a. The `IE` vector is processed to find if there were any entries that had the value of 3, corresponding to events occurring from the 3^{rd} event function described in Line 126.c.iii. It is processed first because it is an easy event to process, with only the distance from the plane having to be checked. If the desired solution is found with this check, no time is wasted on processing the other more involved event functions. Recall that this event was mainly to troubleshoot the problem with shooting from r and finding intersections of that trajectory with the plane defined at `r`.

b. If the $i^{\mathbf{th}}$ entry of `IE` equals 3, then set the candidate solution for `b`, `YE1`, equal to the transpose of the corresponding $i^{\mathbf{th}}$ row in the `YE` matrix. This is done so that `YE1` is a column vector containing the solution point that triggered the $3^{\mathbf{rd}}$ event function.

c. `rydistance` is the distance (norm) between `YE1` and `r`. `FRdist` calculates the distance between the plane and `YE1`, calculated as the dot product of the normal, `N` (normalized), and the vector (`YE1-r`). If this function is close to zero, then `YE1` is on the plane. Furthermore, as a double check confirming the event function results, we again make sure that `YE1` is within epsilon of `BigDelta` away from `r`, (i.e. `BigDeltaLowerBnd` < `rydistance` < `BigDeltaUpperBnd`). Then, if this confirmed, `YE1` is the point, `b`, we've been looking for. Thus, set `b`=`YE1` and break out of the `for` loop that is processing the `IE` vector. In addition, set the `StopValue=1`, so the `while` loop will stop at the next `break` statement, which is made available just after this `for` loop. If `StopValue=1`, then a `break` statement is executed and the outer `while` loop is stopped. Then the `nextcir` routine carries on with the next steps, which involve calling this routine once again...and again...and again....

```
flagIE=0;
for i=1:length(IE)
    if IE(i)==1
        flagIE =1;
        break;
    end
end
```

a. Now the `IE` vector is processed again to locate any entries having the value of 1. Initially, the variable `flagIE` is set to zero, which means that no crossings of the plane by the trajectory occurred. (i.e. no events occurred with the $1^{\mathbf{st}}$event function). If an entry of `IE` is equal to 1, then `flagIE` is set to 1 to denote that a crossing occurred. Since we only want to know if at least one crossing occurred, as soon as an entry of `IE` is found with a value of one, the `flagIE` is set to one and we break out of the processing `for` loop and continue along with the next steps in this routine.

(Note: If `r` is the point from which we are shooting then the first entry in `IE` will always be a 1 since `r` is on the plane by definition of the plane.)

```
149.    for i=1:length(IE)
150.        if IE(i) == 2
151.            YE1 = YE(i,:)';
152.            tanpointFRdist = dot(N/norm(N), (YE1 - r));
153.            rydistance = norm(YE1-r);
154.            if (abs(tanpointFRdist)<.00001
                    && BigDeltaLowerBnd <= rydistance
                    && rydistance <= BigDeltaUpperBnd)
155.                b = YE1;
156.                StopValue = 1;
157.                break;
158.            elseif (rydistance > BigDeltaUpperBnd)
159.                break;
160.            end
161.        end
162.    end
163.    if StopValue==1;
164.        break;
165.    end
```

a. The `IE` vector is now processed for the 3^{rd} and final time to determine if any of the events of the 2^{nd} event function occurred. These are the solution points where the vector tangent to the trajectory pointed in the same direction as the plane, indicating a possible (but usually not) point of tangency with the trajectory and the plane. These points still need to be checked.

b. The possible solution for `b`, `YE1`, is once again set to the transpose of the i^{th} row of the `YE` matrix which contains the solution points at times when events occurred. The distance between this "tangent" point, `YE1`, and the plane is checked in `tanpointFRdist` formula. Also, the distance between `YE1` and `r` is checked with the `rydistance` formula.

c. If `YE1` is on the plane (i.e. `abs(tanpointFRdist<.00001)` and is at a distance `BigDelta` from `r`,
(i.e. `BigDeltaLowerBnd` $\leq$ `rydistance` $\leq$ `BigDeltaUpperBnd`**)**,
then this is the point, `b`, we have been looking for. So, set `b=YE1` and set `StopValue =1` so that the `while` loop will stop. A `break` statement is needed to exit the `for` loop that was processing the entries of `IE` for values of 1. After the `for` loop, there is a check to see if `StopValue ==1`, and if it is, a `break` statement is executed and the outer `while` loop is stopped. Then the `nextcir` routine carries on with the next steps, which involve calling this routine once again...and again...and again... Note: the absolute value of `tanpointFRdist` is used to check distance from the plane. This is done because the value of `tanpointFRdist` might be negative, since the sign of `tanpointFRdist` is simply an indication of the direction in which the (`YE1-r`) vector is pointing: negative implies it is pointing to the left of the plane, positive implies it is pointing to the right of the plane. However, if the value of `tanpointFRdist` is really small in absolute value, this is an indication that `YE1` is close enough to be considered 'on' the plane. However, numerically this small value could be negative or positive, so that is why we use absolute value here to check the distance.

d. Else, if `YE1` is not on the plane and instead it is at a distance larger than `BigDelta` away from `r`, then do not continue looking at any other solution points that satisfy this "tangency" condition since the distance from `r` will only be larger. Again, recall that these "tangent" points are checked b/c the event locator has trouble detecting "obvious" intersections, usually those of the trajectory starting at `r`.

```
if flagIE == 0
    TF = isnan(Y);
    for k=1:length(TF(:,1))
        if TF(k,:)  == [0 0 0]
            last_y = Y(k,:)';
        else
            break;
        end
    end
```

a. Recall that a value of 0 for `flagIE`, means that there were no direct intersections of the current trajectory (having initial condition `y0`) with the plane. Also recall, that `Y` is the solution matrix for this initial value problem, the rows of which are the individual solution points. Thus, we now need to make sure all the points listed in the `Y` vector are valid numbers and not `"NAN"` (Not a Number). This has to be done because sometimes, solutions blow up and then entries of the solution vector, $\boldsymbol{Y}$, do not contain valid numbers.

b. `TF` is equal to a matrix whose entries are 1 (True) if the corresponding entry in `Y` is `NAN`, and 0 (False) if the corresponding entry in `Y` is a valid number. So, this `for` loop runs through the `TF` matrix and determines the index of valid entries in `Y`, setting the variable `last_y` to be the transpose of the $k^{\mathbf{th}}$ row of `Y` so that `last_y` is a column vector. During the loop, if the $k^{\mathbf{th}}$ row of `TF` is no longer a zero vector, indicative of valid entries in the $k^{\mathbf{th}}$ row of `Y`, then the `for` loop exists with a `break` statement and the value given to `last_y` at the `k-1` step is the last valid entry in the `Y` vector. Otherwise, it might be the case that all entries of `Y` are valid, in which case the same thing is accomplished and the loop exits via the counter, `k`.

```
    SideNum = int2str(sign(dot(N/norm(N), (last_y - r))));
    switch SideNum
        case {'-1'}
            leftEP = y0;
            EPexistVal = exist('rightEP');
            if EPexistVal ==1
                rightEP = rightEP;
                y0 = (leftEP + rightEP)/2;
            else
                if norm(y0-r)==0
```

```
185.                        y0 = Rotated_Band(:, 2);
186.                        j = 2;
187.                    else
188.                        y0 = Rotated_Band(:, j+1);
189.                        j = j+1;
190.                    end
191.                end
192.            case {'1'}
193.                rightEP = y0;
194.                EPexistVal = exist('leftEP');
195.                if EPexistVal == 1
196.                    leftEP = leftEP;
197.                    y0 = (leftEP + rightEP)/2;
198.                else
199.                    if norm(y0-r)==0
200.                        y0 = Rotated_Band(:, end);
201.                        j = length(Rotated_Band);
202.                    else
203.                        y0 = Rotated_Band(:, j-1);
204.                        j = j-1;
205.                    end
206.                end
207.        end (end switch statement)
```

a. Since this portion of code is under the case where `flagIE` is zero, there were no crossings of the current trajectory with the plane. Thus, we need to determine on which side of the plane the last point of the trajectory lies, in order to determine the next value for `y0` on `Cr` from which to shoot. `SideNum` is calculated by taking the `sign` of the dot product between (1) the normal vector of the plane, `N`, and (2) the vector formed by, `r`, and the last point, `last_y`, of the present trajectory. `SideNum` converts the value of this dot product into a string having either a value of -1 or 1.

A value of -1 indicates that the point `last_y`, is on the left side of the plane, since `N` points in the direction of `rRight`. (See Line 114 for its definition.) Likewise, a value of 1 indicates that `last_y` is on the right side of the plane.

b. The above `switch` statement selects the next `y0` on `Cr` from which to shoot based on the value of `SideNum` and whether or not the current `y0` is equal to `r` (i.e. `norm(y0,r)==0`).

c. If `SideNum = -1`, then the trajectory ended up on the left side of the plane. Thus, the ideal trajectory that will intersect the plane at the desired value, `b`, must start from an initial condition to the right of the current `y0`. The, the current `y0` becomes the left hand end point, `leftEP`, of the interval in which the 'magic' `y0` will lie. The right hand endpoint, `rightEP`, of this interval is chosen by keeping the current `rightEP` if one exists (hence, the `"exist"` check). If a `rightEP` does not exist, then we simply just select a new `y0` and later on a `rightEP` will be set. The next `y0` is then selected to be a point in the current `Cr` mesh to the right of the current `y0`. If the current `y0`=`r`, (i.e. `norm(y0,r)==0`), (and a `rightEP` does not exist), then the next `y0` to shoot from is selected as the point on the right of `r`, in `Cr`: `y0 = Rotated_Band(:, 2);`. Note that the indexing variable, `j`, is set to 2, to indicate that the current `y0` is in the $j^{\mathbf{th}}$, position of `Rotated_Band`. However, if `y0`$\neq$`r` (and a rightEP does not exist), then the next `y0` to shoot from is selected as the point on the right of `y0`, which will be the `(j+1)`th point from `r`.

d. Similarly, if `SideNum` $=$ **1**, then the trajectory ended up on the right side of the plane. Thus, the trajectory that will intersect the plane must start from an initial condition to the left of the current `y0`. And the current `y0` becomes the right hand end point, `rightEP`, of the interval in which the 'magic' `y0` will lie. The left hand endpoint, `leftEP`, of this interval is chosen by keeping the current `leftEP` if one exists (hence, the `"exist"` check). If a `leftEP` does not exist, then we simply just select a new `y0` and later on a `leftEP` will be set. If the current `y0`=`r`, (i.e. `norm(y0,r)==0`), (and a `leftEP` does not exist), then the next `y0` to shoot from is selected as the point on the left of `r`, in `Cr`: `y0 = Rotated_Band(:, end)`. Note that the indexing variable, `j`, is set to the length of `Cr` (i.e. the number of points in `Cr`), to indicate that the current `y0` is in the $j^{\mathbf{th}}$, position of `Rotated_Band`. However, if `y0`$\neq$`r` (and a `leftEP` does not exist), then the next `y0` from which to shoot is selected as the point on the left of `y0`, which will be the `(j-1)`$^{\mathbf{th}}$ point from `r`.

e. Note that if a `leftEP` and `rightEP` exist (or are defined) in either of the cases above, this interval [`leftEP rightEP`] is bisected, and `y0` is chosen as the midpoint. Hence, this algorithm uses a shooting method coupled with bisection in order to find an appropriate point, `b`, on the plane at a distance `BigDelta` from **`r`.**

```
208.      elseif flagIE==1
```

a. This "`elseif`" indicates that `flagIE=1`, which means there was at least one crossing of the plane by the trajectory. So, this case is now processed.

```
209.          YE = YE';
210.          for i=1:length(TE)
211.              if (IE(i)==2 || IE(i)==3)
212.                  YE(:,i)=[ ];
213.                  TE(i)=[ ];
214.                  break;
215.              end
216.          end
```

a. The rows of the `YE` matrix hold the solutions at the time of an event. This is changed here so that the solutions are now stored in the *columns* of the matrix.

b. Because this is the portion of the code for "`if flag==1`", we want to eliminate all the other entries in `YE` corresponding to events for the 2$^{\mathbf{nd}}$ and 3$^{\mathbf{rd}}$ functions, only keeping the entries of `YE` corresponding to solutions at times when the 1$^{\mathbf{st}}$ event occurred (i.e. when an actual crossing of the plane by the trajectory occurred). So if the i^{th} index of `IE` is *either* 2 or 3, then delete the $i^{\mathbf{th}}$ column of `YE` *and* the $i^{\mathbf{th}}$ entry of the `TE` vector, which contains the times at which events occurred.

```
217.          for i=1:length(TE)
218.              y = YE(:,i);
219.              rydistance = norm(y-r);
```

a. Now, the i^{th} intersection point, `y=YE(:,i)`, is checked to see if it is within `BigDelta` of `r`. As before, this distance is calculated by `rydistance = norm(y-r);`

```
if (rydistance < BigDeltaLowerBnd && i == length(TE))
    TF = isnan(Y);
    for k=1:length(TF)
        if TF(k,:)  == [0 0 0]
            last_y = Y(k,:)';
        else
            break;
        end
    end
```

a. If the distance between `y` and `r`, `rydistance`, is less than `BigDeltaLowerBnd` ***AND*** the current value of the counter index, `i`, is the length of `TE`, (i.e., `y` is the last entry in `YE` to be checked), then we need to determine on which side of the plane the current trajectory ended. This is done, as before, by first finding the last valid entry in the `Y` matrix and setting `last_y` equal to this. Recall that `TF` is equal to a matrix whose entries are 1 (True) if the corresponding entry in `Y` is `NAN`, and 0 (False) if the corresponding entry in `Y` is a valid number. So, this `for` loop runs through the `TF` matrix and determines the index of valid entries in `Y`, setting the variable `last_y` to be the *transpose* of the k^{th} row of `Y` so that `last_y` is a column vector. During the loop, if the k^{th} row of `TF` is no longer a zero vector, indicative of valid entries in the k^{th} row of `Y`, then the `for` loop exists with a `break` statement and the value given to `last_y` at the `k-1` step is the last valid entry in the `Y` vector. Otherwise, all entries of `Y` are valid, in which case the same thing is accomplished and the loop exits via the counter, `k`.

```
    SideNum = int2str(sign(dot(N/norm(N), (last_y - r))));
    if norm(y0 - r) == 0
        switch SideNum
            case {'-1'}
```

```
                    leftEP = r;
                    y0 = Rotated_Band(:,2);
                    j=2;
                    break;
                case {'1'}
                    rightEP = r;
                    y0 = Rotated_Band(:,end);
                    j=length(Rotated_Band);
                    break;
            end %switch
        else
            switch SideNum
                case {'-1'}
                    leftEP = y0;
                    EPexistVal = exist('rightEP');
                    if EPexistVal ==1
                        rightEP = rightEP;
                        y0 = (leftEP + rightEP)/2;
                    else
                        y0 = Rotated_Band(:, j+1);
                        j = j+1;
                    end
                    break;
                case {'1'}
                    rightEP = y0;
                    EPexistVal = exist('leftEP');
                    if EPexistVal == 1
                        leftEP = leftEP;
                        y0 = (leftEP + rightEP)/2;
                    else
```

```
263.                                 y0 = Rotated_Band(:, j-1);
264.                                 j = j-1;
265.                             end
266.                             break
267.                     end (end switch)
268.             end
```

a. Now, as before, `SideNum` is calculated. And a new `y0` is chosen based on whether the current trajectory ended up on the left or right of the plane, with special instructions for selecting the next `y0` when the current `y0` is equal to `r`. See the comments for the code similar to this in lines 175-207. Again, note that if a `leftEP` and `rightEP` exist (or are defined) in either of the cases above, this interval [`leftEP rightEP`] is bisected and `y0` is chosen as the midpoint.

```
269.        elseif(BigDeltaLowerBnd <= rydistance
                && rydistance <= BigDeltaUpperBnd)
```

a. This "`elseif`" statement means that the intersection point, `y`, that we are checking is not only on the plane but at the right distance from `r`.

```
270.            b = y;
271.            StopValue = 1;
272.            break;
```

a. So, `y` is the solution point, `b`, that we are seeking. Thus, set, `b=y` and set `StopValue` to 1, so that the outer `while` loop exits and the `nextcir` routine carries on with the next steps, which involve calling this routine once again...and again...and again...

```
273.        elseif rydistance > BigDeltaUpperBnd
274.            options=odeset('RelTol',1e-4,...
                        'AbsTol',[1e-4 1e-4 1e-5],'InitialStep',...
                        .0001,'MaxStep',.0001);
275.            [tback, yback] = ode45(OdeFile,...
                        [0 .001], y, options);
276.            warning off MATLAB:ode45:IntegrationTolNotMet
```

```
277.                last_y = yback(end,:)';
```

a. This "`elseif`" statement means that the intersection point, `y`, that we are checking is at a distance greater than `BigDeltaUpperBnd`. So for cases like this, in order to determine the next `y0` from which to shoot, we need to know from which side of the plane this trajectory came and not the side where it ended. Thus, we need to take the intersection point, `y`, and integrate *forward* (backward for calculating an unstable manifold) for a small amount of time. Then we set `last_y` to the endpoint, `yback`, of this "sampling" trajectory, if you will, and check on which side of the plane `last_y` lies. Once we have calculated, `yback`, and set `last_y=yback`, the code is very similar to previous code for selecting the next `y0`. Essentially, because this trajectory crossed the plane at a distance greater than `BigDelta`, it forms a natural boundary (on the left or right depending on whether it came from the left or right, respectively) for the left or right endpoint of the interval in which the 'magic' `y0` lies.

```
278.                SideNum = int2str(sign(dot(N/norm(N), (last_y - r))));
```

a. `SideNum` is calculated as before, but here it determines the side from which the current trajectory came, instead of the side where it ended. Again, a value of -1 indicates that the trajectory came from the left of the plane and a value of 1 indicates the trajectory came from the right of the plane.

```
279.                if norm(y0-r)==0
280.                    switch SideNum
281.                        case {'-1'}
282.                            leftEP = r;
283.                            y0 = Rotated_Band(:,2);
284.                            j=2;
285.                        case {'1'}
286.                            rightEP = r;
287.                            y0 = Rotated_Band(:,end);
288.                            j=length(Rotated_Band);
```

```
289.            end %switch
290.        else
291.            switch SideNum
292.                case {'-1'}
293.                    leftEP = y0;
294.                    EPexistVal = exist('rightEP');
295.                    if EPexistVal == 1
296.                        rightEP = rightEP;
297.                        y0 = (leftEP + rightEP)/2;
298.                    else
299.                        y0 = Rotated_Band(:,j+1);
300.                        j = j+1;
301.                    end
302.                case {'1'}
303.                    rightEP = y0;
304.                    EPexistVal = exist('leftEP');
305.                    if EPexistVal == 1
306.                        leftEP = leftEP;
307.                        y0 = (leftEP + rightEP)/2;
308.                    else
309.                        y0 = Rotated_Band(:,j-1);
310.                        j=j-1;
311.                    end
312.            end %switch
313.        end %if normy0=0
```

a. A new `y0` is chosen based on whether the current trajectory *came from* the left or right of the plane, with special instructions for selecting the next `y0` when the current `y0` is equal to `r`. See the comments for the code similar to this in lines 175-207. Again, note that if a `leftEP` and `rightEP` exist (or are defined) in either of the cases above, this interval [`leftEP rightEP`] is bisected and `y0` is chosen as the midpoint.

314. `break;`

315. `end` (end the `if` statement that checks `rydistance`)

   a. If the current intersection point, `y`, meets any of these "`rydistance`" criteria in the above `if/elseif/elseif` statement, then a new `y0` would have been chosen and we can ‘`break`’ out of the current `for` loop and continue with the steps of the `while` loop. Otherwise, the "`rydistance`" criteria are checked for the next intersection point in the `YE` matrix and we continue through the `for` loop.

316. `end` (end the `for` loop)

317. `end` (end the `if/elseif` statement regarding the value of `flagIE`)

318. `StopThis = StopThis + 1;`

319. `if StopThis > 50`

320. `fprintf(’We didn”t converge!  :( \n’)`

321. `b=[0; 0; 0];`

322. `StopValue = 1;`

323. `break;`

324. `end`

   a. The value of the `StopThis` counter is increased by increments of 1 when the next `y0` has been selected. This is to ensure that the `while` loop won’t go on forever. The maximum number of iterations to calculate through the `while` loop is set to 50; Sometimes, a greater number will actually allow for an intersection to be found, but mostly it just wastes time. If the value of `StopThis` gets above 50, then "`We didn’t converge!  :(` " is printed to the screen to alert the user of failure and the `b` is set to a vector of zeros, which will tell the `nextcir` routine after we exit this routine, that something went wrong. Then, the program will abort. To exit the `while` loop, since we failed, set `StopValue=1` and break from the `while` loop. This is somewhat redundant, but it makes sure we don’t keep shooting for points.

325. `end` (end the `while` loop)

326. ---

327. Below, is the events function, `FrzeroEvent`, that was mentioned earlier in Line 126:a-c. Three event functions are specified in the value vector below. Zeros of these functions are found while the ODE system is being integrated.

```
328. function [value,isterminal,direction] = FrzeroEvent(t,y)
329. load('r_BigDelta_N.mat');
330. load('odefunctionname.mat');
331. dydt = feval(odefunction,t,y);
332. v1 = [dydt(1), dydt(2), dydt(3)];
333. yN = v1/norm(v1);
334. NN = N/norm(N);
335. value = [N(1)*(y(1)-r(1)) + N(2)*(y(2)-r(2)) + N(3)*(y(3)-r(3));...
336. NN(1)*yN(1) + NN(2)*yN(2) + NN(3)*yN(3);BigDelta-norm(y-r)];
337. isterminal = [0; 0; 0];
338. direction = [0; 0; 0];
```

a. The values for `r`, `BigDelta`, and `N` are loaded from the file in which they were saved previously. Also, the name of the ODE function is loaded from the `odefunctionname.mat` file.

b. Line 331-333 evaluates the vector field at the solution point, `y`, (`dydt`), stores this value in the vector `v1`, and normalizes it so that the result, `yN`, is a unit vector, pointing in the direction of `v1`. This is used in the 2nd event function, which determines when the dot product of N and the derivative of the trajectory evaluated at a solution is zero. This would indicate that `y` was a "tangency" point. See Lines 126.c.ii for more details.

c. `value`, as mentioned, is a vector containing the three event functions. The first event function is to detect crossings of the trajectory with the plane: i.e. when the dot product of `N` with the vector (`y-r`) is zero. The second event function was described in point `b.` above and the third event function determines when the trajectory is at a distance `BigDelta` from `r`: i.e. when the difference between `BigDelta` and the distance between `y` and `r` (i.e. `norm(y-r)`) is zero.

339. ————————————————————————

340. Below, the `Accuracy` routine is explained.

341. `function [AccuracyChecker, Next_BigDelta, BDIncrease] =`
`        Accuracy(Pkminus1, Pk, Pkplus1, BigDelta, BDIncrease)`

   a. The `Accuracy` routine checks the angle formed by (1) the line through `Pkminus1` and `Pk` and (2) the line through `Pk` and `Pkplus1`, where:

      i. `Pkminus1` is a point in `Cp = Band(:,:,1)`,

      ii. `Pk` is the corresponding point in `Cr = Band(:,:,2)`, and

      iii. `Pkplus1` is the corresponding point in `Cb =Band(:,:,3)`.

   b. See [47, 63, 64] for more details about this accuracy check and the angles checked

342. `alphaMin = 0;`

343. `alphaMax = 0.4;`

344. `BDalphaMin = .1;`

345. `BDalphaMax = 1.0;`

346. `alpha1 = (1/norm(Pkminus1 - Pk))* norm((Pk - Pkminus1)`
`        + (Pk - Pkplus1)*norm(Pkminus1 - Pk)/norm(Pk - Pkplus1));`

347. `BDalpha1 = BigDelta*alpha1;`

   a. Define the minimum and maximum values for the `alpha1` and `BDalpha1` angles and define `alpha1` and `BDalpha1`.

   b. The definition of `alpha1` is from [62], however, it was necessary to modify it since it appears that the formula given in [62] is incorrect. Also, if the angle defined by `alpha1` is zero this means that `Pkminus1`, `Pk`, and `Pkplus1` lie in a straight line, which is the desired outcome. For values of `alpha1` larger than zero, the three points do not lie in a straight line; however up to some reasonable `alpha1` max value the 'curvature' of the line will be acceptable. In [62] and [63] a minimum value of `alpha1` is specified (0.2 for the Lorenz manifold); however, the minimum should be 0 and perhaps the maximum value can be within an acceptable range from 0.2 to 0.3. This particular aspect was not made clear in either of the above papers.

348. `if (alpha1 < alphaMax && BDalpha1 < BDalphaMax)`

```
349.     Next_BigDelta = BigDelta;
350.     AccuracyChecker = 1;
351.     if BDalpha1 > BDalphaMin
352.         BDIncrease == 0;
353.     end
354.     if (BDalpha1 < BDalphaMin) && (BDIncrease ~= 0)
355.         Next_BigDelta = 2*BigDelta;
356.     end
357. elseif (alpha1 > alphaMax | BDalpha1 > BDalphaMax)
358.     fprintf(’alpha1 = %f and BDalpha1 = %f ’, alpha1, BDalpha1);
359.     AccuracyChecker = 0;
360.     Next_BigDelta = BigDelta/2;
361. end
```

a. Check if the angles are within the specified bounds. If they are not, then set `AccuracyChecker` equal to zero and divide the value of `BigDelta` in half and then store `Next_BigDelta`. When the `Accuracy` routine exits and returns to the `nextcir` routine, `Next_BigDelta` will become the new value of `BigDelta`, and the `nextcir` routine must start over with the very first point in `Cr` to calculate the next circle. Even though some good points may have previously been found, if one point fails the accuracy test, then all have to be recalculated so that in the end, all the points in the new circle are at the same distance from the previous circle. If the angles are within the specified bounds, then `AccuracyChecker` is set to 1, and depending on the value of `BDIncrease`, the value of `BigDelta` is doubled or left alone and then stored in `Next_BigDelta`. When the `Accuracy` routine exits and returns to the `nextcir` routine, `b` will be accepted as the next point in the circle, `Cb`, and the `nextcir` routine attempts to find the next point.

b. `BDIncrease` is either 0 or 5. It is initialized at the beginning of the `nextcir` routine to a value of 5, on page 69, Line 73. If Line 351 is true for even one of the new acceptable points in the current circle being calculated, then `BDIncrease` will be set to 0. Then, when it comes time to set the new value of `BigDelta` (within the `nextcir` routine) after all the points of the new circle have been approved, if `BDIncrease` is 0, `BigDelta` will not be doubled and instead simply remain the same. If the value of `BDIncrease` is still 5 after all the points in the next circle have been found, then this means that all the new points met the extra requirement and it is OK to increase (double) `BigDelta`. So, `BDIncrease` is really just keeping track of whether all the new acceptable points, `b`, meet a certain condition beyond the acceptable angle requirement. Note, that this only comes into play when the main accuracy angles are within in the prescribed bounds and `b` is accepted. Again, see [47, 62, 63, 64] for more info.

362. ————————————————————————

363. Next the `MeshQuality` routine is explained. This routine is a subroutine within the **Manifold** m-file and it is responsible for the addition and deletion of mesh points depending, on the distance between successive mesh points. Also, it stores the data from the `Cr` and `Cb` circles, before any mesh points are added or deleted, as well as the data from the updated `Cr` and `Cb` circles, denoted, `Updated_Cr` and `Updated_Cb`. It takes, as input, the 3-d matrix `Band` which contains, `Cp`, `Cr`, and `Cb` and checks to see if points in `Cb` are well spaced. If there are not enough points in `Cb` (i.e. the distance between neighboring points is greater than the `MaxMeshDistance`), then points are added. This is done by first adding points to `Cr` and then calling the `Shooting_For_b` routine for each of these new points in order to find additional points for `Cb`, if needed. If the distance between neighboring points in `Cb` is smaller than the `MinMeshDistance`, then one of them is deleted. To keep the dimensions of each page of the `Band` the same, a zero vector is added to the matrix `Cp` in the position where the point in `Cr` was added. After this routine is finished and the mesh is drawn, `Cp` is set to `Cr` and `Cr` is set to `Cb` and `Cb` is reset to a bunch of zeros to get ready for drawing the next circle! This routine relies on two other subroutines, `NeighborDistance` and `DrawMesh`, both of which are a part of the `Manifold.m` file and appear after the `MeshQuality` routine.

```
364. function [Updated_Cr, Updated_Cb]=MeshQuality(Band,MinMeshDistance,...
             MaxMeshDistance,BigDelta,TheOdeFile,CircleNumber,SystemName)
365. New_CpBand = Band(:,:,1);
```

366. `New_CrBand = Band(:,:,2);`

367. `New_CbBand = Band(:,:,3);`

368. `tempBand = Band;`

   a. The variables `New_CpBand`, `New_CrBand`, and `New_CbBand` are used below to store not only the current data for `Cp`, `Cr`, and `Cb`, but also the extra points that might need to be added. Each is first initialized to the corresponding pages of the `Band` matrix structure.

369. `All_Done = 0;`

370. `while All_Done == 0`

   a. This begins the `while` loop that will determine if new points need to be added or subtracted from the mesh. The reason that this might possibly be a continuous thing (hence the loop) is that, even though points might be added or deleted the first time through, the distance between the new points and the old points in the mesh (or between the points that are left over once some have been deleted) must also meet the mesh distance requirements. So, on the first time through the loop, points are added or subtracted from the entire `Cb` mesh and then the updated `Cb` mesh is checked again until all the points are an acceptable distance apart.

371. `    tempBand = New_CrBand;`

372. `    NbrDist = NeighborDistance(New_CbBand);`

373. `    NumOfPoints = length(NbrDist);`

374. `    k=0;`

   a. The variable `tempBand` is used below to temporarily store points that are added to the `Cr` mesh. Note that the points in `Cr`, because it is the previously calculated circle, already have passed the mesh quality test. However, points need to be added to the `Cr` mesh so that the algorithm has points from which to shoot, to find a corresponding new point, `b`, on `Cb`. Also, when the triangular surface mesh is drawn between `Cr` and `Cb`, these extra points will be necessary. However, we also need the 'old' version of the `Cr` circle that doesn't contain the extra points, so that the triangular surface mesh between `Cp` and `Cr` can be drawn.

b. In order to find the distance between neighboring mesh points of `Cb`, a call to the `NeighborDistance` routine (Line 466) is made. This routine is explained below the `MeshQuality` routine. `NbrDist(i)` is the distance between the points in the `i` and `(i+1)`columns of `New_CbBand`. Also needed is the number of distances calculated, or the number of points in `New_CbBand`, which is simply the length of the `NbrDist`. The counter, `k`, is initialized to zero.

```
375.    for i = 1:NumOfPoints
376.        if NbrDist(i) > MaxMeshDistance
377.            if i == NumOfPoints
378.                fprintf('%2.0f - 1 ', i);
379.                ExtraPoint = (tempBand(:,i) + tempBand(:, 1) )/2;
380.                New_CrBand = [New_CrBand(:,:)  ExtraPoint];
381.                Rotated_Band=[New_CrBand(:,end) New_CrBand(:,1:end-1)];
382.                extra_b = Shooting_For_b(Rotated_Band,...
                            BigDelta, TheOdeFile, NumOfPoints);
383.                if extra_b == [0;0;0]
384.                    extra_b;
385.                    Updated_Cr = Band(:,:,2);
386.                    Updated_Cb = zeros(3,length(Updated_Cr));
387.                    return;
388.                end
389.                New_CbBand = [New_CbBand(:,:)  extra_b];
390.                New_CpBand = [New_CpBand(:,:)  [0; 0; 0]];
```

a. Now we check the distances found in the `NbrDist` vector. If the i^{th} distance in the `NbrDist` vector is larger than the `MaxMeshDistance`, then an extra point will need to be added to `Cr` and we need to shoot from this point to find a corresponding point, `b`, using the `Shooting_For_b` routine. However, depending on which pair of mesh points in `Cr` were too far apart, the point to be inserted will need to be calculated and inserted carefully, since `Cr` is stored, after all, in a matrix with a first entry and a last entry. So...

b. IF the $i^{\mathbf{th}}$ position is the ***last*** position of the `NbrDist` vector, then the `ExtraPoint` will be the point between the $i^{\mathbf{th}}$ (or last) column of `tempBand` (which is just a copy of `New_CrBand`) and the first column. These points are added and divided by two to get the `ExtraPoint`. Then `New_CrBand` is updated so that `ExtraPoint` is inserted after the last column of the `New_CrBand` matrix. We define `RotatedBand` in a similar way as was done in the `nextcir` routine so that the point from which we are shooting, `ExtraPoint`, is always in the first column of this matrix. Then the `Shooting_For_b` routine is called and the `extra_b` point is (hopefully) found. There is a check to make sure that it is a valid extra point, meaning that if the routine returned a zero vector for `b`, then an extra point could not be found and the algorithm aborts, by way of setting the output data, `Updated_Cb`, to a matrix of zeros. If a valid extra point is found, then the `extra_b` point is added after the last column of the `New_CbBand` matrix. In addition, a column vector of zeros is added to the `New_CpBand` in order to make sure that all three matrices, `New_CpBand`, `New_CrBand`, and `New_CbBand` have the same dimension. This is important for storing the data, etc. Also, the initial print statement will tell the user between which points the extra point is being found. `NumOfPoints` gives the current index of the point in `New_CrBand` from which we're shooting.

```
        elseif i ~= NumOfPoints %when i = 1...NumOfPoints-1
            if i==1
                fprintf('\n 1-2 -- ');
            else
                fprintf('%2.0f-%2.0f -- ', i,i+1);
            end
            ExtraPoint = ( tempBand(:, i) + tempBand(:, i+1) )/2;
            pointnumber = NumOfPoints + k;
            New_CrBand = [New_CrBand(:,1:i+k) ExtraPoint ...
                        New_CrBand(:,i+1+k:end)];
            Rotated_Band=[New_CrBand(:, i+1+k:end)...
                        New_CrBand(:,1:i+k)];
```

```
      extra_b = Shooting_For_b(Rotated_Band,...
               BigDelta, TheOdeFile, pointnumber);
      if extra_b == [0;0;0]
         Updated_Cr = Band(:,:,2);
         Updated_Cb = zeros(3,length(Updated_Cr));
         return;
      end
      New_CbBand = [New_CbBand(:,1:i+k) extra_b ...
               New_CbBand(:, i+1+k:end)];
      New_CpBand = [New_CpBand(:,1:i+k) [0; 0; 0] ...
               New_CpBand(:, i+1+k:end)];
      k=k+1;
   end
end
```

a. IF the i^{th} position is NOT the last position of the `NbrDist` vector, then the extra points are added to `Cr`, `Cb`, and `Cp` with different indexing. The previous set of code was the "special" case when the point needed to be added between the first and last points of the mesh. However, here we need to keep track of any points that have been added previously. That is what the `k` counter is for. The "`+ k`" in the lines of code, makes up for the new points that have been added, so that the indexing is correct.

b. For `Cr`, the `ExtraPoint` point is calculated as the point between the `i` and `i+1` points of `tempBand` (which is a copy of the ***current*** `New_CrBand`).

c. The variable `pointnumber` gives the current index of the point in `New_CrBand` and is simply the sum of the values of `NumOfPoints` and `k`. This `Pointnumber` which is passed to the `Shooting_For_b` routine can be used in that routine to draw all the shooting orbits of the initial conditions used to find this particular `b`. This can be a helpful troubleshooting strategy.

d. `ExtraPoint` is inserted in the `New_CrBand` between the `i+k` and `i+1+k` position of the `New_CrBand`.

e. RotatedBand is formulated as usual, with the ExtraPoint in the first column.

f. As before in the previous -if- portion, a call to the Shooting_For_b routine is made and the extra_b point is (hopefully) found. There is a check to make sure that it is a valid extra point, meaning that if the routine returned a zero vector for b, then an extra point could not be found and the algorithm aborts, by way of setting the output data, Updated_Cb, to a matrix of zeros. Now, the difference between the previous -if- part and this -ifelse- part is that if a valid extra point is found, then the extra_b point is added, not after the last column of the New_CbBand matrix, but rather between the i+k and i+1+k columns. In addition, a column vector of zeros is added to the New_CpBand in order to make sure that all three matrices, New_CpBand, New_CrBand, and New_CbBand have the same dimension. The zero vector is added between the i+k and i+1+k columns of the current New_CpBand. This is important for storing the data, etc. Also, the initial fprintf statement will tell the user between which points the extra point is being found.

412. `end` (end the `for` loop)

413. `if k == 0`

414. `    All_Done = 1;`

415. `end`

a. Once the for loop exits, if the value of k is zero, this means that all the mesh distances were in the correct range, and the while loop can be exited, by setting the All_Done variable to 1.

416. **`end` (end the `while` loop)**

417. `NbrDist = NeighborDistance(New_CbBand);`

418. `NumOfPoints = length(NbrDist);`

419. `Deleted= zeros(1,NumOfPoints);`

a. NeighborDistance is called again on the New_CbBand to check for distances between points that are smaller than MinMeshDistance. This check is done after points are added, if any, so that if newly created points put the distance between adjacent points to close together, they can be deleted.

b. `Deleted` is a vector the length of `NumOfPoints`, (which is calculated again on the ***current*** `New_CbBand`) to keep track of any columns (in all three matrices for `Cp`, `Cr`, and `Cb`) that need to be deleted after the drawing is done.

```
for i = 1:NumOfPoints
    if NbrDist(i) < MinMeshDistance
        Deleted(i) = i;
        if i == 1
            fprintf('\n-%2.0f ', i);
            New_CbBand = [New_CbBand(:,2) New_CbBand(:,2:end)];
        elseif i == NumOfPoints
            fprintf('-%2.0f ', i);
            New_CbBand = [New_CbBand(:,1:i-1) New_CbBand(:,1)];
        else %when i == 2..NumOfPoints-1
            fprintf('-%2.0f ', i);
            New_CbBand = [New_CbBand(:,1:i-1) New_CbBand(:, i+1) ...
                        New_CbBand(:, i+1:end)];
        end
    end
end
```

a. Now we check the distances found in the newly calculated `NbrDist` vector. If the $i^{\mathbf{th}}$ distance in the `NbrDist` vector is smaller than the `MinMeshDistance`, then we set the $i^{\mathbf{th}}$ column of `New_CbBand` equal to the point in the $(i+1)^{\mathbf{th}}$ column. Note that depending on what position we are at in the `NbrDist` vector, the syntax for accomplishing this is different; in particular, when `i=1` and `i=NumOfPoints`, corresponding to the first and last columns of the `New_CbBand`, respectively. Later, we actually delete the 'repeated' column from the matrix, using the `Deleted` matrix, which is keeping track of the $i^{\mathbf{th}}$ columns that are 'marked' for deletion. Until then, though, all the points in the matrix are needed for the `DrawMesh` routine. A `fprintf` statement lets the user know which points are being deleted, preceded by a minus sign.

```
435. DrawBand(:,:,1) = New_CrBand;
436. DrawBand(:,:,2) = New_CbBand;
437. DrawMesh(DrawBand);
```

a. Finally, the mesh can be drawn between `Cr` and `Cb`. A call to the `DrawMesh` routine (Line 475) is called with the data from `New_CrBand` and `New_CbBand` passed in the 2-page matrix structure called `DrawBand`. The `DrawMesh` routine is described after this routine and the `NeighborDistance` routine.

```
438. drawwidth = length(New_CbBand);
439. drawwidthfn = [SystemName, 'DW_', int2str(CircleNumber), '.mat'];
440. save(drawwidthfn, 'drawwidth')
441. DrawBandFN = [SystemName, 'DrawCircle_', int2str(CircleNumber), 'img'];
442. multibandwrite(DrawBand, DrawBandFN,'bsq','precision','double');
```

a. Write data to a file for later use to draw again. Note that the filename is specific to the current circle number. Note also, that the variable `drawwidth` which has the value of the length of the `New_CbBand`, is also stored and the filename is also specific to the current circle number. Then, if one wants to re-draw the mesh after having calculated 10 circles, say, then the circles do not have to be re-calculated, but can simply recalled from the data stored in the specific circle number files. As long as the file exists for a particular circle number, it can be drawn.

```
443. j=0;
444. if isempty(Deleted) == 0
445.     for i = 1:length(Deleted)
446.         if Deleted(i) ~= 0
447.             k = Deleted(i);
448.             New_CbBand(:,k-j) = [];
449.             New_CrBand(:,k-j) = [];
450.             New_CpBand(:,k-j) = [];
451.             j=j+1;
452.         end
```

453. `    end`

454. `end`

   a. In order to calculate the next circle, the points in `Cr` and `Cb` that were ‘removed’(See Line 420-434.a) now have to be deleted so as not to confuse things when we save the data used to compute subsequent circles.

455. `Updated_Cr = New_CrBand;`

456. `Updated_Cb = New_CbBand;`

457. `CalcFromBand(:,:,1) = Updated_Cr;`

458. `CalcFromBand(:,:,2) = Updated_Cb;`

459. `width = length(Updated_Cb);`

460. `widthfilename = [SystemName,'BW_',int2str(CircleNumber), '.mat'];`

461. `save(widthfilename, 'width');`

462. `CalcFromFN = [SystemName, 'CalcFrom_', int2str(CircleNumber), '.img'];`

463. `multibandwrite(CalcFromBand,CalcFromFN,'bsq','precision','double');`

   a. Assign the variables, `Updated_Cr` and `Updated_Cb`, to `New_CrBand` and `New_CbBand`, respectively. This is necessary since `Updated_Cr` and `Updated_Cb` are specified as the outputs of the `MeshQuality` routine. Then, save data for `Updated_Cr` and `Updated_Cb` in a multipage matrix structure, `CalcFromBand`, and save this to a file with a filename specific to the current circle number. The files saved here are used to calculate the subsequent circles, so that if one wants to start calculating the mesh from circle 3, circles 0-2 do not have to re-calculated. As long as the file exists for a particular circle number, the mesh can be calculated, starting from that circle number.

464. ————————————————————————

465. Now described is the `NeighborDistance` routine which calculates the distance between adjacent mesh points. The routine takes, as input, a matrix, `Points`, whose columns represent a triple `(x,y,z)` and returns a vector, `NbrDist`, containing the distances between the adjacent points (column vectors). The distances are calculated one at time starting with the distance between points 1 and 2, then 2 and 3 and so on up until

NumOfNeighbors-1, which is the distance between the second-to-last point and the last point in the Points matrix. The distance between the last point and the first point of the matrix, Points, is calculated separately after the loop. This routine is used in the MeshQuality routine for determining whether the mesh points are at a proper distance from one another.

466. `function [NbrDist] = NeighborDistance(Points)`

467. `NumofNeighbors = length(Points);`

468. `NbrDist = zeros(1, NumofNeighbors);`

469. `for i = 1:(NumofNeighbors -1)`

470. `    NbrDist(i) = norm(Points(:,i)-Points(:, i+1));`

471. `end`

472. `NbrDist(NumofNeighbors) = norm(Points(:, NumofNeighbors)-Points(:,1));`

473. ————————————————————————

474. The last routine in the Manifold m-file is DrawMesh, which basically connects the dots of the mesh as a nice triangle surface. The routine takes, as input, a multi-page matrix structure, Band. The first page contains the New_CrBand matrix and the second page contains the New_CbBand matrix, both before any points were deleted. See Lines 435-437.a where this function is called. This function can also be a completely separate m-file, which is useful to call for redrawing the mesh after a number of circles have already been calculated.

475. `function DrawMesh(Band)`

476. `steps = length(Band(:,:,2));`

   a. steps is the number of points (columns) of Cr (and hence also of Cb).

477. `X = [Band(1,:,1) Band(1,:,2)];`

478. `Y = [Band(2,:,1) Band(2,:,2)];`

479. `Z = [Band(3,:,1) Band(3,:,2)];`

   a. X, Y, and Z are vectors that give the X-coords, Y-coords, and Z-coords of points in Cr and Cb, concatenated together for use with the trisurf function for graphing the triangle mesh. The first 'steps' points are from Cr and the 'steps+1' to 'steps

`+ steps`' points are from `Cb`. This indexing is used in the next part when defining the 'face' matrix `TRI`.

480. `TRI = zeros(steps,3);`
481. `TRI2 = zeros(steps,3);`
482. `for j=1:steps-1;`
483. `    TRI(j,:)  = [j steps+j steps+j+1];`
484. `end`
485. `TRI(steps,:)= [steps 2*steps steps+1];`
486. `for j=1:steps-1;`
487. `    TRI2(j,:)  = [j j+1 steps+j+1];`
488. `end`
489. `TRI2(steps,:)= [steps 1 steps+1];`

   a. Now, we form the triangles with vertices from `Cr` and `Cb`. `TRI` is a (`steps x 3`)-matrix that defines the triangles whose vertices are indexed by the matrices `X`, `Y`, and `Z`. For instance, the first row of `TRI` is `[1 21 22]` if `steps=20`. This vector, `[1 21 22]`, says that the first vertex of the first triangle is the 3-vector defined by

      i. the $1^{\mathbf{st}}$ elements in the matrices `X`, `Y`, and `Z`
      ii. the $21^{\mathbf{st}}$ elements in the matrices `X`, `Y`, and `Z` and lastly,
      iii. the $22^{\mathbf{nd}}$ elements in the matrices `X`, `Y`, and `Z`.

   This equates to the 'first point' of the `Cr` circle, the 'first point' of the `Cb` circle and 'second point' of the `Cb` circle. Then the indices are increased in increments of 1 each time through the loop, moving "around the circle". This indexing works for the first '`steps-1`' points, but the last triangle reuses the first point of the second circle which is indexed in `X`, `Y`, and `Z` as the `steps+1` points, so this point is placed in the last row of `TRI` after the loop. NOTE: `steps` is the number of points in `Cr`.

   b. `TRI2` defines the other set of triangles of the mesh in a similar manner. Together, the two sets of triangles form the band between the previous circle, Cr, and the new circle Cb.

490. `trisurf(TRI, X, Y, Z);`

491. `hold on`

492. `trisurf(TRI2,X,Y,Z);`

493. `hold on`

494. `drawnow`

495. `xlabel('X'); ylabel('Y'); zlabel('Z');`

    a. `trisurf` takes the matrix `TRI` that defines the triangles by vertices indexed through `X`, `Y`, and `Z` and creates a surface of triangles.

496. This concludes the `Manifold.m` file code

Two separate files that must be defined in order to run the `Manifold.m` file are the "`basics`" file and the "`rhs`" (i.e. right hand side) file. Below, these files specific to the Lorenz system are given. First the basics code:

```
function [Saddle, eVects, eVals, negevals, negevects] = lorenzbasics
syms X Y Z
sigma = 10; q = 28; beta = 8.0/3;
Xprime = sigma*(Y-X);
Yprime = q*X - Y - X*Z;
Zprime = X*Y - beta*Z;
Jack = jacobian([Xprime; Yprime; Zprime],[X, Y, Z]);
Saddle = [0.0, 0.0, 0.0];
JackAtSaddle = subs(Jack, [X, Y, Z], Saddle);
[eVects,eVals] = eig(JackAtSaddle);
eVals=diag(eVals);
[negevals, negevects] = evf(eVals, eVects);
v1 = negevects(:,1);
v2 = negevects(:,2);
N = cross(v1, v2);
v3 = cross(v1,N);
negevects(:,1) = v1;
negevects(:,2) = v3;
----------------------------------------------------
function [negevals, negevects] = evf(eVals, eVects)
k = length(eVals);
j1=1; j2=1; j3=1;
for i=1:k
    if eVals(i) < 0
        negevals(j1)=eVals(i);
        negevects(:,j1)=eVects(:,i);
        j1=j1+1;
    elseif eVals(i) > 0
        posvals(j2)=eVals(i);
        posevects(:,j2)=eVects(:,i);
        j2=j2+1;
    else
        zeroevals(:,j3)=eVals(i);
        j3=j3+1;
    end
end
```

a. The `basics` file must specify the following information:

   i. The equations of the system and the parameter values (in this file the system and parameters are defined as symbolic variables/parameters so that algebraic computations can be done)

   ii. The fixed point of interest (In this case it is labeled `Saddle`)

b. The "`basics`" file uses MatLab provided subroutines for finding the jacobian matrix (`jacobian`) and evaluating it at the fixed point (`subs`), as well as finding the eigenvalues and eigenvectors (`eig`). A subroutine within the `basics` file, `evf`, sorts the eigenvalues and specifically returns the negative eigenvalues and associated eigenvectors for the Lorenz system; however, all the eigenvalues and eigenvectors are found. The code would just need to be modified slightly to return the positive eigenvectors. Note also that a normalized eigenvector is created with respect to the cross product of the two eigenvectors (associated with the two negative (positive) eigenvalues) and used as one of the eigenvectors to be returned by the `basics` file. This is so that the initial circle calculated in the `Manifold.m` code is a circle and not an ellipse.

The "`rhs`" file must be in the form required by MatLab ode solver functions, such as `ode45`, which is used in the `Manifold.m` file. A `dydt` vector is formed and the variables of the system are indices of `y`: `y(1), y(2), y(3)`, for the `x`, `y`, and `z` notation used in the basics file and in equations 3.1. Below is the simple code for the rhs file for the Lorenz system:

```
function dydt = lorenzrhs(t,y)
sigma = 10.0; q = 28.0; beta = 8.0/3;
dydt = zeros(3,1);
dydt(1) = sigma*(y(2) - y(1));
dydt(2) = q*y(1) - y(2) - y(1)*y(3);
dydt(3) = y(1)*y(2) - beta*y(3);
```

3.4 LIMITATIONS AND CHALLENGES

Despite the success with the Lorenz system, the implementation is not fail proof. Unfortunately, in fact, the code discussed above does not currently reproduce the Lorenz pictures that it once generated and which are displayed here. Hence, the m-file will not be made available with this thesis. This inadequacy may be due in part to the fact that the m-file may be dependent on the version of MatLab in which it was written (ver 6.5R13, student version). Even so, in the original computation, a start and stop hand tweeking process was necessary in order to generate as many rings of the lorenz manifold as possible. The length of integration time, for instance, along with other algorithm parameters all contribute to whether the code is able to find the next point in the mesh. Many angles have been examined in the troubleshooting process. However, since the development of the code was not the focal point of the dissertation, more research time was not dedicated to further troubleshooting.

There is also a problem that occurs after the algorithm finds a point, b, but the angle requirement repeatedly fails, no matter how small BigDelta was made. This might be a result of the initial circle not being close enough to the fixed point (i.e. delta was too large) and small inaccuracies grew into larger ones later on. In addition, the algorithm is not particularly fast, and as the mesh is grown, more and more mesh points are added so that it takes longer and longer to calculate subsequent circles. It can be especially time consuming, for example, when the 100^{th} mesh point of the next circle is being calculated and the angle accuracy condition fails. Then, `BigDelta` has to be shrunk and the algorithm has to start all over with point 1, in the calculation of the next circle.

The most difficult aspect of the implementation of this algorithm is choosing the next point, `y0`, on `Cr` from which to shoot in order to find the next point, `b`, in the mesh. If every trajectory intersected the half plane, this would be rather simple (depending on the accuracy and success of the event locator in finding the intersections!); however, as was shown in the code above, many trajectories simply "time out" on one side of the plane without ever crossing it. Then, it has to be determined where the trajectory landed at the end of the integration time. In some cases, it is necessary to find out where the trajectory was coming from in order to choose the next correct point. This decision tree was something not explicitly outlined in [63] and which proved to be the most challenging aspect of the design of the code.

Another issue found along the way included the manner in which data was stored and saved for future use. The multiband matrices that MatLab provides is a nice way to organize and keep track of data, and the `multiband` read and write functions make it easy to save this data in `.mat` files that can be loaded for later use. Also, finding an appropriate way to determine when a solution point along a trajectory met some requirement (e.g. was a certain distance from the point, `r`, or from the plane) was an especially difficult task. The first version of the Manifold code included a type of "manual" check at each time step along the solution to determine if an event occurred, which, not surprisingly, took an enormous amount of time, since very small time steps needed to be taken. It was obvious that something more sophisticated would be necessary. Then, the option of defining an event function for an event locator for the ode solvers was found. The event locator finds the time and corresponding solution of when/where specified events occurred with respect to the integrated system. Once the proper event functions were defined, this proved to be a huge improvement, both in accuracy and time. Even so, time spent "processing" the event locations that the event functions returns may be a source of inefficiency (See Lines 127-165.a-d in the previous section describing the `Manifold.m` code.) Furthermore, in some cases, no events occur, which means having to determine the next `y0` in another way.

We were able to use the algorithm for illustration in the above mentioned endotoxin tolerance paper and it was decided not to pursue any further troubleshooting of the algorithm as it would not benefit the direction of the thesis. However, there is currently no tool for public use for calculating and displaying two dimensional manifolds of three dimensional ODE systems, so future research could include making this a reality. It definitely remains an interesting problem to examine and some of the other algorithms mentioned in the first section, specifically [44], might prove to be more conducive to being programmed.

As mentioned previously, all the methods mentioned in the introductory paragraphs can theoretically be used to produce manifolds of degree greater than two. The difficulty is in the visualization of such methods and much research is ongoing in this area. In addition, Osinga in [87] demonstrates the visualization of an example of a 2-dimensional manifold of a 4-dimensional ODE system, projecting the manifold onto a plane in $\mathbb{R}^4$ and coloring the points on the manifold according to geodesic distance from the saddle point. Features of the particular system explored are taken advantage of to enhance the visualization and its interpretation. In general, such problems remain a challenge. Hence, there are many parts of this field that remain open for exploration.

4.0 MATHEMATICAL EXPLORATION OF TOLERANCE

4.1 INTRODUCTION

In Chapter 2 we analyzed biological phenomena known as endotoxin tolerance and potentiation via a small ODE model of the acute inflammatory response. An excitatory variable (N^*) could have a damped response (*tolerance*) when given an excitatory stimulus (P_E) prior to a challenge stimulus, compared to the response in the absence of a preconditioning input. On the other hand, the response with a preconditioner could instead be enhanced (*potentiation*). In the course of exploring this, it became evident that such behavior would be interesting to examine from a purely mathematical perspective since it deals with transient behavior in an ODE system, a topic not well covered in the dynamical systems literature. Most dynamical systems behaviors are studied in the asymptotic limit or around fixed points, where linearization can be employed.

The work presented in this chapter shows novel results that explore the behavior of transients under certain conditions. Related, perhaps, to this exposition is research on isochronicity, which deals with behaviors that occur within the same interval of time. [100, 41] For instance, Sabatini in [100] defines a critical point classified as a center to be *isochronous* if every nontrivial cycle within a neighborhood of the critical point has the same period. As Sabatini comments, the "intuitive idea of isochronicity is related to phenomena occurring at equal time intervals, so that one does not have to restrict to study the behaviour of solutions in the neighborhood of a critical point." Studying the behavior of solutions away from a critical point proves a necessary aspect of the work presented in this chapter. However, the research aiming to locate isochronous sections of autonomous differential systems deals mostly with systems which are oscillatory in nature [100, 41, 52] Hence, this body of work on isochronous sections was not used in the results of this chapter, but it could be a potential source for additional insight in the future.

In the sections that follow, the tolerance behavior is examined in both linear and nonlinear 2-dimensional ODE systems. The first section establishes mathematical definitions and notation for exploring tolerance (and potentiation). Then, theorems for general 2D ODE systems are presented. These extend the tolerance "window" by continuity arguments. Following these initial theorems are results showing sufficient or necessary conditions for tolerance to occur in very particular situations. Some of these results are used in later sections. Afterward, the general 2D linear case is explored. This case is complete, containing theorems that pinpoint exactly where tolerance will or will not occur in a system given an initial condition. Next, the general 2D nonlinear case is presented, which is a more difficult case than the linear one, since exact analytical solutions are not available, in general. However, there are statements that are made, via the concept of inhibition and the use of isoclines to narrow down the possibility of the presence of tolerance. Specific examples are used here to illustrate this approach. Finally, two purely numerical approaches for locating tolerance are shown, the first of which is not restricted to 2D systems.

4.2 PRELIMINARY DEFINITIONS

Consider the ODE system

$$\left.\begin{aligned}\dot{x} &= f(x,y)\\ \dot{y} &= g(x,y),\end{aligned}\right\} \qquad (4.1)$$

where $x, y \in \mathbb{R}$, f and g are locally Lipschitz.

(A1) Assume $(0,0)$ is a stable fixed point of (4.1), the eigenvalues of which are real and negative. (eliminates spirals and centers)

Let $\phi(t)$ be the solution to the initial value problem of (4.1) with initial value

$$\phi(0) = (x_0, y_0),\ x_0 > 0.$$

Similarly, let $\psi(t)$ be the solution to the initial value problem of (4.1) with initial value

$$\psi(0) = (\tilde{x}_0, \tilde{y}_0) \equiv \phi(s) + (x_0, 0) \text{ for some } 0 \leq s < \infty.$$

(A2) Assume $\phi(t)$ and $\psi(t)$ are nonnegative for all $t \geq 0$ and that both (x_0, y_0) and $(\tilde{x}_0, \tilde{y}_0)$ lie in the basin of attraction for $(0,0)$ in the first quadrant.

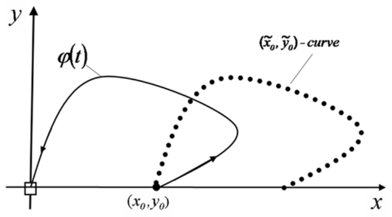


Figure 21: The original trajectory $\varphi(t)$ and the curve of all possible $(\tilde{x}_0, \tilde{y}_0)$ points, formed by shifting the graph of $\varphi(t)$ in the x-direction by an amount of x_0.

Note that $(\tilde{x}_0, \tilde{y}_0)$ is essentially a shifting of the point $\phi(s)$ in the x-direction by the amount, x_0. We refer to this as the 'shift' amount. Here and throughout, we assume that this shift amount is the value of x_0, the first component of the initial condition, $\phi(0)$. Thus, for different values of s ranging from 0 to ∞, a curve of possible $(\tilde{x}_0, \tilde{y}_0)$'s is formed as illustrated by Figure 21. Now, we define what it means for (4.1) to exhibit tolerance and potentiation for a given pair $< (x_0, y_0), s >$ and make a few other definitions and remarks that will be useful throughout this chapter:

Definition 1. *The system (4.1) is said to* exhibit tolerance for $< (x_0, y_0), s >$*, if there exists* $\tau > 0$ *such that* $\psi_1(\tau) < \phi_1(\tau)$*.*

Definition 2. *On the other hand, if* $\psi_1(t) \geq \phi_1(t)$ *for all* $t \in [0, \infty)$*, then (4.1) is said to* exhibit potentiation for $< (x_0, y_0), s >$ *or, equivalently, that (4.1)* does not exhibit tolerance for $< (x_0, y_0), s >$*. The latter will be the preferred terminology.*

Remark 1. *Note that for fixed* (x_0, y_0)*, every* $< (x_0, y_0), s >$ *defines a unique point,* $(\tilde{x}_0, \tilde{y}_0)$*. Thus, at times it might be stated that "*$(\tilde{x}_0, \tilde{y}_0)$ *produces (or does not produce) tolerance in (4.1)," which will be equivalent to saying that (4.1) exhibits (or does not exhibit) tolerance for* $< (x_0, y_0), s >$*.*

Definition 3. *For* $< (x_0, y_0), s >$*, for which tolerance is exhibited, define* s *as the* jump time*.*

Definition 4. *For* $\tau > 0$ *such that* $\psi_1(\tau) < \phi_1(\tau)$*, define* τ *as the* compare time*.*

Definition 5. *Define* $\phi(t)$ *as the* original solution or trajectory*.*

Definition 6. *Define* $\psi(t)$ *as the* competing solution or trajectory*.*

Remark 2. *The notation* $(x_0, y_0) \cdot t$ *is the image of the point* (x_0, y_0) *under the flow of (4.1) for time* t*. The set of points,* $\{(x_0, y_0) \cdot t | t \geq 0\}$*, is then the solution curve or trajectory of the initial value problem with initial value* (x_0, y_0)*. This set is also referred to as the graph of the solution.*

Remark 3. *Whenever the existence of tolerance is mentioned, it is with respect to the first component,* x*, of a system unless otherwise stated.*

4.3 THEOREMS EXTENDING THE WINDOW OF TOLERANCE

Definition 1 of tolerance includes only the presence of one time point, $\tau > 0$ where $\psi_1(\tau) < \phi_1(\tau)$. However, a continuity argument can extend this window, from a single time point to an open interval,(t_1, t_2), around the compare time, τ. Furthermore, it will be shown that at $t = t_1$, $\psi_1(t_1)$ is actually equal to $\phi_1(t_1)$ and this will lead to the result that the value of the vector field, in the (negative) x-direction, at $\psi_1(t_1)$, i.e. $f(\psi(t_1))$, will be *less than* the value of the vector field, in the (negative) x-direction, at $\phi_1(t_1)$, i.e. $f(\phi(t_1))$. This is the content of Proposition 1 below, which will be used in Corollaries 3 and 5, of Theorem 2 and Theorem 4, respectively and which will also be important in Section 4.6. Figure 22 illustrates Proposition 1 with time courses of relevant solutions.

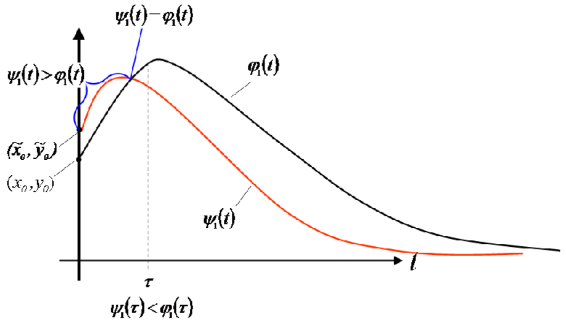


Figure 22: Time courses illustrating Proposition 1

Proposition 1. *Assume (A1) and (A2). If (4.1) exhibits tolerance for $< (x_0, y_0), s >$ in x, at $\tau > 0$, then there exists an open neighborhood (t_1, t_2) around τ such that $\psi_1(\hat{t}) < \phi_1(\hat{t})$ for every $\hat{t} \in (t_1, t_2)$ and $\psi_1(t_1) = \phi_1(t_1)$. Furthermore, $f(\psi(t_1)) \leq f(\phi(t_1))$.*

Proof. By the definition of $(\tilde{x}_0, \tilde{y}_0)$, the initial condition of $\psi(t)$,we know that $\tilde{x}_0 > x_0$ or, equivalently, $\psi_1(0) > \phi_1(0)$. Since (4.1) exhibits tolerance in x at $\tau > 0$, then $\psi_1(\tau) < \phi_1(\tau)$. Consider the set

$$\Gamma = \{t | \psi_1(t) = \phi_1(t), 0 < t < \tau\}.$$

Since $\phi_1(t)$ and $\psi_1(t)$ are continuous, by the intermediate value theorem, there exists $0 < \tau^* < \tau$ such that $\psi_1(\tau^*) = \phi_1(\tau^*)$. Hence, Γ is not empty. Since $(0, \tau)$ is a bounded interval, let $t^* = \sup_{t>0} \Gamma$. Since Γ is not empty, there exists an increasing sequence of time values, t_n, such that $t_n \to t^*$ as $n \to \infty$ and $t_n \in \Gamma$ for all n. Thus, by continuity of the solutions ψ_1 and ϕ_1, $\psi_1(t^*) = \phi_1(t^*)$. Since $t^* = \sup_{t>0} \Gamma$, we have that $0 < t^* \leq \tau$. However, we just established that $\psi_1(t^*) = \phi_1(t^*)$, yet earlier it was shown that $\psi_1(\tau) < \phi_1(\tau)$. Hence t^* is strictly less than τ: $t^* < \tau$. Since Γ is not empty, we may conclude that $t^* \in \Gamma$. Furthermore, using the continuity of the solutions ψ_1 and ϕ_1 again, we have that $\psi_1(\hat{t}) < \phi_1(\hat{t})$ for all $\hat{t} \in (t^*, \tau + \delta) \equiv (t_1, t_2)$ for some $\delta > 0$. Furthermore, since $\psi_1(\hat{t}) < \phi_1(\hat{t})$ for all $\hat{t} \in (t_1, t_2)$ and $\psi_1(t_1) = \phi_1(t_1)$, it can be concluded that $f(\psi(t_1)) \leq f(\phi(t_1))$. $\square$

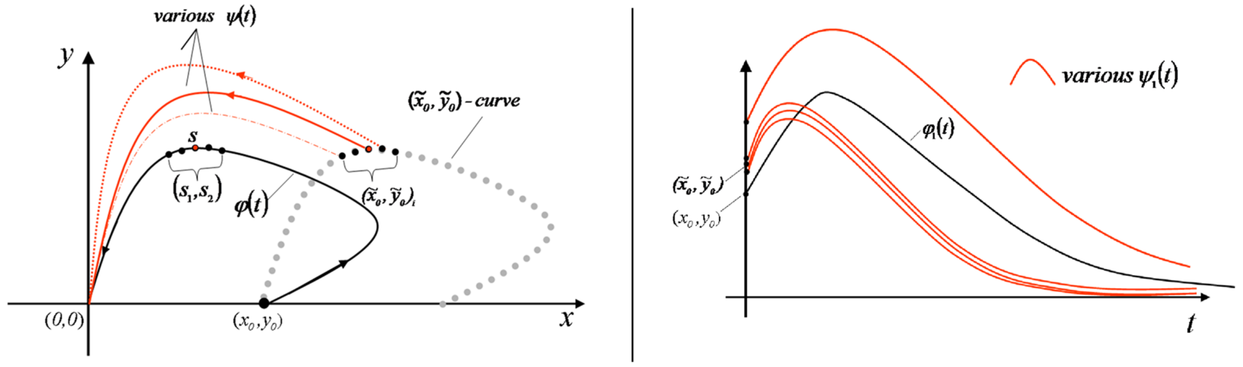


Figure 23: Left Panel: Illustration of Theorem 2 in the phase plane. Right Panel: Illustration of Theorem 2 with time courses of relevant solutions.

Proposition 1 above extends the window of tolerance with respect to the *compare time*, $\tau > 0$, where $\psi_1(\tau) < \phi_1(\tau)$. Theorem 2 below extends the window to tolerance with respect to the *jump time*, $s > 0$, where tolerance is known to exist for a particular $< (x_0, y_0), s >$. An open neighborhood, (s_1, s_2), around s is found such that there exists a compare time, $t_p > 0$, for every jump time, $s_p \in (s_1, s_2)$, where tolerance is exhibited for $< (x_0, y_0), s_p >$. Corollary 3 then extends the neighborhood around the particular compare time, t_p, associated with the jump time, s_p, by using Proposition 1. Figure 23 illustrates Proposition 2 in the phase plane (left panel) and with time courses of relevant solutions (right panel).

Theorem 2. *Assume (A1) and (A2). If (4.1) exhibits tolerance for $< (x_0, y_0), s >$ in x, then there exists an open interval, (s_1, s_2), around s such that (4.1) exhibits tolerance in x for $< (x_0, y_0), s_p >$ for all $s_p \in (s_1, s_2)$.*

Proof. Assume the set up as above with the two initial value problems involving $\phi(t)$ and $\psi(t)$ (in particular, assumptions (A1) and (A2)), and also consider the new IVP with solution $\hat{\psi}(t)$ to (4.1) with initial condition $\hat{\psi}(0) = (\phi_1(s_p), \phi_2(s_p)) + (x_0, 0) \equiv (\hat{x}_0, \hat{y}_0)$, where $0 < s_p = s + p$ for some $p \in R$. We wish to show that for $|p|$ sufficiently small there exists $t_p > 0$ such that $\hat{\psi}_1(t_p) < \phi_1(t_p)$.

By Proposition 1, since (4.1) exhibits tolerance for $< (x_0, y_0), s >$, there exists a neighborhood (t_1, t_2) such that $\psi_1(t) < \phi_1(t)$ for all $t \in (t_1, t_2)$. Since the RHS of (4.1) is locally Lipschitz (with Lipschitz constant L) and continuous in x and y, then given $\epsilon > 0$, there exists $\delta > 0$ such that if $||(\tilde{x}_0, \tilde{y}_0) - (\hat{x}_0, \hat{y}_0)|| < \delta$ then $||\psi(t) - \hat{\psi}(t)|| < \epsilon e^{Lt}$, on the interval for which ψ and $\hat{\psi}$ solve their respective initial value problems ($\psi, \hat{\psi}$ continuously depend on initial conditions). Thus, if $||(\tilde{x}_0, \tilde{y}_0) - (\hat{x}_0, \hat{y}_0)|| < \delta$ then we need $||\phi(s + p) - \phi(s)|| < \delta$. Since ϕ is continuous with respect to t, there exists $\delta_p > 0$ such that if $|p| < \delta_p$ then $||\phi(s + p) - \phi(s)|| < \delta$, as needed.

Therefore, if $\|(\tilde{x}_0, \tilde{y}_0) - (\hat{x}_0, \hat{y}_0)\| < \delta$, or equivalently $|p| < \delta_p$ then $|\psi_1(t) - \hat{\psi}_1(t)| < \epsilon e^{Lt}$ and for a particular t value, t_p, we can say that $|\psi_1(t_p) - \hat{\psi}_1(t_p)| < \epsilon e^{Lt_p} \equiv \hat{\epsilon}$. So, let $t_p \in (t_1, t_2)$, the interval in which $\psi_1(t) < \phi_1(t)$. Let ϵ be such that $\hat{\epsilon} = \frac{1}{2}(\phi_1(t_p) - \psi_1(t_p))$. Then, by the above, there exists $\delta_p > 0$ such that if $|p| < \delta_p$, then $|\psi_1(t_p) - \hat{\psi}_1(t_p)| < \hat{\epsilon} = \frac{1}{2}(\phi_1(t_p) - \psi_1(t_p))$. Hence,

$$
\begin{aligned}
\hat{\psi}_1(t_p) = \hat{\psi}_1(t_p) + \psi_1(t_p) - \psi_1(t_p) &< |\hat{\psi}_1(t_p) - \psi_1(t_p)| + \psi_1(t_p) \\
&= |\psi_1(t_p) - \hat{\psi}_1(t_p)| + \psi_1(t_p) \\
&< \frac{1}{2}(\phi_1(t_p) - \psi_1(t_p)) + \psi_1(t_p) \\
&= \hat{\epsilon} + \psi_1(t_p) \\
&= \hat{\epsilon} + \phi_1(t_p) - d, \text{ where } d = \phi_1(t_p) - \psi_1(t_p) > 0 \\
&= \frac{1}{2}d + \phi_1(t_p) - d \\
&< \phi_1(t_p)
\end{aligned}
$$

Therefore, there exists a $t-$value, namely t_p, where $\hat{\psi}_1(t_p) < \phi_1(t_p)$. (Also, we need to make sure that δ_p is small enough so that $(\hat{x}, \hat{y})$ is in the basin of attraction of $(0, 0)$. This is possible since (4.1) depends continuously on initial conditions.) Thus, let $(s_1, s_2) \equiv (s - \delta_p, s + \delta_p)$.

Then, for all $\delta_p \in (s_1, s_2)$ there exists $t_p > 0$ such that $\hat{\psi}_1(t_p) < \phi_1(t_p)$, completing the proof of Theorem 2. $\square$

Corollary 3. *With respect to the results of Theorem 2, there exists an open interval around t_p, (t_{p_1}, t_{p_2}), such that $\hat{\psi}_1(t) < \phi_1(t)$ for all $t \in (t_{p_1}, t_{p_2})$.*

Proof. This follows immediately by using Proposition 1 on $< (x_0, y_0), s_p >$. $\square$

Similarly to the previous statements, Theorem 4 below extends the window of tolerance given that (4.1) exhibits tolerance for a given $< (x_0, y_0), s >$. However, this theorem extends the window of tolerance around the initial condition, (x_0, y_0), of $\phi(t)$, where tolerance is know to exist for $< (x_0, y_0), s >$, rather than around the *jump time*, s, or the *compare time*, τ. An open ball, B_r, of radius, r, is found around (x_0, y_0), such that for any point, (x_p, y_p), in B_r, (4.1) exhibits tolerance for $< (x_p, y_p), s >$: i.e. there exits a *compare time*, $t_p > 0$ such that $\hat{\psi}_1(t_p) < \hat{\phi}_1(t_p)$, where $\hat{\phi}(0) = (x_p, y_p)$ and $\hat{\psi}(0) = \hat{\phi}(s) + (x_p, 0)$. Corollary 5 then extends the neighborhood around the particular compare time, t_p, associated with $< (x_p, y_p), s >$, by using Proposition 1.

Theorem 4. *Assume (A1) and (A2). If (4.1) exhibits tolerance for $< (x_0, y_0), s >$ in x, then there exists an open ball, B_r, of radius, r, around (x_0, y_0) such that if $(x_p, y_p) \in B_r((x_0, y_0))$ then (4.1) exhibits tolerance in x for $< (x_p, y_p), s >$.*

Proof. Assume the set up as above with the two initial value problems involving $\phi(t)$ and $\psi(t)$ (in particular, assumptions (A1) and (A2)), and also consider two new IVPs with solutions $\hat{\phi}(t)$ to (4.1) with initial condition $\hat{\phi}(0) = (x_0, y_0) + (p_1, p_2) \equiv (x_p, y_p)$ and $\hat{\psi}(t)$ to (4.1) with initial condition $\hat{\psi}(0) = (\hat{\phi}_1(s), \hat{\phi}_2(s)) + (x_p, 0) \equiv (\tilde{x}_p, \tilde{y}_p)$. We wish to show that there exists $t_p \geq 0$ such that $\hat{\psi}_1(t_p) < \hat{\phi}_1(t_p)$.

Given $\epsilon > 0$, since (4.1) is locally Lipschitz (with Lipschitz constant L) and continuous in x and y, there exists $\delta_p > 0$ such that if $||(\tilde{x}_0, \tilde{y}_0) - (\tilde{x}_p, \tilde{y}_p)|| < \delta$ then $||\psi(t) - \hat{\psi}(t)|| < \epsilon e^{Lt}$, on the interval for which ψ and $\hat{\psi}$ solve their respective initial value problems. (solutions of (4.1) continuously depend on initial conditions).

Thus, if $\|(\tilde{x}_0, \tilde{y}_0) - (\tilde{x}_p, \tilde{y}_p)\| < \delta$, then $|\psi_1(t) - \hat{\psi}_1(t)| < \epsilon e^{Lt}$ and for a particular t value, t_p, we can say that $|\psi_1(t_p) - \hat{\psi}_1(t_p)| < \epsilon e^{Lt_p} \equiv \hat{\epsilon}$. Fix $t_p > 0$ such that $\phi_1(t_p) > \psi_1(t_p)$.

Let ϵ be such that $\hat{\epsilon} = \frac{1}{2}(\phi_1(t_p) - \psi_1(t_p))$. Then, if $\|(\tilde{x}_0, \tilde{y}_0) - (\tilde{x}_p, \tilde{y}_p)\| < \delta$ we have that $|\psi_1(t_p) - \hat{\psi}_1(t_p)| < \hat{\epsilon} = \frac{1}{2}(\phi_1(t_p) - \psi_1(t_p))$. Then,

$$\begin{aligned}
\hat{\psi}_1(t_p) &= \hat{\psi}_1(t_p) - \psi_1(t_p) + \psi_1(t_p) \\
&\leq |\hat{\psi}_1(t_p) - \psi_1(t_p)| + \psi_1(t_p) \\
&< \hat{\epsilon} + \phi_1(t_p) - d, \text{ where } d = \phi_1(t_p) - \psi_1(t_p) > 0, \text{ since } \phi_1(t_p) > \psi_1(t_p) \\
&= \frac{1}{2}d + \phi_1(t_p) - d \\
&= \phi_1(t_p) - \frac{1}{2}d
\end{aligned}$$

So, since we have established that $\phi_1(t_p) - \frac{1}{2}d > \hat{\psi}_1(t_p)$ we have

$$\begin{aligned}
\hat{\phi}_1(t_p) - \hat{\psi}_1(t_p) &> \hat{\phi}_1(t_p) - (\phi_1(t_p) - \frac{1}{2}d) \\
&\geq -|\hat{\phi}_1(t_p) - \phi_1(t_p)| + \frac{1}{2}d
\end{aligned}$$

Since $\phi(t)$ depends continuously on initial conditions then given $\epsilon_p > 0$, there exists $\delta_p > 0$ such that if $\|(x_0, y_0) - (x_p, y_p)\| < \delta_p$ then $|\phi_1(t) - \hat{\phi}_1(t)| < \hat{\epsilon}_p$, where $\hat{\epsilon}_p = \epsilon_p e^{Lt_p}$. Thus, let $\epsilon_p > 0$ be such that $\hat{\epsilon}_p = \frac{1}{2}d$ so that if $\|(x_0, y_0) - (x_p, y_p)\| < \delta_p$, then $|\hat{\phi}_1(t_p) - \phi_1(t_p)| < \frac{1}{2}d$ and $-|\hat{\phi}_1(t_p) - \phi_1(t_p)| > -\frac{1}{2}d$. Hence, continuing from the last inequality, we have

$$\begin{aligned}
\hat{\phi}_1(t_p) - \hat{\psi}_1(t_p) &> -|\hat{\phi}_1(t_p) - \phi_1(t_p)| + \frac{1}{2}d \\
&> -\frac{1}{2}d + \frac{1}{2}d \\
&= 0.
\end{aligned}$$

Hence, $\hat{\phi}_1(t_p) > \hat{\psi}_1(t_p)$. Therefore, there exists a $t-$value, namely t_p, where $\hat{\psi}_1(t_p) < \hat{\phi}_1(t_p)$. Also, make sure that δ and δ_p are small enough to ensure (x_p, y_p) and $(\tilde{x}_p, \tilde{y}_p)$ lie in the basin of attraction for $(0, 0)$. Let $B_r((x_0, y_0)) \equiv B_{\delta_p}((x_0, y_0))$. Finally, note that for (x_0, y_0) on the x-axis, we only wish to consider those points that are both in $B_r((x_0, y_0))$ and in the first quadrant. Then, for every $(x_p, y_p) \in B_r((x_0, y_0)) \cap \mathbb{R}^{2+}$, there exists $t_p > 0$ such that $\hat{\psi}_1(t_p) < \hat{\phi}_1(t_p)$. $\square$

Corollary 5. *With respect to the results of Theorem 4, there exists an open neighborhood around (t_{p_1}, t_{p_2}), such that $\hat{\psi}_1(t) < \hat{\phi}_1(t)$ for all $t \in (t_{p_1}, t_{p_2})$.*

Proof. This follows immediately by using Proposition 1 on $< (x_p, y_p), s >$. $\square$

This concludes the continuity arguments for extending the window of tolerance in a system when there exists a $< (x_0, y_0), s >$ for which tolerance is known to occur. Next, we present a number of results that were observed during the initial stages of this mathematical tolerance research.

4.4 FIRST OBSERVATIONS

"We must start by admitting that almost nothing beyond general statements can be made about most nonlinear systems...and that any other tool in the workshop of applied mathematics...can and should be brought to bear on a specific problem." ~Guckenheimer and Holmes [43]

Indeed, truer words have not been written about the study of nonlinear systems and this "disclaimer" is adopted here and particularly in section 4.6. In fact, the problem we are seeking to elucidate is immune to the use of the powerful tool of linearization since what is being studied is a transient behavior and not one occurring particularly close to fixed points. The goal, as encouraged by Guckenheimer and Holmes, is to showcase every possible dynamical systems tool available in order to generate results about the existence of tolerance in general 2D nonlinear systems. The hope is that the unique nature of the problem and the creativity of the methods used to explore it will, not only be interesting, but insightful as well.

Overall, we are interested in necessary and sufficient conditions for tolerance to be exhibited in (4.1). This section covers several results about tolerance that were initially observed, and which are mostly for specific cases. Some of these results, however, are used in subsequent sections that show more general results. Recall the following assumptions:

(A1) $(0,0)$ is a stable fixed point of (4.1), the eigenvalues of which are real and negative. (eliminates spirals and centers)

(A2) $\phi(t)$ and $\psi(t)$ are nonnegative for all $t \geq 0$ and both (x_0, y_0) and $(\tilde{x}_0, \tilde{y}_0)$ lie in the basin of attraction for $(0,0)$ in the first quadrant.

The next proposition is specific to general 1-dimensional ODE systems and shows that tolerance cannot be exhibited for any $< (x_0, y_0), s >$ in a 1D system.

Proposition 6. *Assume (A1) and (A2) for a* 1*-dimensional system* $\dot{x} = f(x)$, $x \in \mathbb{R}$*:* 0 *is a locally stable fixed point of* $\dot{x} = f(x)$ *and* x_0 *and* $\tilde{x}_0$ *lie in the basin of attraction for* 0 *on the real line. Assume also that* f *is locally Lipschitz on* $[0, \infty)$*. Then,* $\dot{x} = f(x)$ *cannot exhibit tolerance.*

Proof. Let $\phi(t), t \geq 0$, be the solution to the initial value problem

$$\begin{aligned}\dot{x} &= f(x)\\ \phi(0) &= x_0 > 0.\end{aligned}$$

Let $\psi(t), t \geq 0$, be the solution to the initial value problem

$$\begin{aligned}\dot{x} &= f(x)\\ \psi(0) &= \tilde{x}_0 \equiv \phi(s) + x_0 \text{ for some } 0 \leq s < \infty.\end{aligned}$$

Since x_0 and $\tilde{x}_0$ are in the basin of attraction of 0, then both $\phi(t)$ and $\psi(t) \to 0$ monotonically for all $t \geq 0$. Since $\phi(t) \to 0$ monotonically for all $t \geq 0$, $\phi(T) < \phi(t)$ for every $T > t$. Further, since $\tilde{x}_0 > x_0$ by definition and $\psi(t) \to 0$ monotonically for all $t \geq 0$, there exists $t^* > 0$ such that $\psi(t^*) = \phi(0) = x_0$ and $\psi(t) > \phi(t)$ for all $t \leq t^*$. Hence, for all $\delta > 0, \phi(\delta) = \psi(t^* + \delta)$. Since $t^* + \delta > \delta$ and $\phi(t) \to 0$ monotonically, $\phi(t^* + \delta) < \phi(\delta) = \psi(t^* + \delta)$. This, along with the result that $\psi(t) > \phi(t)$ for all $t \leq t^*$, implies $\psi(t) > \phi(t)$ for all $t \in [0, \infty)$. Therefore, tolerance cannot be exhibited in $\dot{x} = f(x)$ where $x \in \mathbb{R}$. Instead $\dot{x} = f(x)$ exhibits potentiation for every $< x_0, s >$. □

The following proposition expands the results of Proposition 6 for solutions $\phi(t)$ and $\psi(t)$ of 2D systems which converge monotonically in the 1^{st} component to $(0, 0)$ and which are subsets of the same solution curve. Figure 24 illustrates this situation. This result will be used later in Section 4.6.1.

Proposition 7. *Assume (A1) and (A2). Assume $\phi_1(t)$ and $\psi_1(t) \to 0$ monotonically for all $t \geq 0$. Given $< (x_0, y_0), s >$, if there exists $\hat{t} > 0$ such that $\phi(-\hat{t}) = \phi(s) + (x_0, 0) \equiv \psi(0) \equiv (\tilde{x}_0, \tilde{y}_0)$, then $(\tilde{x}_0, \tilde{y}_0)$ does not produce tolerance in (4.1).*

Proof. For a given $< (x_0, y_0), s >$, let $\hat{t} > 0$ such that $\phi(-\hat{t}) = \psi(0) \equiv (\tilde{x}_0, \tilde{y}_0)$. This implies that $\phi(t)$ and $\psi(t)$ are both subsets of the same larger solution curve of the vector field defined by (4.1). Therefore, assuming $\phi_1(t)$ and $\psi_1(t) \to 0$ monotonically for all $t \geq 0$ and knowing, by definition, that $\psi_1(0) > \phi_1(0)$, the same arguments made in Proposition 6 can be made for $\phi_1(t)$ and $\psi_1(t)$, resulting in the fact that $\psi_1(t) > \phi_1(t)$ for all $t \in [0, \infty)$.

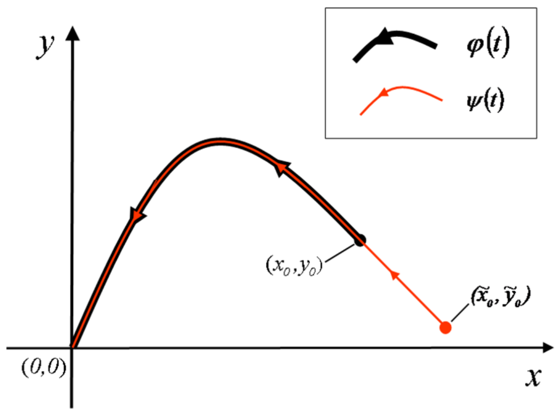


Figure 24: Illustration of the case when $\phi(t)$ and $\psi(t)$ converge monotonically in the 1st component and lie on the same solution curve. $\phi(t)$ (black) and $\psi(t)$ (orange) are actually separate trajectories being compared even though they belong to the same solution curve $\phi(t) \cup \psi(t)$. Note that $\phi_1(0) < \psi_1(0)$ and that at some time point $\tau > 0$, $\psi_1(\tau) = \phi_1(0)$. Also, note that both $\phi(t)$ and $\psi(t)$ are monotonically decreasing in the x-direction.

Hence, given $< (x_0, y_0), s >$, if $\phi_1(t)$ and $\psi_1(t) \to 0$ monotonically and there exists $\hat{t} > 0$ such that $\phi(-\hat{t}) = \psi(0) \equiv (\tilde{x}_0, \tilde{y}_0)$, then $(\tilde{x}_0, \tilde{y}_0)$ does not produce tolerance in (4.1). $\square$

The next proposition is a formal statement describing the "rescue" tolerance scenario presented in Chapter 2. A non-preconditioned trajectory (i.e. $\phi(t)$) does not decay toward $(0,0)$ but instead flows toward a higher fixed point, yet the preconditioned trajectory (i.e. $\psi(t)$) decays to $(0,0)$. Hence, there exists a compare time, $\tau > 0$, where tolerance occurs. (See Figs. 6a and 6b.) In other words, even though (x_0, y_0) is not in the basin of attraction for $(0,0)$, it is possible that the bumped point, $(\tilde{x}_0, \tilde{y}_0)$, is. The proof of proposition 8 can be generalized to n-dimensional systems, since it only examines the one component of an ODE system (e.g. the x-component in our current setup.)

Proposition 8 (Sufficiency). *Assume (A1) and that ϕ and ψ are nonnegative for all $t \geq 0$. If $\phi_1(t) \to p > 0$ as $t \to \infty$ (i.e. (x_0, y_0) is not in the basin of attraction for $(0,0)$), yet $\psi_1(t) \to 0$ as $t \to \infty$, then (4.1) exhibits tolerance for $< (x_0, y_0), s >$.*

Proof. Since $\psi_1(t) \to 0$ as $t \to \infty$, $\psi_1(t)$ is continuous, and $p > 0$, there exists, by continuity, a $\tau > 0$ such that $\psi_1(\tau) < p$. Thus, $\psi_1(t) < \phi_1(t)$ for all $t \geq \tau$. $\square$

Proposition 9 below addresses a very specific case when the initial condition, (x_0, y_0), of the original trajectory, $\phi(t)$, begins on an invariant x-axis. It is shown that such a case is equivalent to the case of a 1-dimensional ODE system, addressed above in Proposition 6.

Proposition 9. *Assume (A1) and (A2). If* $(x_0, y_0) = (x_0, 0)$, *(i.e.* (x_0, y_0) *is on the* x*-axis away from* $(0, 0)$*) and* $g(x, 0) = 0$ *in (4.1) for all* $x > 0$. *(i.e. the* x*-axis is invariant), then (4.1) cannot exhibit tolerance for* $< (x_0, y_0), s >$, $s \geq 0$.

Proof. This statement corresponds with Proposition 6 which states that tolerance cannot occur in a 1-D ODE. Let $(x_0, y_0) = (x_0, 0)$. Since (x_0, y_0) is on the x-axis (away from $(0, 0)$) and $g(x, 0) = 0$ in (4.1) for all $x > 0$, the solution $\phi(t) = \{(x_0, 0) \cdot t, |t \geq 0\}$ will remain on the x-axis for all $t \geq 0$ as it approaches $(0, 0)$ as $t \to \infty$. Similarly, $\psi(t)$ will also remain on the x-axis for all $t \geq 0$. Hence, the problem reduces to the 1-D case and thus, tolerance cannot be exhibited for any $< (x_0, 0), s >$ when the x-axis is invariant. □

Therefore, we list another assumption:

(A3) If $(x_0, y_0) \equiv (x_0, 0)$, assume that $g(x, 0) \neq 0$ for all $(x, 0)$, $x > 0$.

The next two propositions highlight specific conditions that are sufficient for tolerance to exist. The first of these propositions is for the case when the original trajectory, $\phi(t)$, is assumed to have its initial condition, $(\tilde{x}_0, \tilde{y}_0)$, on the x-axis. Assumption (A3) above is assumed for this proposition, so that $\phi(t)$ does not remain on the x-axis for all $t > 0$. These assumptions, along with assumptions (A1) and (A2) given on page 120, imply that, in this case, the graph of ϕ forms a closed region, S, with the x-axis. S is defined to not include the actual graph of ϕ. See the left panel of Figure 25.

Proposition 10 below considers this case, when the initial condition, $(\tilde{x}_0, \tilde{y}_0)$, of the original trajectory, $\psi(t)$, lies in the region, S. This case is possible given the way the $(\tilde{x}_0, \tilde{y}_0)$-curve is defined, if graph of ϕ is of a certain shape. (See, for example, the left panel of Figure 25.) If $(\tilde{x}_0, \tilde{y}_0)$ meets this condition in this special case, then $(\tilde{x}_0, \tilde{y}_0)$ will produce tolerance in (4.1) with respect to $< (x_0, 0), s >$.

Proposition 10. *Let* $(x_0, y_0) \equiv (x_0, 0)$ *and assume (A1), (A2), and (A3). Define* $(\tilde{x}_0, \tilde{y}_0) \equiv \phi(s) + (x_0, 0)$, *for some* $s > 0$ *and define the region,* S, *as the bounded set bounded by the*

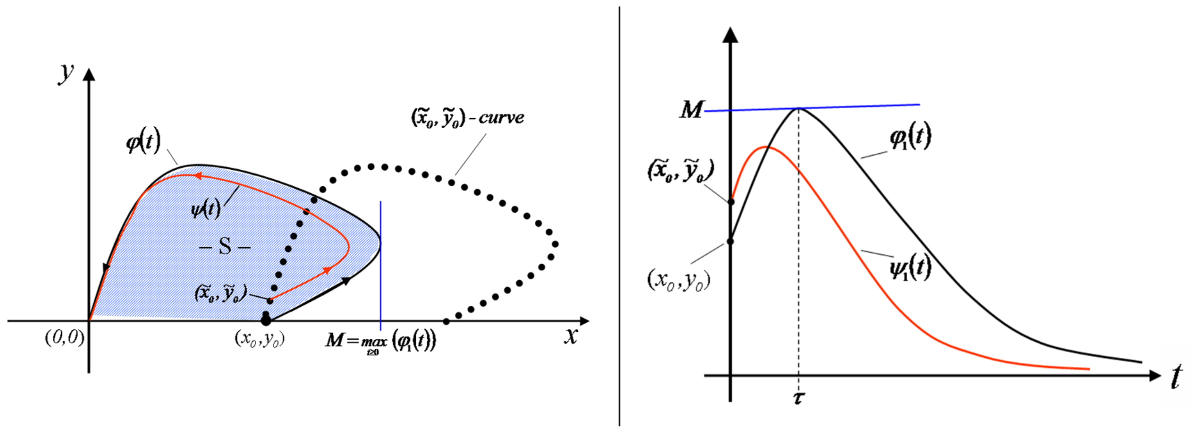


Figure 25: Figure illustrating Proposition 10. Left Panel: An example original trajectory, $\phi(t)$, with intial condition, $(x_0, 0)$, shown along with the $(\tilde{x}_0, \tilde{y}_0)$-curve which intersects the region, S, shown in light blue. An example of a competing trajectory, $\psi(t)$, with initial condition$(\tilde{x}_0, \tilde{y}_0) \in S$, is shown in red. Also, note that the maximum value in the x-direction for $\phi(t)$ is marked with a vertical blue line and denoted, $M \equiv \max_{t \geq 0} \phi_1(t)$. Right Panel: The time courses of both $\phi_1(t)$ (black) and $\psi_1(t)$ (red). $M \equiv \max_{t \geq 0} \phi_1(t)$ is labeled with a horizontal blue line, showing that $\psi_1(t)$ is bounded above by M. The time point, τ, marks the time where $\phi_1(t) = M$, showing that at this time, $\psi_1(t) < M$ and hence, less than $\phi_1(t)$.

x-axis and the graph of ϕ, but not including the graph of ϕ. S is bounded since, (1) $\phi(t) \geq 0$, for all $t \geq 0$, (2) $\phi(t) \to (0,0)$ as $t \to \infty$, and (3) $(x_0, y_0) \equiv (x_0, 0)$. If $(\tilde{x}_0, \tilde{y}_0) \in S$, then (4.1) will exhibit tolerance for $< (x_0, 0), s >$. (An example of a possible region, S, is shown as the light blue area in the left panel Figure 25.)

Proof. Let $(\tilde{x}_0, \tilde{y}_0) \in S$. From assumption (A2), $\phi(t)$ and $\psi(t)$ are nonnegative for all $t \geq 0$. Then, since trajectories cannot cross because of uniqueness of solutions, $\psi(t) = \{(\tilde{x}_0, \tilde{y}_0) \cdot t | t \geq 0\}$ is contained in the bounded set S. Let $M = \max_{t \geq 0} \{\phi_1(t)\}$ which exists because $\phi_1(0) = x_0$, $\phi_1(t) \to 0$ as $t \to \infty$ and ϕ is continuous. Then, because $\psi(t)$ is contained in S and $\phi(t)$ cannot intersect with $\psi(t)$, $\psi_1(t) < M$ for all $t \geq 0$, i.e. $\psi_1(t)$ is bounded above by M. Thus, if we let $\tau > 0$ be the time when $\phi_1(\tau) = M$ then $\phi_1(\tau) > \psi_1(\tau)$. Hence, for every s such that $(\tilde{x}_0, \tilde{y}_0) \in S$, (4.1) exhibits tolerance for $< (x_0, y_0), s >$. $\square$

The right panel of Figure 25 shows the time courses of both $\phi_1(t)$ and $\psi_1(t)$. This illustrates the following: Since the solution $\psi_1(t)$ is bounded above by the maximum value, M, of $\phi_1(t)$, occurring at time, τ, then the value of $\psi_1(t)$ at τ will be less than M and hence, less than $\phi_1(t)$. Thus, at τ, $\psi_1(\tau) < \phi_1(\tau)$. Notice that there is a value $t^* < \tau$ (not labeled) where $\phi_1(t^*) = \psi_1(t^*)$ and for which $\psi_1(t) < \phi_1(t)$ for all $t > t^*$. This t^* value is the time where $\phi_1(t^*)$ (the black curve) intersects $\psi_1(t^*)$ (the red curve) in the right panel of Figure 25.

The second proposition is similar to Proposition 10, but now $(\tilde{x}_0, \tilde{y}_0)$ is actually a point *on* the graph of ϕ, other than (x_0, y_0) itself. In the left panel of Figure 25, an intersection such as this can be seen. Since the $(\tilde{x}_0, \tilde{y}_0)$-curve is not a solution curve, this intersection does not violate any uniqueness properties of an ODE system that has a locally Lipschitz right hand side, such as (4.1). First, a few notational conventions are made.

Formally, define the $(\tilde{x}_0, \tilde{y}_0)$-curve as the set

$$P = \{(\tilde{x}_0, \tilde{y}_0) | (\tilde{x}_0, \tilde{y}_0) = \phi(s) + (x_0, 0), s \geq 0\},$$

and note that the graph of ϕ is the set of points

$$graph(\phi) = \{(x, y) | (x_0, y_0) \cdot t, t \geq 0\}.$$

If $(P \cap graph(\phi)) \neq \emptyset$, this implies that the solution $\phi(t)$ intersects the $(\tilde{x}_0, \tilde{y}_0)$-curve at some time, τ. Now, for the proposition:

Proposition 11. *Assume (A1), (A2), and (A3). Let*

$$\psi(0) = (\tilde{x}_0, \tilde{y}_0) \in P \cap graph(\phi) \backslash (x_0, y_0).$$

Then, $(\tilde{x}_0, \tilde{y}_0)$ will produce tolerance in (4.1) with respect to $< (x_0, y_0), s >$.

Proof. Assume (A1), (A2), and (A3) and that $P \cap graph(\phi) \backslash (x_0, y_0) \neq \emptyset$. Let $\psi(0) = (\tilde{x}_0, \tilde{y}_0) \in P \cap graph(\phi) \backslash (x_0, y_0)$. This implies that

$$\begin{aligned} \psi(0) = (\tilde{x}_0, \tilde{y}_0) &= \phi(s) + (x_0, 0) \\ &= \phi(\tau), \end{aligned}$$

for some $s, \tau > 0$, by definition of $(\tilde{x}_0, \tilde{y}_0)$ and the fact that $(\tilde{x}_0, \tilde{y}_0) \in P \cap graph(\phi) \backslash (x_0, y_0) \neq \emptyset$. Let $M = \max\limits_{t \geq 0} \{\phi_1(t)\}$ which exists because $\phi_1(0) = x_0$, $\phi_1(t) \to 0$ as $t \to \infty$ and ϕ is continuous. Then, there exists t_M such that $\phi_1(t_M) = M$ and $\phi_1(t) < M$ for all $t > t_M$. Thus,

$$\begin{array}{rcl} \phi_1(\tau + t_M) & < & \phi_1(t_M) \\ & \text{and} & \\ \phi_1(\tau + t_M) & = & \psi_1(t_M) \\ \text{Hence, } \psi_1(t_M) & < & \phi_1(t_M). \end{array}$$

In conclusion, (4.1) exhibits tolerance for $< (x_0, y_0), s >$ at the compare time, t_M. □

Now we present a proposition that gives a condition which can be checked when $(x_0, y_0) = (x_0, 0)$ and $\dot{x} > 0$ at (x_0, y_0), to determine if the $(\tilde{x}_0, \tilde{y}_0)$-curve and $\phi(t)$ intersect other than at, possibly, (x_0, y_0). First, we give some notation. Let

$$< x_f, x_g > = < f(x_0, y_0), g(x_0, y_0) >$$

be the vector at (x_0, y_0) defined by the vector field (4.1). For $(0, 0)$ (a stable fixed point of (4.1), denote the slow/weak eigenvector of $(0, 0)$ as $v = < v_1, v_2 >$. Recall the formal set definitions for the $(\tilde{x}_0, \tilde{y}_0)$-curve and the graph of ϕ, given, respectively, by the following:

$$\begin{array}{rcl} P & = & \{(\tilde{x}_0, \tilde{y}_0) | (\tilde{x}_0, \tilde{y}_0) = \phi(s) + (x_0, 0), s \geq 0\}, \text{ and} \\ graph(\phi) & = & \{(x, y) | (x_0, y_0) \cdot t, t \geq 0\}. \end{array}$$

Proposition 12. *Assume (A1), (A2), (A3) and that $\phi(0) = (x_0, y_0) = (x_0, 0)$. Also, assume that $\dot{x} > 0$ at (x_0, y_0). Using the notation given above, if*

$$x_f v_2 > v_1 x_g \tag{4.2}$$

then there exists an $s > 0$ such that $(\tilde{x}_0, \tilde{y}_0) \equiv (x_0, 0) + \phi(s)$ will produce tolerance for (4.1) with respect to $< (x_0, y_0), s >$.

Proof. Assume (A1), (A2), (A3) and that $\phi(0) = (x_0, y_0) = (x_0, 0)$. Also, assume that $\dot{x} > 0$ at (x_0, y_0). Arrange for the slow/weak eigenvector, v, to be such that: $v_1 \geq 0$. Then, by (A1) given on page 120, $v_2 > 0$. Also, from (A2), $\phi(t)$ is nonnegative for all $t \geq 0$ and $\phi(t) \to (0,0)$ as $t \to \infty$. Thus, $\phi(t)$ approaches $(0,0)$ along the slow/weak eigenvector, v. Furthermore, near zero, $\phi(t)$ has a slope approaching $< v_1, v_2 >$. We know that $x_f > 0$ because $\dot{x} > 0$ at (x_0, y_0), and since $\phi(t)$ is nonnegative for all $t \geq 0$, we know that $x_g > 0$ at (x_0, y_0). Since P is simply a translation of the graph of ϕ in the x-direction by an amount of x_0, if the slope of $\phi(t)$ at/near $(0,0)$ is steeper (greater) than the slope of $\phi(t)$ at (x_0, y_0), then P will intersect the graph of $\phi(t)$ in at least one point other than at (x_0, y_0). In other words, if $\frac{v_2}{v_1} > \frac{x_g}{x_f}$ or equivalently, $x_f v_2 > v_1 x_g$, there will exist $(\tilde{x}_0, \tilde{y}_0)$ such that $(\tilde{x}_0, \tilde{y}_0) \in P \cap graph(\phi) \backslash (x_0, y_0)$. This relationship between $< v_1, v_2 >$ and $< x_f, x_g >$ is Condition 4.2 stated above. Proposition 11 can then be used to conclude that there exists an $s > 0$ such that $(\tilde{x}_0, \tilde{y}_0) \equiv (x_0, 0) + \phi(s)$ will produce tolerance for (4.1) with respect to $< (x_0, y_0), s >$. $\square$

This concludes the First Observations section and we now consider general two dimensional linear ODE systems.

4.5 TOLERANCE IN GENERAL 2D LINEAR ODE SYSTEMS

4.5.1 Setup

In this section, we specifically consider 2D *linear* ODE systems and arrive at necessary and sufficient conditions for the existence of tolerance. Consider the linear system

$$\dot{x} = Ax, \tag{4.3}$$

where $A \in M^{2x2}$, $x \in \mathbb{R}^{2+} = [0, \infty) \times [0, \infty)$. Throughout this section, we will assume as before that:

(A1) $(0, 0)$ is a stable fixed point of (4.3), the eigenvalues of which are real and negative. (eliminates spirals and centers)

(A2) $\phi(t)$ and $\psi(t)$ are nonnegative for all $t \geq 0$ and both (x_0, y_0) and $(\tilde{x}_0, \tilde{y}_0)$ lie in the basin of attraction for $(0, 0)$ in the first quadrant.

Let λ_1 and λ_2 be the real, negative eigenvalues of A. To arrive at necessary and sufficient conditions for the existence of tolerance, there are three cases that must be considered regarding the eigenvalues and eigenvectors of A. The first two cases are when the two eigenvalues are identical: $\lambda_1 = \lambda_2 = \lambda < 0$. The second case considers distinct eigenvalues: $\lambda_1 \neq \lambda_2$. Within these cases some have subcases as well. Recall that by definition $\tilde{x}_0 \geq x_0$, a condition that will be used in the cases below. Now we address the first case:

4.5.2 Case 1:

$\lambda_1 = \lambda_2 = \lambda < 0$ and λ has two linearly independent eigenvectors

For this case, λ is an eigenvalue of A with multiplicity two, for which two linearly independent eigenvectors can be found. Let v and w be linear independent eigenvectors of λ. Then, any initial condition can be uniquely written as a linear combination of v and w. For the initial condition (x_0, y_0), we may write $(x_0, y_0) = c_1 v + c_2 w = (c_1 v_1 + c_2 w_1, c_1 v_2 + c_2 w_2)$, with $c_1, c_2 \in \mathbb{R}$. Thus, the solution, $\phi(t)$, to the IVP $\dot{x} = Ax$, $\phi(0) = (x_0, y_0)$ is

$$\phi(t) = c_1 v e^{\lambda t} + c_2 w e^{\lambda t} = (c_1 v + c_2 w)e^{\lambda t} = (c_1 v_1 + c_2 w_1, c_1 v_2 + c_2 w_2)e^{\lambda t} = (x_0, y_0)e^{\lambda t} \tag{4.4}$$

Similarly, consider the initial condition $(\tilde{x}_0, \tilde{y}_0)$ defined to be $(\tilde{x}_0, \tilde{y}_0) \equiv (x_0, 0) + \phi(s)$ for some $s \geq 0$. We may also uniquely write $(\tilde{x}_0, \tilde{y}_0) = d_1 v + d_2 w = (d_1 v_1 + d_2 w_1, d_1 v_2 + d_2 w_2)$, with $d_1, d_2 \in \mathbb{R}$. The solution $\psi(t)$ to the IVP $\dot{x} = Ax$, $\psi(0) = (\tilde{x}_0, \tilde{y}_0)$ is

$$\psi(t) = d_1 v e^{\lambda t} + d_2 w e^{\lambda t} = (d_1 v + d_2 w)e^{\lambda t} = (d_1 v_1 + d_2 w_1, d_1 v_2 + d_2 w_2)e^{\lambda t} = (\tilde{x}_0, \tilde{y}_0)e^{\lambda t}. \quad (4.5)$$

Furthermore, since we know that $\tilde{x}_0 \geq x_0$, we have that

$$d_1 v_1 + d_2 w_1 \geq c_1 v_1 + c_2 w_1. \quad (4.6)$$

Consider the difference between $\phi_1(t)$ and $\psi_1(t)$. Using equations (4.4) and (4.5),we have the following:

$$\begin{aligned} \phi_1(t) - \psi_1(t) &= c_1 v_1 e^{\lambda t} + c_2 w_1 e^{\lambda t} - (d_1 v_1 e^{\lambda t} + d_2 w_1 e^{\lambda t}) \\ &= (c_1 v_1 + c_2 w_1 - d_1 v_1 - d_2 w_1)e^{\lambda t} \\ &\leq 0, \end{aligned}$$

by (4.6) and the fact that $e^{\lambda t} > 0$ for all $t \geq 0$. Thus, $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$, and the following has been shown:

Theorem 13. *Assume (A1), (A2), and that $\lambda_1 = \lambda_2 = \lambda < 0$. Given $< (x_0, y_0), s >$ and hence,$(\tilde{x}_0, \tilde{y}_0)$, if λ has two linearly independent eigenvectors, then $\dot{x} = Ax$ cannot exhibit tolerance for $< (x_0, y_0), s >$. (i.e. $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$.)*

4.5.3 Case 2:

$$\lambda_1 = \lambda_2 = \lambda < 0 \text{ and } \lambda \text{ has only } \textit{one} \text{ eigenvector}$$

In this case, λ is an eigenvalue of A with multiplicity two, for which only one eigenvector (up to a scalar multiple) can be found. Let v be an eigenvector of λ. We wish to find the fundamental solution set. One solution to (4.3) is $x^{(1)}(t) = ve^{\lambda t}$. In order to obtain a second solution to form a generalized solution, it is necessary to find a generalized eigenvector that is linearly independent of v. Let $\bar{v}$ be the generalized eigenvector with respect to v, which is found by solving the following equation:

$$(A - \lambda I)\bar{v} = v,$$

where I is the 2x2 identity matrix. Then, a second solution to (4.3) is $x^{(2)}(t) = vte^{\lambda t} + \bar{v}e^{\lambda t}$. The intial condition, (x_0, y_0), can be uniquely written as a linear combination of v and $\bar{v}$:

$$(x_0, y_0) = c_1 v + c_2 \bar{v} = (c_1 v_1 + c_2 \bar{v}_1, c_1 v_2 + c_2 \bar{v}_2), \text{with } c_1, c_2 \in \mathbb{R}.$$

Then, the solution $\phi(t)$ to the IVP $\dot{x} = Ax$, $\phi(0) = (x_0, y_0)$ is

$$\begin{aligned} \phi(t) &= c_1 ve^{\lambda t} + c_2(vte^{\lambda t} + \bar{v}e^{\lambda t}) \\ &= (c_1 v_1 e^{\lambda t} + c_2(v_1 te^{\lambda t} + \bar{v}_1 e^{\lambda t}), c_1 v_2 e^{\lambda t} + c_2(v_2 te^{\lambda t} + \bar{v}_2 e^{\lambda t})). \end{aligned} \tag{4.7}$$

Similiary, the initial condition, $(\tilde{x}_0, \tilde{y}_0)$, can be uniquely written as a linear combination of v and $\bar{v}$:

$$(\tilde{x}_0, \tilde{y}_0) = d_1 v + d_2 \bar{v} = (d_1 v_1 + d_2 \bar{v}_1, d_1 v_2 + d_2 \bar{v}_2), \text{with } d_1, d_2 \in \mathbb{R},$$

and the solution $\psi(t)$ to the IVP $\dot{x} = Ax$, $\psi(0) = (\tilde{x}_0, \tilde{y}_0)$ is

$$\begin{aligned} \psi(t) &= d_1 ve^{\lambda t} + d_2(vte^{\lambda t} + \bar{v}e^{\lambda t}) \\ &= (d_1 v_1 e^{\lambda t} + d_2(v_1 te^{\lambda t} + \bar{v}_1 e^{\lambda t}), d_1 v_2 e^{\lambda t} + d_2(v_2 te^{\lambda t} + \bar{v}_2 e^{\lambda t})). \end{aligned} \tag{4.8}$$

Furthermore, since we know that $\tilde{x}_0 \geq x_0$, we have that

$$d_1 v_1 + d_2 \bar{v}_1 \geq c_1 v_1 + c_2 \bar{v}_1. \tag{4.9}$$

In the following calculations, we will be interested in the first components of v and $\bar{v}$. For Case 2, there are two subcases that need to be considered. The first subcase deals with the possibility that the first component of v is zero, implying that the first component of $\bar{v}$ is nonzero. Otherwise, v and $\bar{v}$ would not be linearly independent. Since is $\bar{v}$ nonzero, it can then be arranged for its first component to be 1. The second subcase is for when the first component of v is nonzero. Then, designating the first component of $\bar{v}$ as the free parameter, we can choose it to be zero and arrange for $\bar{v}_2 > 0$. Thus, we consider two subcases: (a) $v_1 = 0$ and $\bar{v}_1 = 1$ (b) $v_1 = 1$ and $\bar{v}_1 = 0$.

4.5.3.1 Case 2a:

$$v_1 = 0 \text{ and } \bar{v}_1 = 1$$

For this first subcase of Case 2, (4.9) becomes

$$d_2 \geq c_2. \tag{4.10}$$

Consider the difference between $\phi_1(t)$ and $\psi_1(t)$. Using equations (4.7) and (4.8) and that in this case $v_1 = 0$ and $\bar{v}_1 = 1$, we have the following:

$$\begin{aligned}
\phi_1(t) - \psi_1(t) &= c_1 v_1 e^{\lambda t} + c_2(v_1 t e^{\lambda t} + \bar{v}_1 e^{\lambda t}) - (d_1 v_1 e^{\lambda t} + d_2(v_1 t e^{\lambda t} + \bar{v}_1 e^{\lambda t})) \\
&= c_2 e^{\lambda t} - d_2 e^{\lambda t} \\
&= (c_2 - d_2) e^{\lambda t} \\
&\leq 0,
\end{aligned}$$

by (4.10) and the fact that $e^{\lambda t} > 0$ for all $t \geq 0$. Thus, $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$. Therefore, the following has been shown:

Theorem 14. *Let $< (x_0, y_0), s >$ and, hence, $(\tilde{x}_0, \tilde{y}_0)$, be given. Assume (A1), (A2), that $\lambda_1 = \lambda_2 = \lambda < 0$ and that λ has only one eigenvector. Let v be an eigenvector of λ and let $\bar{v}$ be the generalized eigenvector, linearly independent of v, found by solving the equation $(A - \lambda I)\bar{v} = v$, where I is the 2x2 identity matrix . If $v_1 = 0$ and $\bar{v}_1 = 1$, then $\dot{x} = Ax$ cannot exhibit tolerance for $< (x_0, y_0), s >$. (i.e. $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$.)*

4.5.3.2 Case 2b:

$$v_1 = 1 \text{ and } \bar{v}_1 = 0$$

In this second subcase of Case 2, (4.9) becomes

$$d_1 \geq c_1. \tag{4.11}$$

Consider the difference between $\phi_1(t)$ and $\psi_1(t)$. Using equations (4.7) and (4.8) and that in this case $v_1 = 1$ and $\bar{v}_1 = 0$, we have the following:

$$\begin{aligned}
\phi_1(t) - \psi_1(t) &= c_1 v_1 e^{\lambda t} + c_2(v_1 t e^{\lambda t} + \bar{v}_1 e^{\lambda t}) - (d_1 v_1 e^{\lambda t} + d_2(v_1 t e^{\lambda t} + \bar{v}_1 e^{\lambda t})) \\
&= c_1 e^{\lambda t} + c_2 t e^{\lambda t} - (d_1 e^{\lambda t} + d_2 t e^{\lambda t}) \\
&= (c_1 - d_1) e^{\lambda t} + (c_2 - d_2) t e^{\lambda t} \\
&= (c_1 - d_1 + (c_2 - d_2)t) e^{\lambda t}. \qquad (4.12)
\end{aligned}$$

Since (4.11) does not give a relationship between d_2 and c_2, two cases will have to be considered: $(c_2 - d_2) \leq 0$ and $(c_2 - d_2) > 0$.

Case 2b.i: $(c_2 - d_2) \leq 0$

In this case, we then have that $(c_2 - d_2)t \leq 0$ for all $t \geq 0$. Also, (4.11) implies that $(c_1 - d_1) \leq 0$. Thus, $(c_1 - d_1 + (c_2 - d_2)t) \leq 0$ for all $t \geq 0$. Since $e^{\lambda t} > 0$ for all $t \geq 0$, we have that $(c_1 - d_1 + (c_2 - d_2)t)e^{\lambda t} \leq 0$ for all $t \geq 0$, implying from (4.12) that $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$. Therefore, the following has been shown:

Theorem 15. *Let $<(x_0, y_0), s>$ and, hence, $(\tilde{x}_0, \tilde{y}_0)$, be given. Assume (A1), (A2), that $\lambda_1 = \lambda_2 = \lambda < 0$ and that λ has only one eigenvector. Let v be an eigenvector of λ and let $\bar{v}$ be the generalized eigenvector, linearly independent of v, found by solving the equation $(A - \lambda I)\bar{v} = v$, where I is the 2x2 identity matrix . If $v_1 = 1$, $\bar{v} = 0$ and $(c_2 - d_2) \leq 0$, then $\dot{x} = Ax$ cannot exhibit tolerance for $<(x_0, y_0), s>$. (i.e. $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$.)*

Case 2b.ii: $(c_2 - d_2) > 0$

In this case, since $v_1 = 1$ and $\bar{v}_1 = 0$, we have from equations (4.7) and (4.8) that the first components of solutions $\phi(t)$ and $\psi(t)$ are, repectively,

$$\begin{aligned} \phi_1(t) &= c_1 v_1 e^{\lambda t} + c_2(v_1 t e^{\lambda t} + \bar{v}_1 e^{\lambda t}) = c_1 e^{\lambda t} + c_2 t e^{\lambda t} = (c_1 + c_2 t)e^{\lambda t} && (4.13)\\ \psi_1(t) &= (d_1 v_1 e^{\lambda t} + d_2(v_1 t e^{\lambda t} + \bar{v}_1 e^{\lambda t})) = d_1 e^{\lambda t} + d_2 t e^{\lambda t} && (4.14)\end{aligned}$$

We are assuming in this subcase that $(c_2 - d_2) > 0$, and we know that $te^{\lambda t} > 0$ for $t > 0$. Thus, we have

$$(c_2 - d_2)te^{\lambda t} > 0, \qquad (4.15)$$

for $t > 0$. Recall that at $t = 0$, $\psi_1(0) > \phi_1(0)$, by definition. Continuing the string of equations from (4.12) and factoring out the quantity $(c_2 - d_2)t$, we have:

$$\begin{aligned} \phi_1(t) - \psi_1(t) &= (c_1 - d_1 + (c_2 - d_2)t)e^{\lambda t}\\ &= \left(\frac{c_1 - d_1}{(c_2 - d_2)t} + 1\right)(c_2 - d_2)te^{\lambda t}. && (4.16)\end{aligned}$$

We would like to know when, if ever, (4.16) is greater than zero, which would then imply that $\psi_1(t) < \phi_1(t)$. Thus, because of (4.15), we are actually interested in when

$$\left(\frac{c_1 - d_1}{(c_2 - d_2)t} + 1\right) > 0. \qquad (4.17)$$

The inequality (4.17) is true if and only if

$$\begin{aligned} (c_1 - d_1) + (c_2 - d_2)t &> 0\\ &\Leftrightarrow (c_2 - d_2)t > (d_1 - c_1)\\ &\Leftrightarrow t > \frac{d_1 - c_1}{c_2 - d_2}.\end{aligned}$$

(Note: $\frac{d_1 - c_1}{c_2 - d_2} \geq 0$, by (4.11) and the assumption for this case that $(c_2 - d_2) > 0$.

Furthermore, from (4.11), we know that $(c_1 - d_1) \leq 0$. This, along with the fact that $(c_2 - d_2)t > 0$ for $t > 0$, implies that

$$\begin{aligned} \frac{c_1 - d_1}{(c_2 - d_2)t} &\leq 0 \\ \Rightarrow \left(\frac{c_1 - d_1}{(c_2 - d_2)t} + 1 \right) &\leq 1. \end{aligned}$$

Hence, when $t > \frac{d_1 - c_1}{c_2 - d_2}$,

$$\begin{aligned} 0 &< \left(\frac{c_1 - d_1}{(c_2 - d_2)t} + 1 \right) \leq 1 \\ &\Rightarrow \; 0 < \left(\frac{c_1 - d_1}{(c_2 - d_2)t} + 1 \right) (c_2 - d_2)te^{\lambda t} \leq (c_2 - d_2)te^{\lambda t} \\ &\Rightarrow \; 0 < \phi_1(t) - \psi_1(t) \leq (c_2 - d_2)te^{\lambda t}. \end{aligned}$$

Thus, the following theorem has been proven:

Theorem 16. *Let $< (x_0, y_0), s >$ and, hence, $(\tilde{x}_0, \tilde{y}_0)$, be given. Assume (A1), (A2), that $\lambda_1 = \lambda_2 = \lambda < 0$ and that λ has only one eigenvector. Let v be an eigenvector of λ and let $\bar{v}$ be the generalized eigenvector, linearly independent of v, found by solving the equation $(A - \lambda I)\bar{v} = v$, where I is the 2x2 identity matrix . If $v_1 = 1$, $\bar{v} = 0$ and $(c_2 - d_2) > 0$, then there exists $T > 0$ such that the 2D linear system $\dot{x} = Ax$ will exhibit tolerance for $< (x_0, y_0), s >$ for all $t > T$. (i.e. $\psi_1(t) < \phi_1(t)$ for all $t > T$). Furthermore,*

$$T = \frac{d_1 - c_1}{c_2 - d_2},$$

and the difference between $\phi_1(t)$ and $\psi_1(t)$ at $t > T$ will be less than or equal to $(c_2 - d_2)te^{\lambda t}$. Therefore, $\max_{t>T} \left\{ (c_2 - d_2)te^{\lambda t} \right\} = \frac{d_2 - c_2}{\lambda e}$, which occurs at $t = \frac{-1}{\lambda}$, is the greatest degree of tolerance that is possible.

4.5.4 Case 3:

$$\lambda_1 \neq \lambda_2$$

For this case, where λ_1 and λ_2 are distinct, negative eigenvalues of A, assume without loss of generality that $\lambda_2 < \lambda_1 < 0$. Let v be an eigenvector of λ_1, and let w be an eigenvector of λ_2. Since λ_1 and λ_2 are distinct, v and w are linearly independent eigenvectors. Then, any initial condition can be uniquely written as a linear combination of the eigenvectors v and w. For the initial condition (x_0, y_0) we may write $(x_0, y_0) = c_1v+c_2w = (c_1v_1+c_2w_1, c_1v_2+c_2w_2)$, with $c_1, c_2 \in \mathbb{R}$. Then, the solution, $\phi(t)$, to the IVP $\dot{x} = Ax$, $\phi(0) = (x_0, y_0)$ is

$$\phi(t) = c_1ve^{\lambda_1 t} + c_2we^{\lambda_2 t} = (c_1v_1e^{\lambda_1 t} + c_2w_1e^{\lambda_2 t}, c_1v_2e^{\lambda_1 t} + c_2w_2e^{\lambda_2 t}) \tag{4.18}$$

Similarly, consider the initial condition $(\tilde{x}_0, \tilde{y}_0)$ defined to be $(\tilde{x}_0, \tilde{y}_0) \equiv (x_0, 0) + \phi(s)$ for some $s \geq 0$. We may also uniquely write $(\tilde{x}_0, \tilde{y}_0) = d_1v + d_2w = (d_1v_1 + d_2w_1, d_1v_2 + d_2w_2)$, with $d_1, d_2 \in \mathbb{R}$. The solution $\psi(t)$ to the IVP $\dot{x} = Ax$, $\psi(0) = (\tilde{x}_0, \tilde{y}_0)$ is

$$\psi(t) = d_1ve^{\lambda_1 t} + d_2we^{\lambda_2 t} = (d_1v_1e^{\lambda_1 t} + d_2w_1e^{\lambda_2 t}, d_1v_2e^{\lambda_1 t} + d_2w_2e^{\lambda_2 t}). \tag{4.19}$$

Then, since we know that $\tilde{x}_0 \geq x_0$, we have that

$$d_1v_1 + d_2w_1 \geq c_1v_1 + c_2w_1. \tag{4.20}$$

As was necessary in Case 1, we consider similar subcases for Case 3 (a) $v_1 = 0$ and $w_1 = 1$ (b) $v_1 = 1$ and $w_1 = 0$ and (c) $v_1 = w_1 = 1$.

4.5.4.1 Case 3a:

$$v_1 = 0 \text{ and } w_1 = 1$$

For this case, (4.20) becomes

$$d_2 \geq c_2 \tag{4.21}$$

Consider the difference between $\phi_1(t)$ and $\psi_1(t)$. Using equations (4.18) and (4.19) and that in this case $v_1 = 0$ and $w_1 = 1$, we have

$$\begin{aligned} \phi_1(t) - \psi_1(t) &= c_2 e^{\lambda_2 t} - d_2 e^{\lambda_2 t} \\ &= (c_2 - d_2) e^{\lambda_2 t}. \end{aligned}$$

By (4.21), we have that $(c_2 - d_2) \leq 0$. Thus, because $e^{\lambda_2 t} > 0$ for all $t \geq 0$, we have that $\phi_1(t) - \psi_1(t) \leq 0$ for all $t \geq 0$. Therefore, the following has been shown:

Theorem 17. *Assume (A1), (A2), and that $\lambda_2 < \lambda_1 < 0$. Given $<(x_0, y_0), s>$ and, hence, $(\tilde{x}_0, \tilde{y}_0)$, if $v_1 = 0$ and $w_1 = 1$ for eigenvectors v and w of λ_1 and λ_2, respectively, then $\dot{x} = Ax$ cannot exhibit tolerance for $<(x_0, y_0), s>$. (i.e. $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$.)*

4.5.4.2 Case 3b:

$$v_1 = 1 \text{ and } w_1 = 0$$

For this second subcase of Case 2, (4.20) becomes

$$d_1 \geq c_1 \tag{4.22}$$

Using equations (4.18) and (4.19) and that $v_1 = 1$ and $w_1 = 0$, we have

$$\begin{aligned} \phi_1(t) - \psi_1(t) &= c_1 e^{\lambda_1 t} - d_1 e^{\lambda_1 t} \\ &= (c_1 - d_1) e^{\lambda_1 t}. \end{aligned}$$

By (4.22), we have that $(c_1 - d_1) \leq 0$ and we know that $e^{\lambda_1 t} > 0$ for all $t \geq 0$. Thus, we conclude $\phi_1(t) - \psi_1(t) \leq 0$ for all $t \geq 0$, and the following has been shown:

Theorem 18. *Assume (A1), (A2), and that $\lambda_2 < \lambda_1 < 0$. Given $<(x_0, y_0), s>$ and, hence, $(\tilde{x}_0, \tilde{y}_0)$, if $v_1 = 1$ and $w_1 = 0$ for eigenvectors v and w of λ_1 and λ_2, respectively, then $\dot{x} = Ax$ cannot exhibit tolerance for $<(x_0, y_0), s>$. (i.e. $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$.)*

4.5.4.3 Case 3c:

$$v_1 = w_1 = 1$$

We first find conditions such that $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$ (i.e. when (4.3) does not exhibit tolerance). At $t = 0$, this is clearly true, because $\tilde{x}_0 \geq x_0$ directly implies that $\psi_1(0) \geq \phi_1(0)$. In this subcase we have that $v_1 = w_1 = 1$. Thus, we may rewrite 4.20 as

$$d_1 + d_2 \geq c_1 + c_2 \tag{4.23}$$

As before, consider the difference between $\phi_1(t)$ and $\psi_1(t)$. Using equations (4.18) and (4.19), the fact that $v_1 = w_1 = 1$, and (4.23), we have

$$\begin{aligned}
\phi_1(t) - \psi_1(t) &= c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} - d_1 e^{\lambda_1 t} - d_2 e^{\lambda_2 t} \\
&= (c_1 - d_1)e^{\lambda_1 t} + (c_2 - d_2)e^{\lambda_2 t} \\
&\leq (c_1 - d_1)e^{\lambda_1 t} + (d_1 - c_1)e^{\lambda_2 t}, \text{ by (4.23)} \\
&= (c_1 - d_1)e^{\lambda_1 t} - (c_1 - d_1)e^{\lambda_2 t} \\
&= (c_1 - d_1)(e^{\lambda_1 t} - e^{\lambda_2 t}).
\end{aligned}$$

Since $\lambda_2 < \lambda_1 < 0$, then $e^{\lambda_1 t} - e^{\lambda_2 t} > 0$. If $(c_1 - d_1) \leq 0$, then $\phi_1(t) - \psi_1(t) \leq 0$, which implies that $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$. Similarly, (4.23) can be used to show

$$d_2 - c_2 \geq c_1 - d_1 \tag{4.24}$$

So, we use (4.24) in the 3rd line of the following inequality:

$$\begin{aligned}
\phi_1(t) - \psi_1(t) &= c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} - d_1 e^{\lambda_1 t} - d_2 e^{\lambda_2 t}, v_1 = w_1 = 1 \\
&= (c_1 - d_1)e^{\lambda_1 t} + (c_2 - d_2)e^{\lambda_2 t} \\
&\leq (d_2 - c_2)e^{\lambda_1 t} + (c_2 - d_2)e^{\lambda_2 t}, \text{ by (4.24)} \\
&= (d_2 - c_2)e^{\lambda_1 t} - (d_2 - c_2)e^{\lambda_2 t} \\
&= (d_2 - c_2)(e^{\lambda_1 t} - e^{\lambda_2 t}).
\end{aligned}$$

Since $\lambda_2 < \lambda_1 < 0$, then $e^{\lambda_1 t} - e^{\lambda_2 t} > 0$. If $(d_2 - c_2) \leq 0$, then $\phi_1(t) - \psi_1(t) \leq 0$, which implies that $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$. With this, we have proven the following theorem:

Theorem 19. *Assume (A1), (A2), and that $\lambda_2 < \lambda_1 < 0$. Also, assume that $v_1 = w_1 = 1$, for eigenvectors v and w of λ_1 and λ_2, respectively. Given $<(x_0, y_0), s>$ and, hence, $(\tilde{x}_0, \tilde{y}_0)$, if $c_1 \leq d_1$ OR if $c_2 \geq d_2$, then $\dot{x} = Ax$ cannot exhibit tolerance for $<(x_0, y_0), s>$. (i.e. $\psi_1(t) \geq \phi_1(t)$ for all $t \geq 0$.) Thus, it is necessary that $c_1 > d_1$ AND $c_2 < d_2$ for tolerance to be exhibited.*

Now we show that for Case 3c the necessary conditions $c_1 > d_1$ AND $c_2 < d_2$ from Theorem 19 are also sufficient for tolerance to be exhibited in the $2D$ linear system $\dot{x} = Ax$. Assume that $c_1 > d_1$ AND $c_2 < d_2$. Using equations (4.18) and (4.19) and the fact that $v_1 = w_1 = 1$,

$$\begin{aligned}\phi_1(t) - \psi_1(t) &= c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} - d_1 e^{\lambda_1 t} - d_2 e^{\lambda_2 t} \\ &= (c_1 - d_1) e^{\lambda_1 t} + (c_2 - d_2) e^{\lambda_2 t}.\end{aligned}$$

Factoring out the quantity $(c_1 - d_1)e^{\lambda_2 t}$ from the right hand side gives:

$$\phi_1(t) - \psi_1(t) = \left(e^{(\lambda_1 - \lambda_2)t} + \frac{c_2 - d_2}{c_1 - d_1} \right) e^{\lambda_2 t}(c_1 - d_1).$$

We know that $(c_1 - d_1) > 0$ and $(c_2 - d_2) < 0$, since we assumed $c_1 > d_1$ and $c_2 < d_2$. Thus,

$$e^{\lambda_2 t}(c_1 - d_1) > 0$$

and

$$\frac{(c_2 - d_2)}{(c_1 - d_1)} < 0.$$

Therefore,

$$\begin{aligned}
\phi_1(t) - \psi_1(t) &= \left(e^{(\lambda_1-\lambda_2)t} + \frac{(c_2-d_2)}{(c_1-d_1)} \right) e^{\lambda_2 t}(c_1 - d_1) \\
&> 0 \\
&\Leftrightarrow \left(e^{(\lambda_1-\lambda_2)t} + \frac{(c_2-d_2)}{(c_1-d_1)} \right) > 0 \\
&\Leftrightarrow e^{(\lambda_1-\lambda_2)t} > \frac{(d_2-c_2)}{(c_1-d_1)} \\
&\Leftrightarrow \ln e^{(\lambda_1-\lambda_2)t} > \ln \frac{(d_2-c_2)}{(c_1-d_1)} \\
&\Leftrightarrow t > \frac{\ln \frac{(d_2-c_2)}{(c_1-d_1)}}{(\lambda_1-\lambda_2)}
\end{aligned}$$

Thus, we have proven the following theorem:

Theorem 20. *Assume (A1), (A2), and that $\lambda_2 < \lambda_1 < 0$. Also, assume that $v_1 = w_1 = 1$ for eigenvectors v and w of λ_1 and λ_2, respectively. Given $< (x_0, y_0), s >$ and, hence, $(\tilde{x}_0, \tilde{y}_0)$, if $c_1 > d_1$ AND $c_2 < d_2$ then there exists $T > 0$ such that the 2D linear system $\dot{x} = Ax$ will exhibit tolerance for $< (x_0, y_0), s >$ for all $t > T$. (i.e. $\psi_1(t) < \phi_1(t)$ for all $t > T$). Furthermore,*

$$T = \frac{\ln \frac{d_2-c_2}{c_1-d_1}}{\lambda_1 - \lambda_2}.$$

4.5.5 Eigenvector Configurations and Regions of Tolerance

Now we consider the eigenvector configurations of 2D linear systems for the cases that accommodate solutions that begin and remain in the first quadrant and that converge to $(0, 0)$ as $t \to \infty$. Of the cases discussed above, only cases 2b.ii and 3c yield the possibility of tolerance. For the relevant eigenvector configurations of these cases, we analyze the first quadrant to determine where the nonnegativity requirement is satisfied for solutions originating there. In each of the eigenvector configurations shown in Figure 26, there are several regions of the first quadrant that will be considered. Once the pertinent regions have been identified, we then determine where tolerance will and will not occur.

Remark 4. *In the following analysis to determine relevant regions, we do not consider* (x_0, y_0) *on the y-axis, since this would imply that* $x_0 = 0$*, yet it was assumed that* $x_0 > 0$*. (see page 112)*

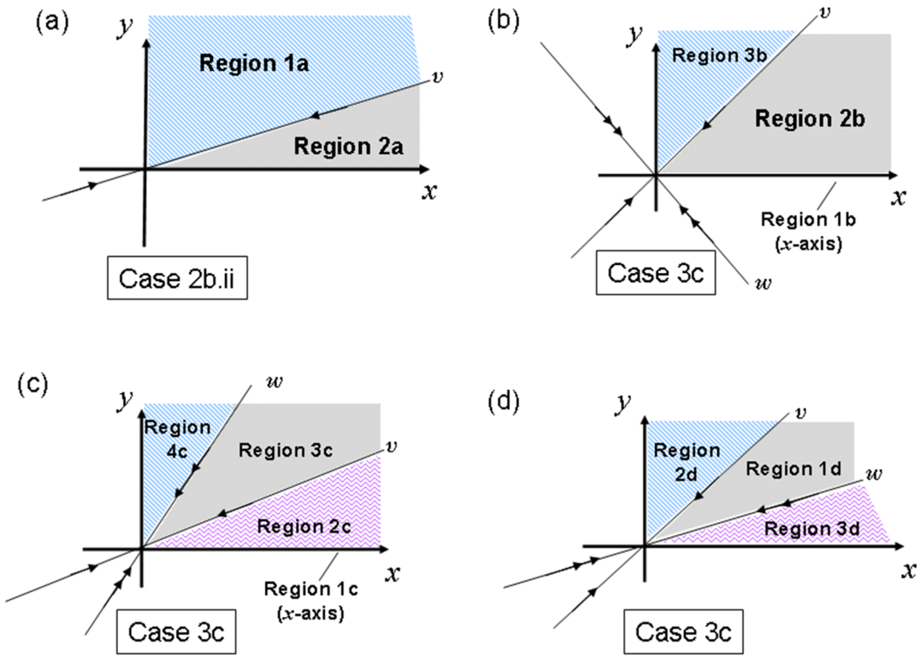


Figure 26: Regions in the first quadrant for all relevant eigenvector configurations

Figure 27 shows a pictoral summary of the results via example trajectories originating in the different regions of the various eigenvector configurations. These results are made explicit in the following analysis. In all the cases, since the initial condition (x_0, y_0) under consideration is in the first quadrant, it is clear that the solution of the IVP, $\phi(t)$, at $t = 0$ is $\phi(0) > \mathbf{0}$, thus we consider the solutions for all $t > 0$.

4.5.5.1 Eigenvector Configuration (a)**:** Eigenvector configuration (a) of Figure 26a illustrates the only possible eigenvector configuration in Case 2b.ii for which there exist solutions that begin and remain in the first quadrant. We first show where in the first quadrant such solutions exist. Recall that in this case, the matrix A in (4.3) has one eigenvalue, λ, of multiplicity two, having only one independent eigenvector, v. A generalized

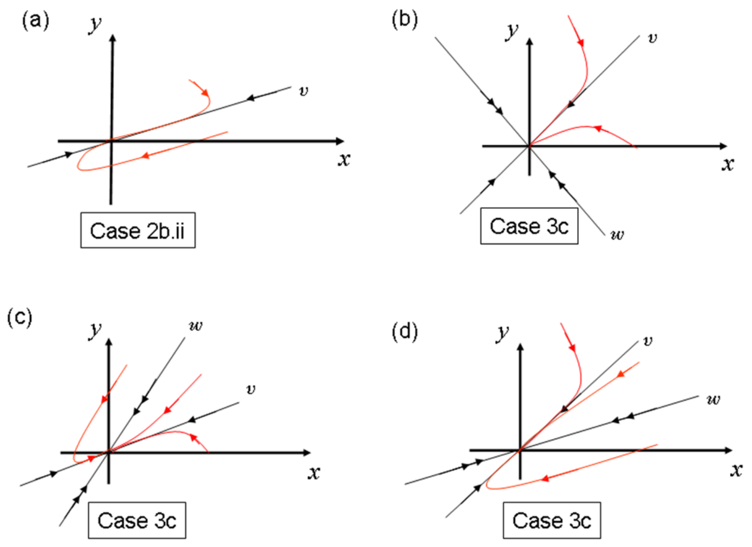


Figure 27: Eigenvector configurations for $2D$ systems which accomodate solutions that begin and remain in the first quadrant and converge to $(0,0)$ as $t \longrightarrow \infty$.

eigenvector, $\bar{v}$, was found so as to construct a general solution, given an intial condition. For initial conditions $(x_0, y_0) = c_1 v + c_2 \bar{v} = (c_1 v_1 + c_2 \bar{v}_1, c_1 v_2 + c_2 \bar{v}_2)$ and $(\tilde{x}_0, \tilde{y}_0) = d_1 v + d_2 \bar{v} = (d_1 v_1 + d_2 \bar{v}_1, d_1 v_2 + d_2 \bar{v}_2)$,with $c_1, c_2, d_1, d_2 \in \mathbb{R}$, the first component of the solutions to the inital value problems are given by equations (4.13) and (4.14), respectively. Furthermore, in configuration (a), $v_1 = 1$, $v_2 > 0$, $\bar{v}_1 = 0$, and $\bar{v}_2 > 0$. In addition, recall (4.11): $d_1 \geq c_1$.

Let (x_0, y_0) be in Region 1a of the first quadrant bounded by v and the y-axis, seen in Figure 26a. We show that the first component of the solution, $\phi(t)$, of the IVP with initial condition (x_0, y_0) will remain positive for all $t > 0$. This will be sufficient since it would guarantee that $\phi_2(t) > 0$ for all $t > 0$, because $\phi(t)$ cannot cross the v-eigenvector. Clearly, at $t = 0$, $\phi_1(0) > 0$. In Region 1a, for any $(x_0, y_0) = c_1 v + c_2 \bar{v}$, we have that $c_1 > 0$ and $c_2 > 0$. Therefore,

$$\begin{aligned} c_1 + c_2 t &> 0, \forall t > 0 \\ \Rightarrow (c_1 + c_2 t)e^{\lambda t} &> 0, \forall t > 0 \\ \Rightarrow \phi_1(t) &> 0, \forall t > 0, \end{aligned}$$

by equation (4.13), which defines $\phi_1(t)$.

Now, consider (x_0, y_0) in Region 2a of eigenvector configuration (a), shown in Figure 26a. We show that the first component of the solution, $\phi(t)$, of the IVP with this initial condition will eventually be negative in the first component. This will be sufficient to confirm that the solution does not stay in the first quadrant. In this region, for any $(x_0, y_0) = c_1 v + c_2 \bar{v}$, we have that $c_1 > 0$ and $c_2 < 0$. Therefore, since $e^{\lambda\tau} > 0$ for all $t > 0$,

$$\begin{aligned} \phi_1(t) &= (c_1 + c_2 t)e^{\lambda t} < 0, \, \forall t > 0 \\ &\Leftrightarrow c_1 + c_2 t < 0, \, \forall t > 0 \\ &\Leftrightarrow c_2 t < -c_1, \, \forall t > 0 \\ &\Leftrightarrow t > \frac{-c_1}{c_2} \end{aligned}$$

Note that because $c_2 < 0$, the inequality changes in the last line when division by c_2 is made. In addition $\frac{-c_1}{c_2} > 0$, since $c_1 > 0$ and $c_2 < 0$. Therefore, $\phi_1(t) < 0$ for all $t > \frac{-c_1}{c_2}$. In other words, for eigenvector configuration (a), solutions starting in Region 2a will eventually be negative in the first component.

Thus, in eigenvector configuration (a), seen in figure 28, there is one region in which to consider initial conditions to explore the existance of tolerance.

- **REGION 1a**: (x_0, y_0) in the first quadrant above v

The conclusion regarding tolerance for this case (Case 2b.ii) was given by Theorem 16, which shows (along with Theorem 15) that the condition $(c_2 - d_2) > 0$ is necessary and sufficient in this case for tolerance to be exhbited in (4.3). In the left panel of Figure 29 an arbitrary point in Region 1a is shown in the context of eigenvector configuration (a), with lines drawn (portions dashed), showing the addition of scalar multiples of the eigenvector v and the generalized eigenvector, $\bar{v}$, in the creation of the point (x_0, y_0). Although the generalized eigenvector, $\bar{v}$, is used to write the intial conditions, there is only one eigendirection, so trajectories can cross $\bar{v}$. Since in this case, $\bar{v} = 0$, the blue line along the y-axis represents $\bar{v}$. The lines showing the creation of (x_0, y_0) are referred to as the c_1-line and c_2-line. They divide the first quadrant into four different regions as shown in the right panel of Figure 29.

In these regions, there are relationships between the coefficients $c_1$and c_2 and the coefficients of other points in the first quadrant written as a linear combination of the eigenvectors,

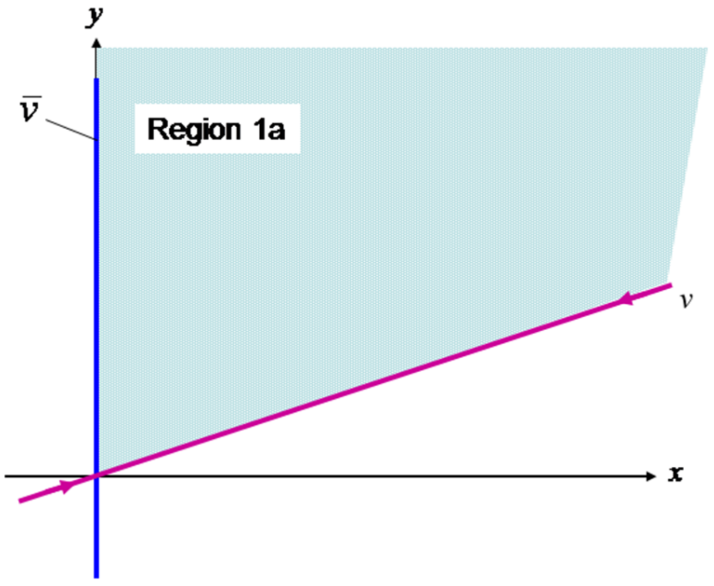


Figure 28: Eigenvector configuration (a) with the first quadrant divided into three regions where initial conditions yield solutions that begin and remain in the first quadrant.

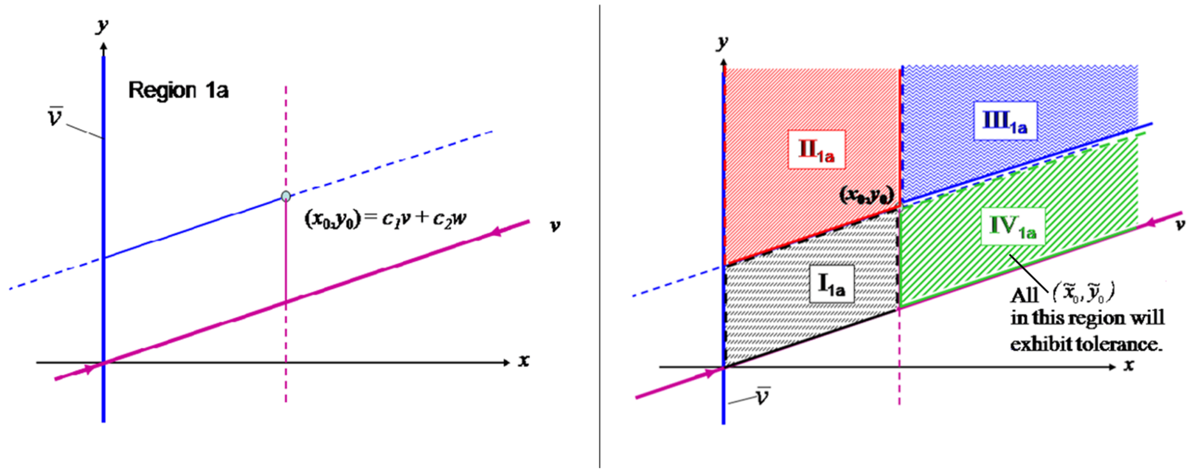


Figure 29: Left Panel: Eigenvector configuration (a) with an arbitrary initial condition (x_0, y_0) labeled in Region 1a (x-axis) along with lines drawn (portions dashed), showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (a) divided into three regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1v + c_2w$ lying in Region 1a.

namely, the coefficients of the bumped initial condition, $(\tilde{x}_0, \tilde{y}_0) = d_1 v + d_2 \bar{v}$. If $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_a$, then using the c_1-line and c_2-line lines for reference, we see that $d_1 < c_1$ and $d_2 < c_2$ in region $\mathbb{I}_{1a}$. Applying this reasoning to each region, we have by (4.11) and Theorems 15 and 16:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1a}$, $d_1 < c_1$ and $d_2 < c_2$. Since $d_1 < c_1$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1a}$, for (x_0, y_0) in Region 1a.
2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{1a}$, $d_1 < c_1$ and $d_2 \geq c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{1a}$, when (x_0, y_0) in Region 1a.
3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1a}$, $d_1 > c_1$ and $d_2 \geq c_2$. Since $d_2 \geq c_2$, tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1a}$, for (x_0, y_0) in Region 1a.
4. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{1a}$, $d_1 \geq c_1$ and $d_2 < c_2$. **Since both $d_1 \geq c_1$ and $d_2 < c_2$, tolerance will be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{1a}$, when (x_0, y_0) is in Region 1a.**

Hence, for eigenvector configuration (a), if (x_0, y_0) is in the first quadrant above the eigenvector, v, then tolerance will be exhibited only when $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{1a}$, which is the green area shown in the right panel of Figure 29. Note that, of course, $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1a}$ contradicts the definition of $(\tilde{x}_0, \tilde{y}_0)$, since in this region, $\tilde{x}_0 < x_0$, yet $\tilde{x}_0$ is defined to be greater than x_0. This is also the case for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{1a}$; however, we are just considering the regions as a whole and once the regions have been classified, we determine which regions are relevant with respect to where the $(\tilde{x}_0, \tilde{y}_0)$-curve is. In the Examples section (4.5.6), Example 1 shows an example of a trajectory, whose initial condition lies in Region 1a, and its corresponding $(\tilde{x}_0, \tilde{y}_0)$-curve, a portion of which lies in the region marked $\mathbb{IV}_{1a}$.

Panels (b)-(d) of Figure 26 depict the only possible eigenvector configurations for Case 3c, for which solutions exist that begin and remain in the first quadrant. For these configurations, there are two eigenvalues for the matrix, A, of (4.3), each having a corresponding eigenvector. Thus, WLOG assume $\lambda_2 < \lambda_1 < 0$, with v and w the linear independent eigenvectors of λ_1 and λ_2, respectively. In this case (Case 3c) it has been arranged for $v = [1\ v_2]^T$ and $w = [1\ w_2]^T$. ($T \equiv$ transpose) In other words, $v_1 = w_1 = 1$. Furthermore, for initial conditions $(x_0, y_0) = c_1 v + c_2 w = (c_1 v_1 + c_2 w_1, c_1 v_2 + c_2 w_2) = (c_1 + c_2, c_1 v_2 + c_2 w_2)$ and $(\tilde{x}_0, \tilde{y}_0) = d_1 v + d_2 w = (d_1 v_1 + d_2 w_1, d_1 v_2 + d_2 w_2) = (d_1 + d_2, d_1 v_2 + d_2 w_2)$,with $c_1, c_2, d_1, d_2 \in \mathbb{R}$, the solutions to the inital value problems are given by equations (4.18) and (4.19), respectively.

Remark 5. *Note that $x_0 > 0$, implies that $c_1 + c_2 > 0 \Rightarrow c_1 > -c_2$ and that $y_0 > 0$, implies that $c_1 v_2 + c_2 w_2 > 0 \Rightarrow c_1 v_2 > -c_2 w_2$.*

Remark 6. *Note that because $\lambda_2 < \lambda_1 < 0$ in this case, we have that $(\lambda_1 - \lambda_2) > 0$ and consequently, that $e^{(\lambda_1-\lambda_2)t} > 1$ for all $t > 0$.*

The above remarks will be used throughout the analysis of eigenvector configurations (b)-(d) in determining relevant regions that satisfy the nonnegativity assumption (A2).

4.5.5.2 Eigenvector Configuration *(b)*:

We first determine which of the regions shown in Figure 26b are relevant with respect to solutions that remain in the first quadrant for all $t > 0$. In eigenvector configuration (b), we have that $v_2 > 0$, $w_2 < 0$, since it is assumed that $v_1 = w_1 = 1$.

We group Regions 1b and 2b of Figure 26b, since the proofs are the same. Thus, consider an arbitrary inital condition, (x_0, y_0), in Region 1b or Region 2b. In either region, $c_1 > 0$ and $c_2 > 0$. Because of the position of the eigenvectors in this configuration, $\phi_1(t) > 0$ for all $t > 0$. Therefore, it must be shown that $\phi_2(t) > 0$ for all $t > 0$. Since $c_1 > 0$ and $v_2 > 0$, Remarks (5) and (6) give the following:

$$\begin{aligned} c_1 v_2 e^{(\lambda_1-\lambda_2)t} &> -c_2 w_2 && (4.25)\\ \Rightarrow c_1 v_2 e^{\lambda_1 t} &> -c_2 w_2 e^{\lambda_2 t} && (4.26)\\ \Rightarrow c_1 v_2 e^{\lambda_1 t} + c_2 w_2 e^{\lambda_2 t} &> 0 && (4.27) \end{aligned}$$

for all $t > 0$. This implies from equation (4.18) that $\phi_2(t) > 0$ for all $t > 0$. In other words, for eigenvector configuration *(b)*, solutions starting in Region 1b or Region 2b will remain nonnegative for all $t \geq 0$.

Now consider an arbitrary intial condition in Region 3b shown in Figure 26b. It is sufficient to show that $\phi_1(t) > 0$ for all $t > 0$ since the nonnegativity of $\phi_2(t)$ is guaranteed

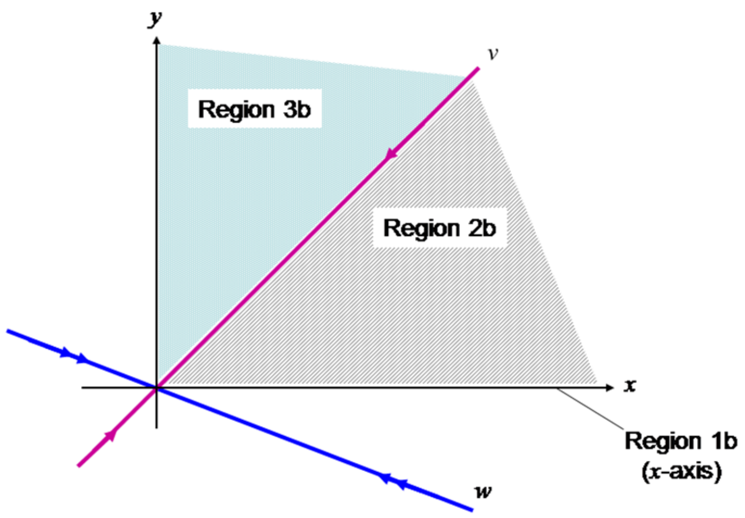


Figure 30: Eigenvector configuration (b) with the first quadrant divided into three regions where initial conditions yield solutions that begin and remain in the first quadrant.

by the position of the eigenvectors v and w, through which the solution cannot cross. In Region 3b, $c_1 > 0$ and $c_2 < 0$. Since $c_1 > 0$, Remarks (5) and (6) give the following:

$$\begin{aligned} c_1 e^{(\lambda_1 - \lambda_2)t} &> -c_2 \\ \Rightarrow c_1 e^{\lambda_1 t} &> -c_2 e^{\lambda_2 t} \\ \Rightarrow c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} &> 0 \end{aligned}$$

for all $t > 0$, which implies from equation (4.18) that $\phi_1(t) > 0$ for all $t > 0$.

Thus, for eigenvector configuration (b), seen in figure 30, there are three regions in which to consider initial conditions:

- **REGION 1b**: (x_0, y_0) on the x-axis
- **REGION 2b**: (x_0, y_0) in the first quadrant below the v eigenvector and above the x-axis
- **REGION 3b**: (x_0, y_0) in the first quadrant above the v eigenvector

REGION 1b: First, we look at the case when the initial condition is on the x-axis. In the left panel of Figure 31, an arbitrary point on the x-axis is shown in the context of

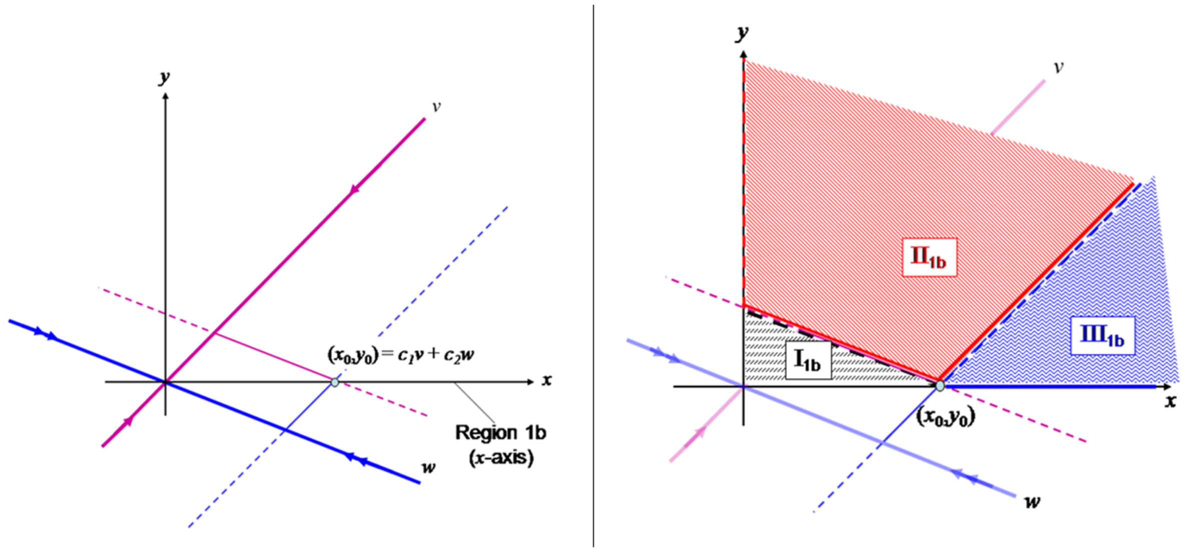


Figure 31: Left Panel: Eigenvector configuration (b) with an arbitrary initial condition (x_0, y_0) labeled in Region 1b (x-axis) along with lines drawn (portions dashed), showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (b) divided into three regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1v + c_2w$ lying in Region 1b.

eigenvector configuration (b), with lines drawn (portions dashed), showing the addition of scalar multiples of the two eigenvectors in the creation of the point (x_0, y_0). The right panel of Figure 31 shows the three regions formed in the first quadrant by the c_1-line and c_2-line. Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1b}$, $d_1 < c_1$ and $d_2 < c_2$. Since $d_2 < c_2$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1a}$, when (x_0, y_0) in Region 1b.
2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{1b}$, $d_1 \geq c_1$ and $d_2 \leq c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{1b}$, when (x_0, y_0) in Region 1b.
3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1b}$, $d_1 > c_1$ and $d_2 > c_2$. Since $d_1 > c_1$, tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1b}$, when (x_0, y_0) in Region 1b.

Hence, for eigenvector configuration (b), if (x_0, y_0) is on the x-axis, there are no regions in the first quadrant where both $d_1 < c_1$ and $d_2 > c_2$ and, thus, for any $(\tilde{x}_0, \tilde{y}_0)$ there can be no tolerance.

REGION 2b: Let (x_0, y_0) be in the first quadrant below the v eigenvector (but not

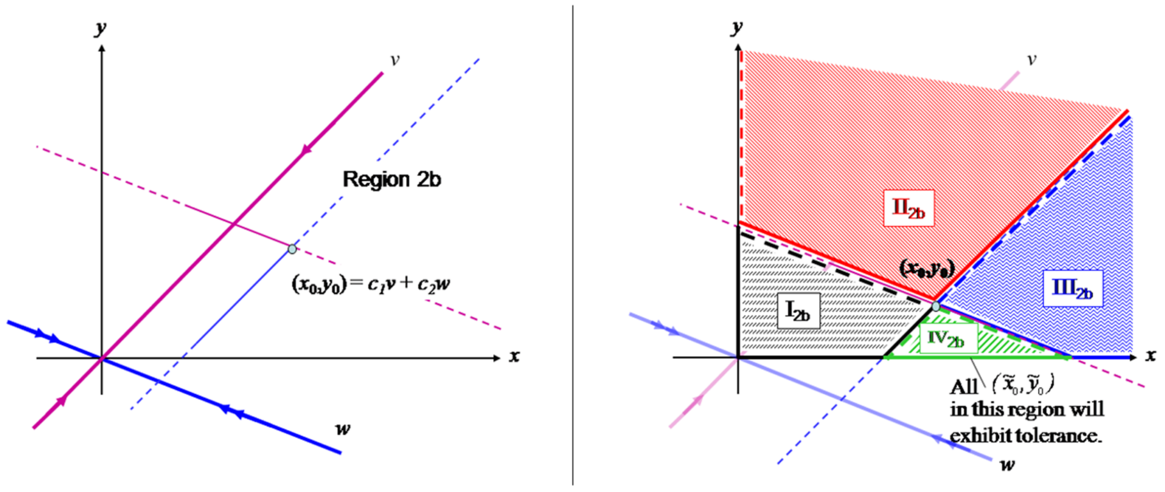


Figure 32: Left Panel: Eigenvector configuration (b) with an arbitrary initial condition (x_0, y_0) labeled in Region 2b along with lines drawn (portions dashed) showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (b) divided into four regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1v + c_2w$ lying in Region 2b.

on the x-axis) in eigenvector configuration (b). Figure 32 shows an arbitrary point in this region, with lines drawn (portions dashed), showing the addition of the two eigenvectors in the creation of the point (x_0, y_0). The right panel of Figure 32 shows the four regions formed in the first quadrant by the c_1-line and c_2-line. Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{2b}$, $d_1 < c_1$ and $d_2 \le c_2$. Since $d_2 \le c_2$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{2b}$, when (x_0, y_0) is in Region 2b.
2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{2b}$, $d_1 \ge c_1$ and $d_2 \le c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{2b}$, when (x_0, y_0) is in Region 2b.
3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{2b}$, $d_1 \ge c_1$ and $d_2 > c_2$. Since $d_1 \ge c_1$, tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{2b}$, when (x_0, y_0) is in Region 2b.
4. **For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2b}$, $d_1 < c_1$ and $d_2 > c_2$. Since both $d_1 < c_1$ and $d_2 > c_2$, tolerance will be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2b}$, when (x_0, y_0) is in Region 2b.**

Hence, for eigenvector configuration (b), if (x_0, y_0) is in the first quadrant below the v eigenvector (but not on the x-axis), then tolerance will be exhibited only when $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2b}$, which is the green area shown in Figure 32. In the Examples section (4.5.6), Example 2 shows

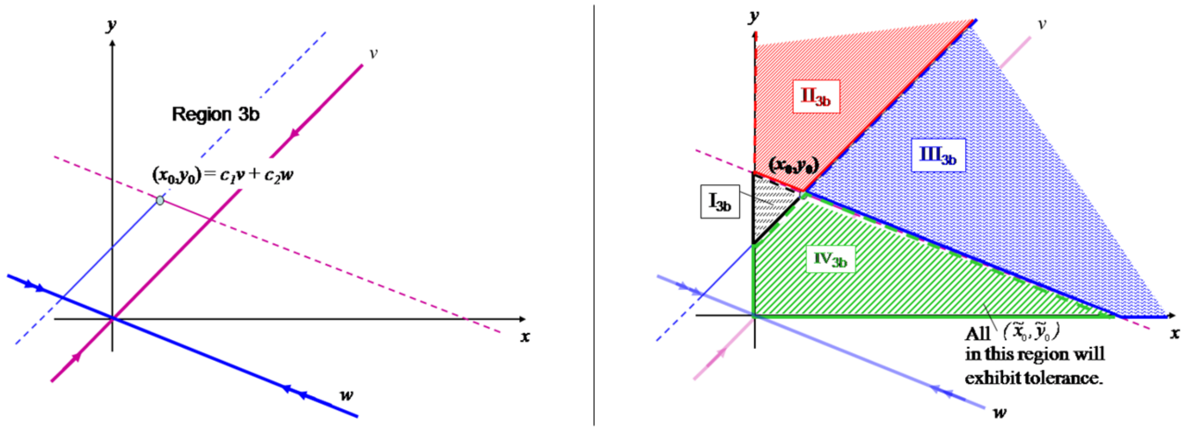


Figure 33: Left Panel: Eigenvector configuration (b) with an arbitrary initial condition (x_0, y_0) labeled in Region 3b along with lines drawn (portions dashed) showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (b) divided into four regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1v + c_2w$ lying in Region 3b.

an example of a trajectory, whose initial condition lies in Region 2b and, its corresponding $(\tilde{x}_0, \tilde{y}_0)$-curve, a portion of which lies in the region marked $\mathbb{IV}_{2b}$.

REGION 3b: Let (x_0, y_0) be in the first quadrant above the v eigenvector in eigenvector configuration (b). Figure 33 shows an arbitrary point in this region, with lines drawn (portions dashed), showing the addition of the two eigenvectors in the creation of the point (x_0, y_0). The right panel of Figure 33 shows the four regions formed in the first quadrant by the c_1-line and c_2-line. Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{3b}$, $d_1 < c_1$ and $d_2 \leq c_2$. Since $d_2 \leq c_2$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{3b}$, when (x_0, y_0) is in Region 3b.
2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{3b}$, $d_1 \geq c_1$ and $d_2 \leq c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{3b}$, when (x_0, y_0) is in Region 3b.
3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{3b}$, $d_1 \geq c_1$ and $d_2 > c_2$. Since $d_1 \geq c_1$, tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{3b}$, when (x_0, y_0) is in Region 3b.
4. **For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{3b}$, $d_1 < c_1$ and $d_2 > c_2$. Since both $d_1 < c_1$ and $d_2 > c_2$, tolerance will be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{3b}$, when (x_0, y_0) is in Region 3b.**

Hence, for eigenvector configuration (b), if (x_0, y_0) is in the first quadrant above the v

eigenvector, then tolerance will be exhibited only when $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{3b}$, which is the green region shown in the right panel of Figure 33. In the Examples section (4.5.6), Example 3 shows an example of a trajectory, whose initial condition lies in Region 3b, and its corresponding $(\tilde{x}_0, \tilde{y}_0)$-curve, a portion of which lies in the region marked $\mathbb{IV}_{3b}$.

4.5.5.3 Eigenvector Configuration (c) For configuration (c), we also first determine which of the regions shown in Figure 26c are relevant with respect to solutions that remain in the first quadrant for all $t > 0$. In this eigenvector configuration, we have that $v_2 > 0$, $w_2 > 0$, since it is assumed that $v_1 = w_1 = 1$.

We group Regions 1c and 2c of Figure 26c, since the proofs are the same. Thus, consider an arbitrary inital condition, (x_0, y_0), in Region 1c or Region 2c. In either region, $c_1 > 0$ and $c_2 < 0$. Because of the position of the eigenvectors in this configuration, it is sufficient to show that $\phi_2(t) > 0$ for all $t > 0$, which will then imply that $\phi_1(t) > 0$ for all $t > 0$, as well. Since $c_1 > 0$ and $v_2 > 0$, Remarks (5) and (6) give the following:

$$\begin{aligned} c_1 v_2 e^{(\lambda_1 - \lambda_2)t} &> -c_2 w_2 \\ \Rightarrow c_1 v_2 e^{\lambda_1 t} &> -c_2 w_2 e^{\lambda_2 t} \\ \Rightarrow c_1 v_2 e^{\lambda_1 t} + c_2 w_2 e^{\lambda_2 t} &> 0 \end{aligned}$$

for all $t > 0$. This implies from equation 4.18 that $\phi_2(t) > 0$ for all $t > 0$.

Now consider an arbitrary intial condition in Region 3c shown in Figure 26c. All solutions originating in this region will be bounded by the eigenvectors v and w and will remain in the first quadrant for all $t > 0$.

For an an arbitrary intial condition in Region 4c shown in Figure 26c, we will show that solutions starting in this region do not remain in the first quadrant for all $t > 0$. It is sufficient to show that $\phi_1(t) < 0$ for some $t > 0$. In Region 4c, $c_1 < 0$ and $c_2 > 0$. We determine when $\phi_1(t) < 0$ or equivalently, when the following inequality holds:

$$\begin{aligned} c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} &< 0 \\ \Leftrightarrow c_1 e^{\lambda_1 t} &< -c_2 e^{\lambda_2 t} \\ \Leftrightarrow c_1 e^{(\lambda_1 - \lambda_2)t} &< -c_2 \\ \Leftrightarrow e^{(\lambda_1 - \lambda_2)t} &> \frac{-c_2}{c_1} \\ &\Leftrightarrow \ln e^{(\lambda_1-\lambda_2)t} > \ln\left(\frac{-c_2}{c_1}\right) \\ &\Leftrightarrow t > \frac{\ln\left(\frac{-c_2}{c_1}\right)}{(\lambda_1 - \lambda_2)}. \end{aligned}$$

Note that $\left(\frac{-c_2}{c_1}\right) > 0$ since $c_1 < 0$ and $c_2 > 0$ in this region. Hence, it has been shown that solutions starting in Region 4c do not remain in the first quadrant.

Thus, for eigenvector configuration (c), seen in Figure 34, there are three regions in which to consider initial conditions:

- **REGION 1c**: (x_0, y_0) on the x-axis
- **REGION 2c**: (x_0, y_0) in the first quadrant below the v eigenvector and above the x-axis
- **REGION 3c**: (x_0, y_0) in the first quadrant above the v eigenvector and below the w eigenvector

REGION 1c: First, we look at the case when the initial condition is on the x-axis. In the left panel of Figure 35, an arbitrary point on the x-axis is shown in the context of eigenvector configuration (c), with lines drawn (portions dashed), showing the addition of the two eigenvectors in the creation of the point (x_0, y_0). Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1c}$, $d_1 < c_1$ and $d_2 > c_2$. **Since both $d_1 < c_1$ and $d_2 > c_2$, tolerance will be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1c}$, when (x_0, y_0) is in Region 1c.**
2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1c}$, $d_1 \geq c_1$ and $d_2 > c_2$. Since $d_1 \geq c_1$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1c}$, when (x_0, y_0) is in Region 1c.

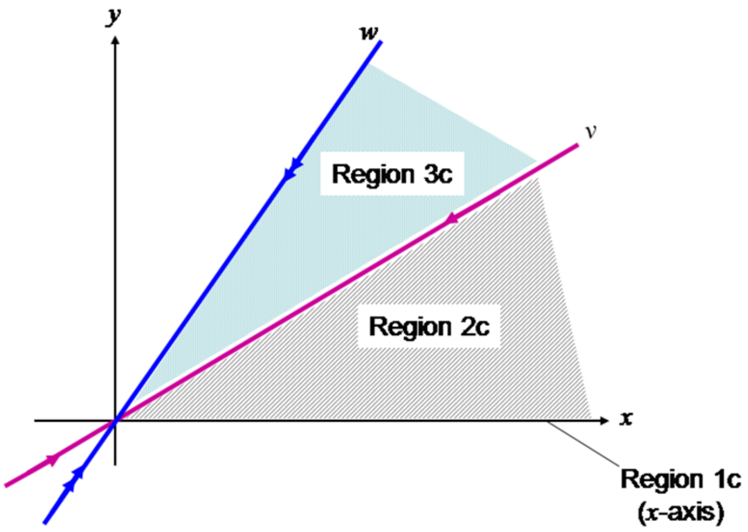


Figure 34: Eigenvector configuration (c) with the first quadrant divided into three regions where initial conditions yield solutions that begin and remain in the first quadrant.

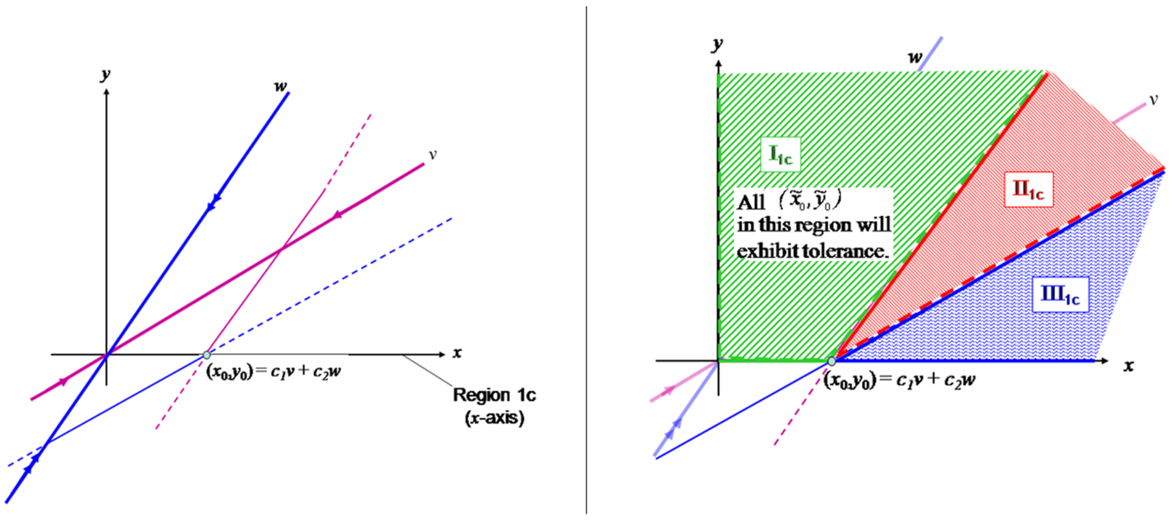


Figure 35: Left Panel: Eigenvector configuration (c) with an arbitrary initial condition (x_0, y_0) labeled in Region 1c (x-axis) along with lines drawn (portions dashed) showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (c) divided into three regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1v + c_2w$ lying in Region 1c.

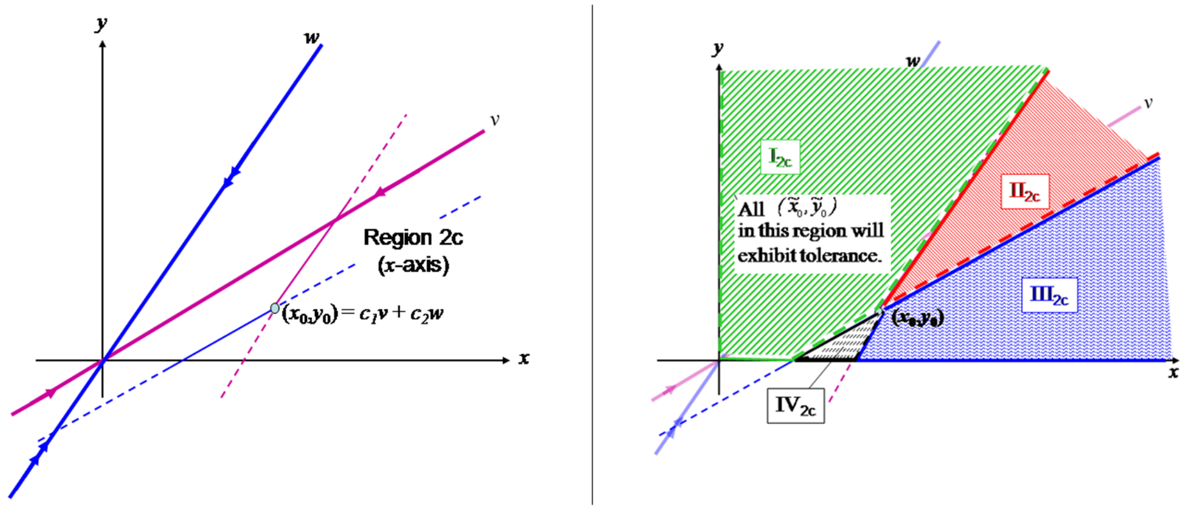


Figure 36: Left Panel: Eigenvector configuration (c) with an arbitrary initial condition (x_0, y_0) labeled in Region 2c along with lines drawn (portions dashed) showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (c) divided into four regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1 v + c_2 w$ lying in Region 2c.

3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IIII}_{1c}$, $d_1 > c_1$ and $d_2 < c_2$. Either of these relationships imply that tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IIII}_{1c}$, when (x_0, y_0) is in Region 1c.

Hence, for eigenvector configuration (c), if (x_0, y_0) is on the x-axis, then tolerance will be exhibited only when $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1c}$, which is the green region shown in the right panel of Figure 35.

REGION 2c: Let (x_0, y_0) be in the first quadrant below the v eigenvector (but not on the x-axis) in eigenvector configuration (c). The left panel of Figure 36 shows an arbitrary point in this region, with lines drawn (portions dashed), showing the addition of the two eigenvectors in the creation of the point (x_0, y_0). The right panel of Figure 36 shows the four regions formed in the first quadrant by the c_1-line and c_2-line. Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{2c}$, $d_1 < c_1$ and $d_2 > c_2$. **Since both $d_1 < c_1$ and $d_2 > c_2$, tolerance will be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{2c}$, when (x_0, y_0) is in Region 2c.**
2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{2c}$, $d_1 \geq c_1$ and $d_2 > c_2$. Since $d_1 \geq c_1$, tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{2c}$, when (x_0, y_0) is in Region 2c.

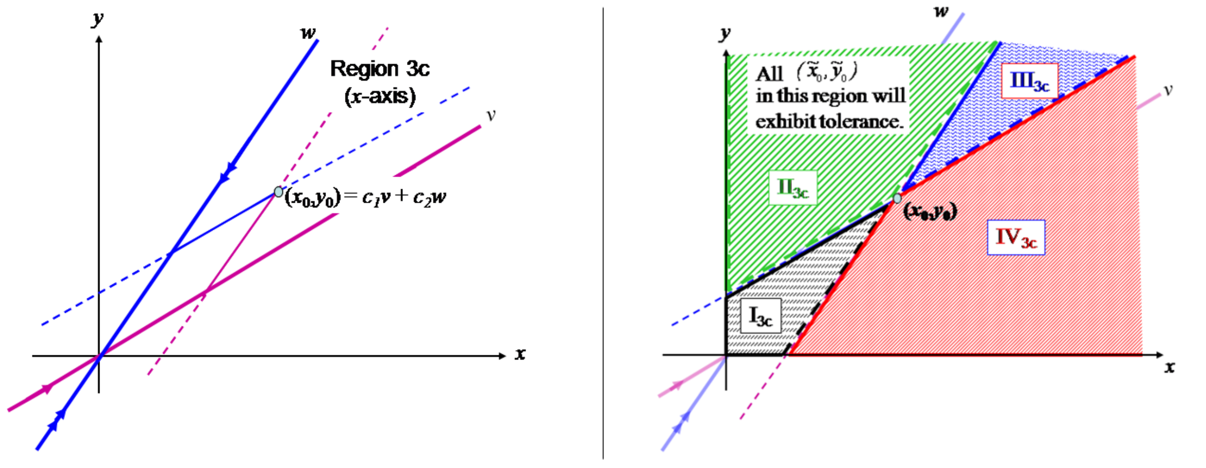


Figure 37: Left Panel: Eigenvector configuration (c) with an arbitrary initial condition (x_0, y_0) labeled in Region 3c along with lines drawn (portions dashed) showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (c) divided into four regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1v + c_2w$ lying in Region 3c.

3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{2c}$, $d_1 \geq c_1$ and $d_2 \leq c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{2c}$, when (x_0, y_0) is in Region 2c.
4. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2c}$, $d_1 < c_1$ and $d_2 \leq c_2$. Since $d_2 \leq c_2$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2c}$, when (x_0, y_0) is in Region 2c.

Hence, for eigenvector configuration (c), if (x_0, y_0) is in the first quadrant below the v eigenvector (but not on the x-axis), then tolerance will be exhibited only when $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{2c}$, which is the green region shown in the right panel of Figure 36. In the Examples section (4.5.6), Example 4 shows an example of a trajectory whose initial condition lies in Region 2c and its corresponding $(\tilde{x}_0, \tilde{y}_0)$-curve, no points of which lie in the region marked $\mathbb{I}_{2c}$.

REGION 3c: Let (x_0, y_0) be in the first quadrant above the v eigenvector and below the w eigenvector in eigenvector configuration (c). The left panel of Figure 37 shows an arbitrary point in this region, with lines drawn (portions dashed) showing the addition of the two eigenvectors in the creation of the point (x_0, y_0). The right panel of Figure 37 shows the four regions formed in the first quadrant by the c_1-line and c_2-line. Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{3c}$, $d_1 < c_1$ and $d_2 \leq c_2$. Since $d_2 \leq c_2$, tolerance cannot be exhibited for

any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{3c}$, when (x_0, y_0) is in Region 3c.

2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{3c}$, $d_1 < c_1$ and $d_2 > c_2$. **Since both $d_1 < c_1$ and $d_2 > c_2$, tolerance will be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{3c}$, when (x_0, y_0) is in Region 3c.**
3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{3c}$, $d_1 \geq c_1$ and $d_2 > c_2$. Since $d_1 \geq c_1$, tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{3c}$, when (x_0, y_0) is in Region 3c.
4. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{3c}$, $d_1 \geq c_1$ and $d_2 \leq c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{3c}$, when (x_0, y_0) is in Region 3c.

Thus, for eigenvector configuration (c), if $(\tilde{x}_0, \tilde{y}_0)$ is in the first quadrant above the v eigenvector and below the w eigenvector, then tolerance will be exhibited only when $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{3c}$, which is the green region shown in the right panel of Figure 37. In the Examples section (4.5.6), Example 5 shows an example of a trajectory, whose initial condition lies in Region 3c, and its corresponding $(\tilde{x}_0, \tilde{y}_0)$-curve, no points of which lie in the region marked $\mathbb{II}_{3c}$.

4.5.5.4 Eigenvector Configuration (d)

For the last configuration, we again determine which of the regions shown in Figure 26d are relevant with respect to solutions that remain in the first quadrant for all $t > 0$. In eigenvector configuration (d), we have that $v_2 > 0$, $w_2 > 0$, since it is assumed that $v_1 = w_1 = 1$.

Consider an arbitrary intial condition in Region 1d shown in Figure 26d. All solutions originating in this region will be bounded by the eigenvectors v and w and will remain in the first quadrant for all $t > 0$.

Now consider an arbitrary intial condition, (x_0, y_0), in Region 2d shown in Figure 26d. It is sufficient to show that $\phi_1(t) > 0$ for all $t > 0$, which would imply that $\phi_2(t) > 0$ for all $t > 0$, given the position of the eigenvectors in this configuration. In Region 2d, $c_1 > 0$ and $c_2 < 0$. Since $c_1 > 0$ and $v_2 > 0$, Remarks (5) and (6) give the following:

$$\begin{aligned} c_1 e^{(\lambda_1 - \lambda_2)t} &> -c_2 \\ \Rightarrow c_1 e^{\lambda_1 t} &> -c_2 e^{\lambda_2 t} \\ \Rightarrow c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} &> 0 \end{aligned}$$

for all $t > 0$. This implies from equation (4.18) that $\phi_1(t) > 0$ for all $t > 0$.

Lastly, consider an arbitray initial condition, (x_0, y_0), in Region 3d shown in Figure 26d. We will show that $\phi_1(t) < 0$ for some $t > 0$, which will imply that solutions starting in Region 3d do not remain in the first quadrant. In Region 3d, $c_1 < 0$ and $c_2 > 0$. We determine when $\phi_1(t) < 0$ or equivalently, when the following inequality holds:

$$\begin{aligned}
c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} &< 0 \\
\Leftrightarrow c_1 e^{\lambda_1 t} &< -c_2 e^{\lambda_2 t} \\
\Leftrightarrow c_1 e^{(\lambda_1 - \lambda_2)t} &< -c_2 \\
\Leftrightarrow e^{(\lambda_1 - \lambda_2)t} &> \frac{-c_2}{c_1} \\
&\Leftrightarrow \ln e^{(\lambda_1 - \lambda_2)t} > \ln \left(\frac{-c_2}{c_1} \right) \\
&\Leftrightarrow t > \frac{\ln \left(\frac{-c_2}{c_1} \right)}{(\lambda_1 - \lambda_2)}.
\end{aligned}$$

Note that $\left(\frac{-c_2}{c_1} \right) > 0$ since $c_1 < 0$ and $c_2 > 0$ in this region. Hence, it has been shown that solutions starting in Region 3d do not remain in the first quadrant.

Thus, for eigenvector configuration (d), seen in Figure 38, there are two regions in which to consider initial conditions:

- **REGION 1d**: (x_0, y_0) in the first quadrant below the v eigenvector and above the w eigenvector
- **REGION 2d**: (x_0, y_0) in the first quadrant above both the eigenvectors

REGION 1d: Let (x_0, y_0) be in the first quadrant below the v eigenvector and above the w eigenvector in eigenvector configuration (d). The left panel of Figure 39 shows an arbitrary point in this region, with lines drawn (portions dashed), showing the addition of the two eigenvectors in the creation of the point (x_0, y_0). The right panel of Figure 39 shows the four regions formed in the first quadrant by the c_1-line and c_2-line. Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1d}$, $d_1 < c_1$ and $d_2 \leq c_2$. Since $d_2 \leq c_2$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{1d}$, when (x_0, y_0) is in Region 1d.

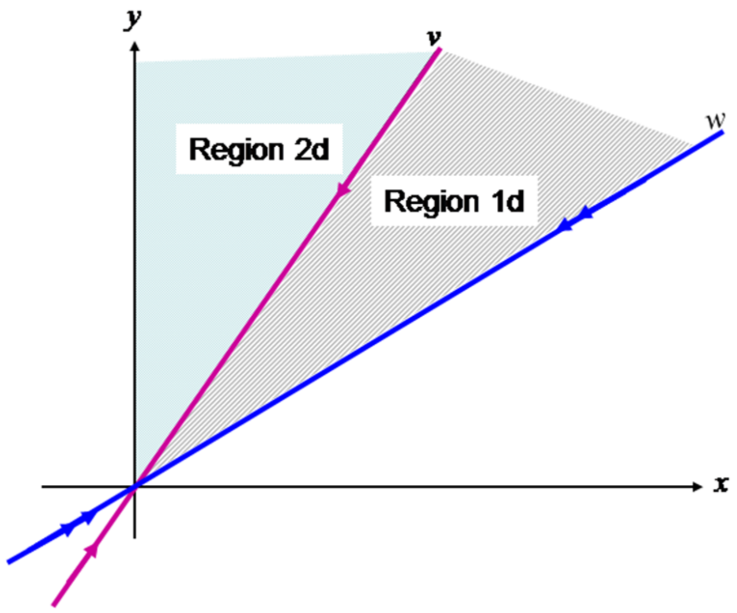


Figure 38: Eigenvector configuration (d) with the first quadrant divided into two regions where initial conditions yield solutions that begin and remain in the first quadrant.

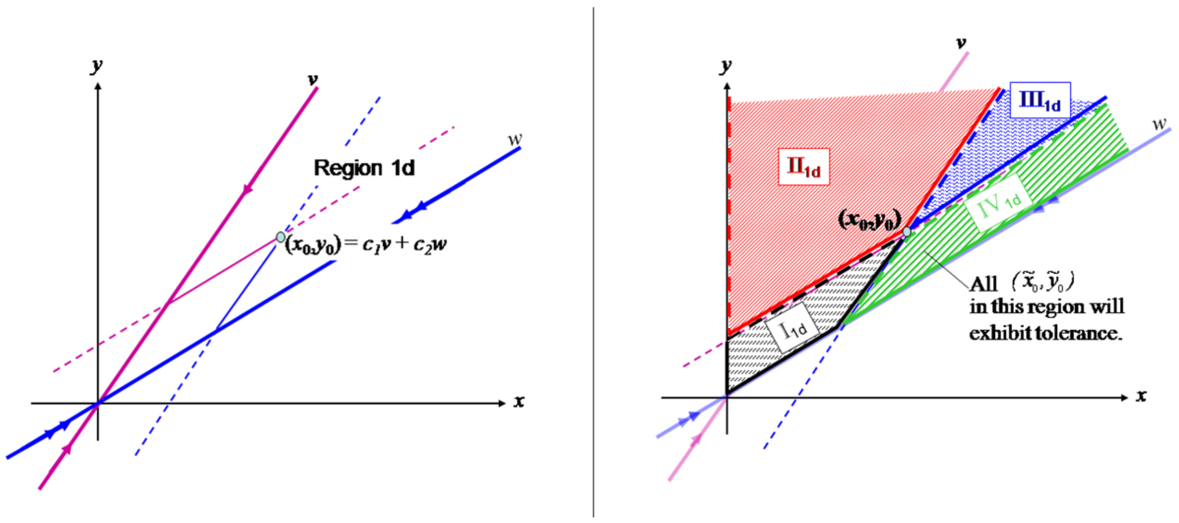


Figure 39: Left Panel: Eigenvector configuration (d) with an arbitrary initial condition (x_0, y_0) labeled in Region 1d along with lines drawn (portions dashed) showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (d) divided into two regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1 v + c_2 w$ lying in Region 1d.

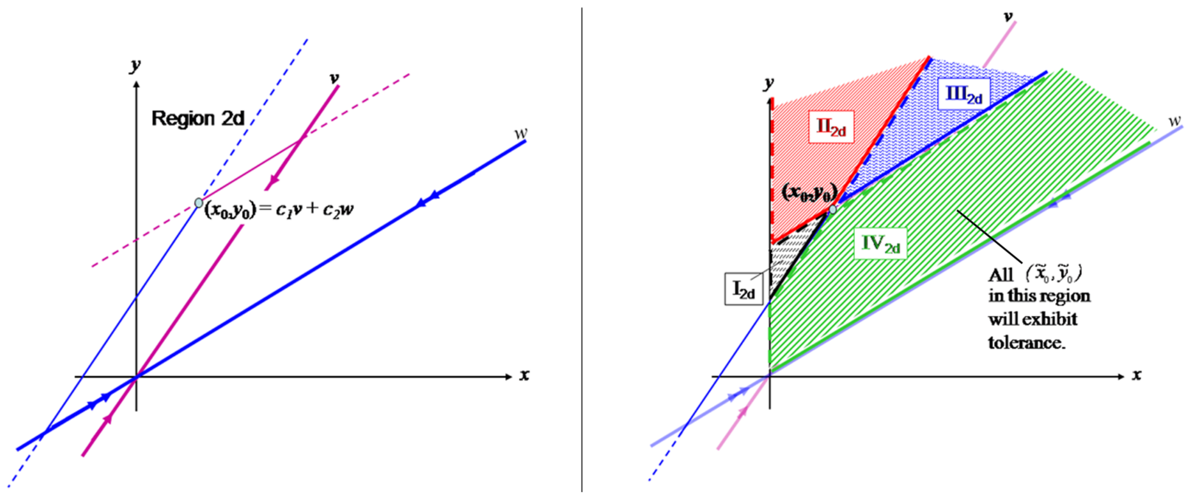

y
Region 2d
v
w
$(x_0, y_0) = c_1 v + c_2 w$
x
y
v
III$_{2d}$
II$_{2d}$
w
(x_0, y_0)
IV$_{2d}$
I$_{2d}$
All $(\tilde{x}_0, \tilde{y}_0)$ in this region will exhibit tolerance.
x

Figure 40: Left Panel: Eigenvector configuration (d) with an arbitrary initial condition (x_0, y_0) labeled in Region 2d along with lines drawn (portions dashed) showing the additon of the two eigenvectors in the creation of (x_0, y_0). Right Panel: The first quadrant of eigenvector configuration (d) divided into four regions by the c_1- and c_2-lines associated with the point $(x_0, y_0) = c_1 v + c_2 w$ lying in Region 2d.

2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{1d}$, $d_1 \geq c_1$ and $d_2 \leq c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{1d}$, when (x_0, y_0) is in Region 1d.

3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1d}$, $d_1 \geq c_1$ and $d_2 > c_2$. Since $d_1 \geq c_1$, tolerance cannot be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{1d}$, when (x_0, y_0) is in Region 1d.

4. **For** $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{1d}$, $d_1 < c_1$ **and** $d_2 > c_2$**. Since both** $d_1 < c_1$ **and** $d_2 > c_2$**, tolerance will be exhibited for** $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{1d}$**, when** (x_0, y_0) **is in Region 1d.**

Hence, for eigenvector configuration (d), if (x_0, y_0) is in the first quadrant below the v eigenvector and above the w eigenvector, then tolerance will be exhibited only when $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{1d}$, which is the green region shown in the right panel of Figure 39. In the Examples section (4.5.6), Example 6 shows an example of a trajectory, whose initial condition lies in Region 1d, and its corresponding $(\tilde{x}_0, \tilde{y}_0)$-curve, no points of which lie in the region marked $\mathbb{IV}_{1d}$.

REGION 2d: Let (x_0, y_0) be in the first quadrant above above both the eigenvectors in eigenvector configuration (d). The left panel of Figure 40 shows an arbitrary point in this region, with lines drawn (portions dashed), showing the addition of the two eigenvectors in the creation of the point (x_0, y_0). The right panel of Figure 40 shows the four regions formed in the first quadrant by the c_1-line and c_2-line. Using this and Theorems 19 and 20 we have:

1. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{2d}$, $d_1 < c_1$ and $d_2 \leq c_2$. Since $d_2 \leq c_2$ tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{I}_{2d}$, when (x_0, y_0) is in Region 2d.
2. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{2d}$, $d_1 \geq c_1$ and $d_2 \leq c_2$. Either of these relationships imply that tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{II}_{2d}$, when (x_0, y_0) is in Region 2d.
3. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{2d}$, $d_1 \geq c_1$ and $d_2 > c_2$. Since $d_1 \geq c_1$, tolerance cannot be exhibited for any $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{III}_{2d}$, when (x_0, y_0) is in Region 2d.
4. For $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2d}$, $d_1 < c_1$ and $d_2 > c_2$. **Since both $d_1 < c_1$ and $d_2 > c_2$, tolerance will be exhibited for $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2d}$, when (x_0, y_0) is in Region 2d.**

Thus, for eigenvector configuration (d), if $(\tilde{x}_0, \tilde{y}_0)$ is in the first quadrant above above both the eigenvectors, then tolerance will be exhibited only if $(\tilde{x}_0, \tilde{y}_0) \in \mathbb{IV}_{2d}$, which is the green region given in the right panel of Figure 40. In the Examples section (4.5.6), Example 7 shows an example of a trajectory, whose initial condition lies in Region 2d, and its corresponding $(\tilde{x}_0, \tilde{y}_0)$-curve, a portion of which lies in the region marked $\mathbb{IV}_{2d}$.

4.5.6 Examples

Example 1. *Eigenvector configuration (a) with* (x_0, y_0) *in Region 1a.*

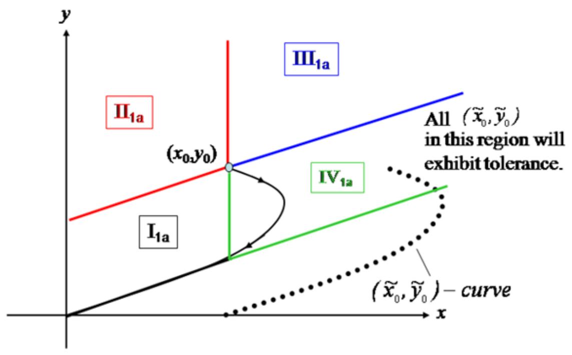

y
III_{1a}
II_{1a}
All $(\tilde{x}_0, \tilde{y}_0)$ in this region will exhibit tolerance.
(x_0, y_0)
IV_{1a}
I_{1a}
$(\tilde{x}_0, \tilde{y}_0)$ – curve
x

Figure 41: Example 1

Example 2. *Eigenvector configuration (b) with* (x_0, y_0) *in Region 2b.*

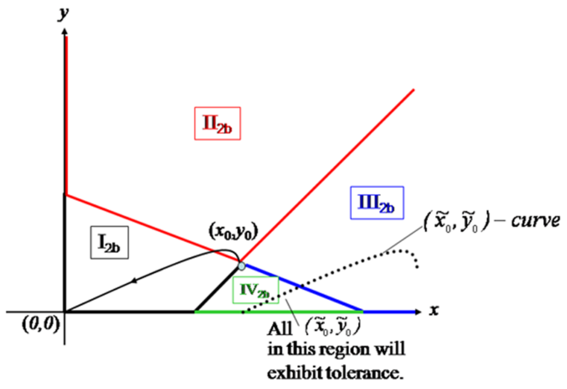


Figure 42: Example 2

Example 3. *Eigenvector configuration (b) with* (x_0, y_0) *in Region 3b.*

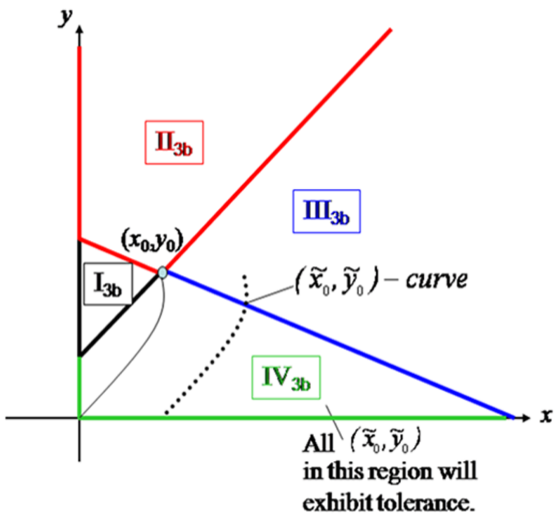


Figure 43: Example 3

Example 4. *Eigenvector configuration (c) with (x_0, y_0) in Region 2c.*

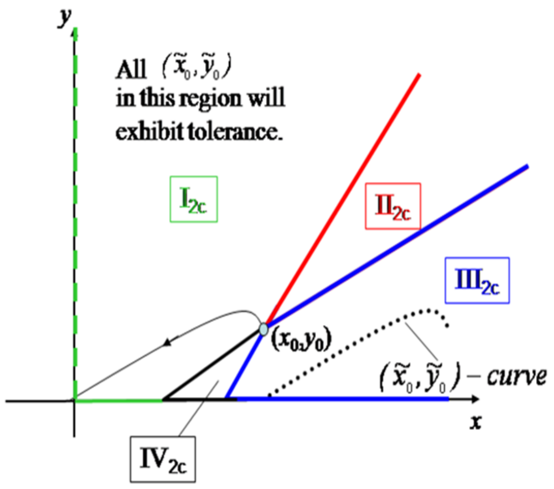


Figure 44: Example 4

Example 5. *Eigenvector configuration (c) with* (x_0, y_0) *in Region 3c.*

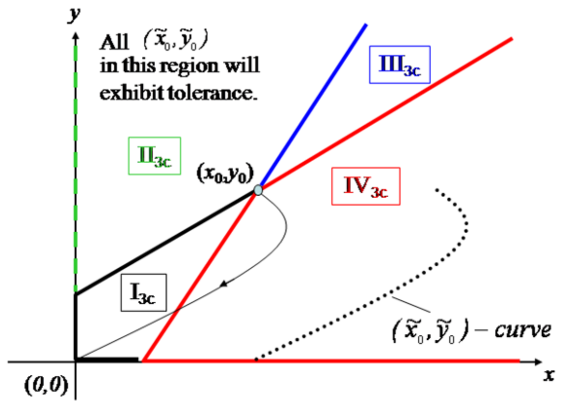
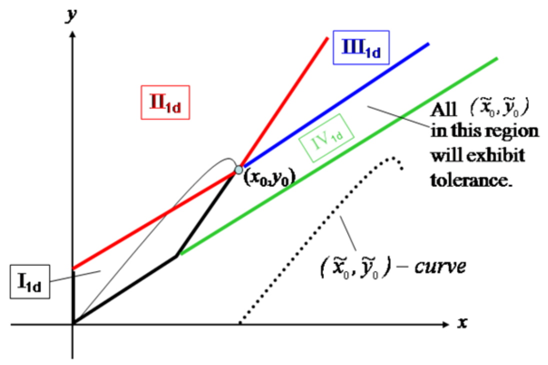

y
All $(\tilde{x}_0, \tilde{y}_0)$ in this region will exhibit tolerance.
III_{3c}
II_{3c}
(x_0, y_0)
IV_{3c}
I_{3c}
$(\tilde{x}_0, \tilde{y}_0)$ – curve
(0,0)
x

Figure 45: Example 5

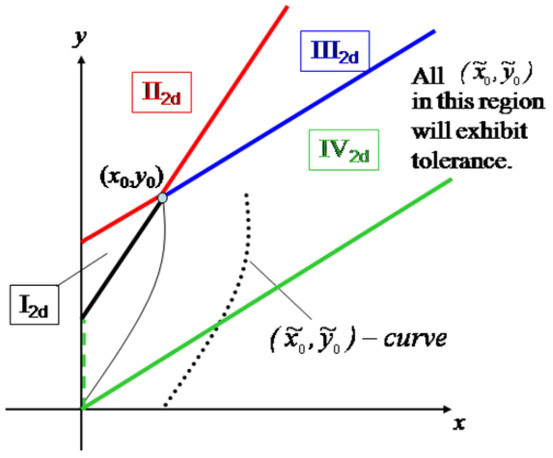
Example 6. *Eigenvector configuration (d) with* (x_0, y_0) *in Region 1d.*



y
III1d
II1d
All $(\tilde{x}_0, \tilde{y}_0)$ in this region will exhibit tolerance.
IV1d
(x_0, y_0)
I1d
$(\tilde{x}_0, \tilde{y}_0)$ – curve
x

Figure 46: Example 6

Example 7. *Eigenvector configuration (d) with* (x_0, y_0) *in Region 2d.*



y
II_{2d}
III_{2d}
All $(\tilde{x}_0, \tilde{y}_0)$ in this region will exhibit tolerance.
IV_{2d}
(x_0, y_0)
I_{2d}
$(\tilde{x}_0, \tilde{y}_0)$ – curve
x

Figure 47: Example 7

This concludes the results regarding 2D general linear ODE systems. The next section presents results for the existence of tolerance for 2D general *nonlinear* ODE systems.

4.6 TOLERANCE IN 2D GENERAL NONLINEAR ODE SYSTEMS

In Section 4.4, a few results were presented that gave some criteria that implied the existence of tolerance in system (4.1). These, however, were for very specific cases. For example, Proposition 8 considers an original trajectory, $\phi(t)$, that does not converge to the fixed point $(0,0)$, while the competing trajectory, $\psi(t)$, does converge to $(0,0)$, providing an obvious case when tolerance will be exhibited. Not as obvious, but still specific in their statements, are Propositions 9 and 10. In these propositions, the original trajectory, $\phi(t)$, and competing trajectory, $\psi(t)$, both converge to $(0,0)$, however, they are specifically written in the context of an initial condition of $\phi(t)$ that is an element of the positive x-axis, i.e. $(x_0, y_0) = (x_0, 0)$. The results, though interesting, are only for a small subset of the initial conditions that can be considered for $\phi(t)$. It would be nice to eliminate the restriction of only considering initial conditions that begin on the positive x-axis.

In the present chapter, additional tools are introduced that can be more widely applied to system (4.1) than the results of Section 4.4. In Section 4.6.1, we introduce the use of isoclines and the concept of inhibition as a method for determining in a general 2D ODE system if, for a given (x_0, y_0), there exists an $s \geq 0$ such that $(\tilde{x}_0, \tilde{y}_0) = \phi(s) + (x_0, 0)$ produces tolerance. Recall that for a given (x_0, y_0), there exists a continuous curve of points (parameterized by $s \geq 0$), denoted as the $(\tilde{x}_0, \tilde{y}_0)$-curve (see Figure 21) from which a competing trajectory, $\psi(t)$, can originate. Thus, for every $< (x_0, y_0), s >$, there is a corresponding $(\tilde{x}_0, \tilde{y}_0)$ point that might or might not produce tolerance. Using the isocline and inhibition strategy, portions of the $(\tilde{x}_0, \tilde{y}_0)$-curve will be identified as containing points that will not produce tolerance (i.e. $\psi(t) > \phi(t)$ for all $t \geq 0$) and points that *might* produce tolerance. The inhibition and isocline strategy is illustrated with several specific ODE examples.

Then, in Section 4.6.2, some numerical tools are presented, including an algorithm for numerically locating tolerance for a given initial condition, (x_0, y_0), or a set of initial conditions of the original trajectory, $\phi(t)$. In addition, this subsection also uses isoclines to acquire a numerical estimate for the time it take for a trajectory to, essentially, get from Point A to Point B. The idea is to estimate the time it takes for the original trajectory, $\phi(t)$, to go from (x_0, y_0) to some final point (x_f, y_f) and compare it to an estimate of the corresponding time for the competing trajectory, $\psi(t)$, to go from $(\tilde{x}_0, \tilde{y}_0)$ to (x_f, y_f). The use of isoclines in the Section 4.6.1, however, proves to be more insightful and functional.

4.6.1 Isoclines

Consider the ODE (4.1) and assume both (A1) and (A2) given previously and reproduced below:

(A1) $(0, 0)$ is a stable fixed point of (4.1) the eigenvalues of which are real and negative. (eliminates spirals and centers)

(A2) ϕ and ψ are nonnegative for all $t \geq 0$ and both (x_0, y_0) and $(\tilde{x}_0, \tilde{y}_0)$ lie in the basin of attraction for $(0, 0)$ in the first quadrant.

Definition 7. *The* x-isoclines *of (4.1) are the family of curves (or level sets) defined by* $f(x, y) = C_1 \in \mathbb{R}$.

Along an isocline, $f(x, y) = C_1$, the speed of the vector field in the x direction is given by C_1. (Similarly, for the y-isocline, $g(x, y) = C_2$, the speed in the y direction is given by C_2.) A nullcline, for instance, is an isocline in which $C = 0$, meaning that the velocity of the vector field along the curve is 0 in the particular direction.

Remark 7. *For convenience, we will drop the x- and just use isocline, since we do not consider the y-isoclines here.*

Before illustrating the use of isoclines for determining the existence of tolerance in general 2D ODE systems, we first introduce the concept of *inhibition*. Inhibition is a widely used term for describing the suppression of one variable by another. For instance, in the model used in Chapter 2, the anti-inflammatory mediator inhibited the production of pro-inflammatory mediators as well as its own production. However, the use of this term, in this and similar situations, while intuitive and heuristically understood, is not mathematically precise.

Hence, we give a precise definition of inhibition below, prove two results relating to inhibition and tolerance, and use these results, in conjunction with isoclines, in several examples of 2D nonlinear ODE systems, to show for which points of the $(\tilde{x}_0, \tilde{y}_0)$-curve tolerance can and cannot be produced. As a transition into using inhibition and isoclines in these examples, they are first illustrated using a linear example, specifically the type shown in Figure 32. Now, we precisely define the concept of inhibition:

Definition 8. *Define* inhibition *of x by y in a region, $R \subseteq \mathbb{R}^{2+}$, as the property of (4.1) that satisfies the following inequality: for $(x, y_1), (x, y_2) \in R$,*

$$f(x, y_1) > f(x, y_2) \text{ whenever } y_2 > y_1.$$

The next two propositions explore the relationship between inhibition and tolerance. First, another definition:

Definition 9. *The graph of ψ is said to be* bounded below *by the graph of ϕ if $\phi_2(s_1) < \psi_2(s_2)$, whenever $\phi_1(s_1) = \psi_1(s_2)$ for some $s_1, s_2 > 0$, not necessarily equal. For brevity, we say ψ is* bounded below *by ϕ.*

Theorem 21. *Assume (A1) and (A2) and that (4.1) exhibits tolerance for some $< (x_0, y_0), s >$. If the graph of ψ is bounded below by the graph of ϕ, then there exists a region of inhibition, R, and $s_1, s_2 \in \mathbb{R}^+$, such that $\psi_1(s_1) = \phi_1(s_2)$ and $\psi_1(s_1), \phi_1(s_2) \in R$.*

Proof. Assume tolerance exists for some $< (x_0, y_0), s >$. Now, assume by way of contradiction that y does not inhibit x in any region, R, which contains points $(\psi_1(s_1), \psi_2(s_1))$ and $(\phi_1(s_2), \phi_2(s_2))$, where $\psi_1(s_1) = \phi_1(s_2)$ and $s_1, s_2 \in \mathbb{R}^+$. Since tolerance exists for $< (x_0, y_0), s >$, by Proposition 1, there exists t^* such that $\psi_1(t^*) = \phi_1(t^*)$ and that $\psi_1(\hat{t}) < \phi_1(\hat{t})$ for all $\hat{t} \in (t^*, t^* + \delta)$ for some $\delta > 0$. Since the graph of ψ is bounded below by the graph of ϕ, we have that at t^*, $\psi_2(t^*) > \phi_2(t^*)$. Since $\psi_1(t^*) = \phi_1(t^*)$, our assumption implies that y does not inhibit x in any region R, containing the points $\psi(t^*)$ and $\phi(t^*)$. Thus, $f(\psi(t^*)) > f(\phi(t^*))$. However, from Proposition 1, since $\psi_1(t^*) = \phi_1(t^*)$ and $\psi_1(\hat{t}) < \phi_1(\hat{t})$ for all $\hat{t} \in (t^*, t^* + \delta)$, it can be concluded that $f(\psi(t^*)) \leq f(\phi(t^*))$, which is a contradiction. Hence, if the graph of ψ is bounded below by the graph of ϕ, then, in order for (4.1) to exhibit tolerance for $< (x_0, y_0), s >$, it is necessary that there exists a region, R, of inhibition and $s_1, s_2 \in \mathbb{R}^+$, such that $\psi_1(s_1) = \phi_1(s_2)$ and $\psi_1(s_1), \phi_1(s_2) \in R$. □

Theorem 21 states that a region of inhibition is necessary, although not sufficient for tolerance to occur when the competing trajectory, $\psi(t)$, is bounded *below* by the original trajectory, $\phi(t)$. However, for ψ bounded *above* by ϕ, inhibition can be a detriment to the presence of tolerance. In this case, if y inhibits x in a region, R, where $\phi(t)$, $\psi(t) \in R$, for all $t \geq 0$, then, as the next theorem states, tolerance cannot be exhibited. First, we define what it means for ψ to be *bounded above* by ϕ:

Definition 10. *The graph of ψ is said to be* bounded above *by the graph of ϕ if $\phi_2(s_1) > \psi_2(s_2)$, whenever $\phi_1(s_1) = \psi_1(s_2)$ for some $s_1, s_2 > 0$, not necessarily equal. For brevity, we say ψ is bounded above by ϕ.*

Theorem 22. *Assume (A1) and (A2). Given $< (x_0, y_0), s >$, if the graph of ψ is bounded above by the graph of ϕ, and y inhibits x in a region, R, such that $\phi(t)$, $\psi(t) \in R$ for all $t \geq 0$, then (4.1) cannot exhibit tolerance for $< (x_0, y_0), s >$.*

Proof. Assume by way of contradiction that (4.1) exhibits tolerance for some $< (x_0, y_0), s >$. Then, by Proposition 1, there exists t^* such that $\psi_1(t^*) = \phi_1(t^*)$ and that $\psi_1(\hat{t}) < \phi_1(\hat{t})$ for all $\hat{t} \in (t^*, t^* + \delta)$ for some $\delta > 0$. Also, $\phi_2(t^*) > \psi_2(t^*)$ since $\phi_1(t^*) = \psi_1(t^*)$ and the graph of ψ is bounded above by the graph of ϕ. Since y inhibits x in a region, R, where $\phi(t)$, $\psi(t) \in R$, for all $t \geq 0$, we have that $f(\phi(t^*)) < f(\psi(t^*))$. However, from Proposition 1, since $\psi_1(t^*) = \phi_1(t^*)$ and $\psi_1(\hat{t}) < \phi_1(\hat{t})$ for all $\hat{t} \in (t^*, t^* + \delta)$, it can be concluded that $f(\phi(t^*)) \geq f(\psi(t^*))$, which is a contradiction. Hence, it must be the case that (4.1) cannot exhibit tolerance for $< (x_0, y_0), s >$, if the graph of ψ is bounded above by the graph of ϕ and y inhibits x in a region, R, where $\phi(t)$, $\psi(t) \in R$, for all $t \geq 0$. $\square$

Furthermore, the contrapositive of Theorem 22 states that in order for tolerance to be a possibility for a competing trajectory, ψ, that is bounded above by the original trajectory, ϕ, there must exist at least one pair, $s_1, s_2 \in \mathbb{R}^+$, such that $\psi_1(s_1) = \phi_1(s_2)$ and $\psi_1(s_1), \phi_1(s_2) \notin R$, for any region of inhibition, R. An example of a linear system is given below, where it will be seen that for ψ bounded above by ϕ, the *absence* of a region of inhibition containing all of $\psi(t)$ and $\phi(t)$ makes tolerance possible, although not guaranteed, for ψ. On the other hand, for ψ bounded *below* by ϕ, it will be seen that the absence of a region of inhibition for any $s_1, s_2 \in \mathbb{R}^+$ such that $\psi_1(s_1) = \phi_1(s_2)$ eliminates the possibility of tolerance altogether for such ψ.

In the examples that are presented here, the isoclines are used to quickly locate any regions of inhibition. Then, Theorems 21 and 22 are used identify a subset of points from the $(\tilde{x}_0, \tilde{y}_0)$-curve which *might* produce tolerance. Example 8 will show that this set is larger than the actual set of $(\tilde{x}_0, \tilde{y}_0)$ points that do produce tolerance; however, the previous results of Section 4.4 can also be employed to hone in closer to the actual set of $(\tilde{x}_0, \tilde{y}_0)$ points that do produce tolerance. Now, we give the example of a linear system of the type shown in Figure 32:

Example 8. *Consider the linear system*

$$\left.\begin{array}{rcccl} \dot{x} & = & f(x,y) & = & -x+y \\ \dot{y} & = & g(x,y) & = & .3x-y \end{array}\right\} \tag{4.28}$$

The isoclines for this system are the family of curves given by $-x+y=C$, $C \in \mathbb{R}$, which are simply the lines $y = x + C$ parameterized by $C \in \mathbb{R}$, having slope 1 and y-intercept, C. In Figure 48, the isoclines are drawn for various values of $C \in [-2.4, 0.0]$, in increments of 0.1. Of course, since the isoclines are a continuous family of curves, they fill the entire space; however, viewing it as such is not particularly helpful. Note that as C ranges from -2.4 up to 0, the speed of the isoclines in the x-direction decreases monotonically going from right to left, toward the origin. Now we explain the features in Figure 49:

- $\phi(t)$ is the curve shown in solid black for initial condition $\phi(0) = (x_0, y_0) = (1, 0.25)$.
- The orange curve, denoted as $\hat{\phi}$, is the curve of points obtained by essentially integrating $\phi(t)$ in backward time from $t = 0$ to $t \approx -0.9$, at which time it intersects the x-axis at $\hat{x} \approx 2.19$. In other words:

$$\hat{\phi} \equiv \{\phi(-t) | t \in (0.0, 0.9) \text{ and } \phi(-0.9) = (2.19, 0.0)\}.$$

- The blue dotted curve denotes the $(\tilde{x}_0, \tilde{y}_0)$-curve. Formally, define this set of points as

$$P \equiv \{(\tilde{x}_0, \tilde{y}_0) | (\tilde{x}_0, \tilde{y}_0) = \phi(s) + (x_0, 0), s \geq 0\}.$$

- Let R_1 be the region shown in light green which is bounded inclusively by the black vertical line, $y = x_0$, the orange curve, $\hat{\phi}$, and the x-axis. Note that every point $(\tilde{x}_0, \tilde{y}_0)$ will lie to the right of the line $y = x_0$, by the definition of $\tilde{x}_0$.

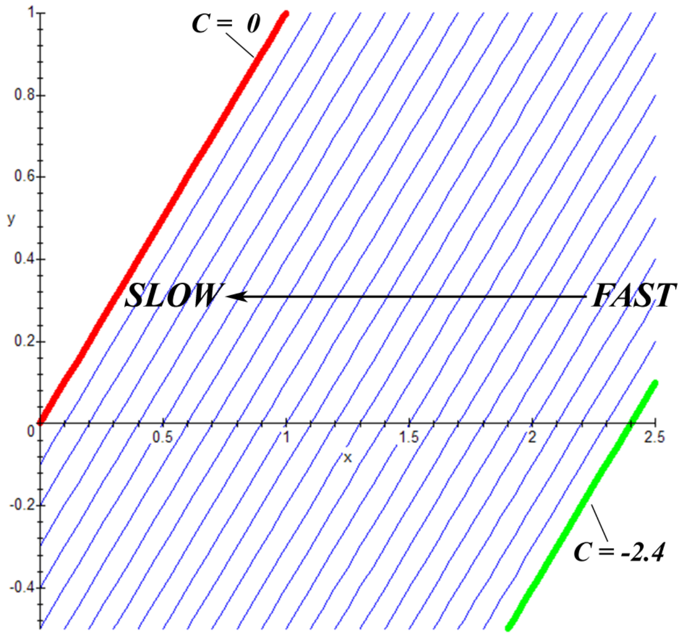


Figure 48: The isoclines for the linear system given in Example 8. The isoclines shown are for values of $C \in (-2.4, 0.0)$ in increments of 0.1. The red isocline is the x-nullcline for $C = 0$ and the green isocline is for the C-value of -2.4. Moving from one isocline through another from right to left, the speeds of the isoclines decrease in the x-direction. (i.e. the speed becomes less negative and closer to zero). Isoclines for $C > 0$ are not shown here since they do not need to be considered for this example.

- Define the set $S_{T_1} \neq \emptyset$ to be the intersection of the $(\tilde{x}_0, \tilde{y}_0)$-curve with the complement of R_1 in the first quadrant :

$$S_{T_1} \equiv \left(R^{2+} \backslash R_1\right) \cap P$$

- Define the set $S_{T_2} \neq \emptyset$ to be the intersection of the light green region, R_1, and the $(\tilde{x}_0, \tilde{y}_0)$-curve:

$$S_{T_2} \equiv R_1 \cap P$$

Remark 8. *The slow/weak eigenvector of this system is* $v = [1.0 \quad .548]$, *which can be viewed as the line passing through the origin,* $(0,0)$, *and the point,* $(1.0, .548)$*:* $y = 0.548x$. *The nullcline,* $y = x$ *has a steeper slope than that of the line representing* v. *Hence, solutions that originate below* v *will approach* $(0,0)$ *along* v *and not cross the nullcline* (*red isocline in Figure 49*). *This means that the isoclines for* $C \geq 0$ *do not need to be considered in the example.*

Consider the isoclines for $C \in [-2.4, 0.0)$. Also, assume $0 \leq x_1 < 2.4$, which encompasses the x-intercepts of the isoclines under consideration, since $x = -C$ when $y = 0$. Thus, given $x_1 \in (0.0, 2.4)$ and any two isoclines defined for $C_1, C_2 \in [-2.4, 0.0)$, whenever $C_1 < C_2 < 0$ (i.e. whenever C_1 is more negative than C_2) we have that

$$\begin{aligned} y_1 &\equiv x_1 + C_1 \\ &< x_1 + C_2 \\ &\equiv y_2. \end{aligned}$$

In other words, for any $x_1 \in (0.0, 2.4)$, whenever $y_1 < y_2$, then,

$$f(x_1, y_1) < f(x_1, y_2).$$

Hence, for any subset of the first quadrant, containing the trajectories of $\phi(t)$ and $\psi(t)$, there exist no regions of inhibition. (See Definition 8.) The sets S_{T_1} and S_{T_2} are formed so that for $(\tilde{x}_0, \tilde{y}_0) \in S_{T_1}$, ψ will be bounded below by ϕ, and for $(\tilde{x}_0, \tilde{y}_0) \in S_{T_2}$, ψ will be bounded above by ϕ. The the graph of $\hat{\phi}$, in orange, creates a natural boundary (by uniqueness of solutions) with which to divide the $(\tilde{x}_0, \tilde{y}_0)$-curve.

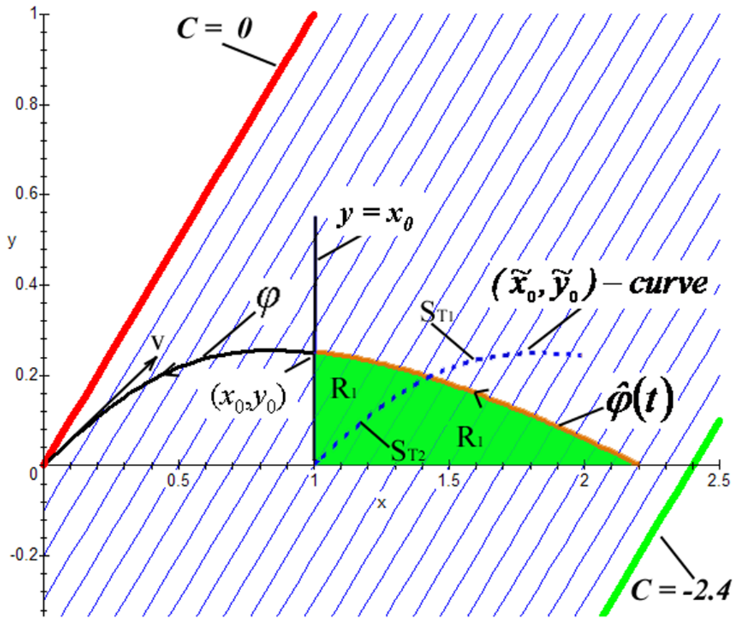


Figure 49: Two regions in which the points in the $(\tilde{x}_0, \tilde{y}_0)$-curve lie for Example 8. The points on the $(\tilde{x}_0, \tilde{y}_0)$-curve fall into one of two regions defined by (1) the light green area (including its borders) and (2) the complement of this area with respect to $\mathbb{R}^{2+}$.

Case 1: Let $(\tilde{x}_0, \tilde{y}_0) \in S_{T_1}$. Then, ψ will be bounded below by ϕ. Thus, by Theorem 21, since there are no regions of inhibition present for $\psi(t)$ and $\phi(t)$, any $(\tilde{x}_0, \tilde{y}_0) \in S_{T_1}$ cannot produce tolerance.

Case 2: Let $(\tilde{x}_0, \tilde{y}_0) \in S_{T_2}$. Then, ψ will be bounded above by ϕ. Thus, by the contrapositive of Theorem 22, since there are no regions of inhibition present for $\psi(t)$ and $\phi(t)$, $(\tilde{x}_0, \tilde{y}_0)$ might produce tolerance, although tolerance is not guaranteed. Furthermore, consider the $(\tilde{x}_0, \tilde{y}_0)$ point that lies on the orange curve, where the $(\tilde{x}_0, \tilde{y}_0)$-curve intersects $\hat{\phi}$. For this $(\tilde{x}_0, \tilde{y}_0)$, $\psi(t)$ and $\phi(t)$ are subsets of the same larger solution curve in the vector field of 4.28, and, in this particular example, both $\phi_1(t)$ and $\psi_1(t) \to (0, 0)$ monotonically as $t \to \infty$. By Proposition 7, this particular $(\tilde{x}_0, \tilde{y}_0)$ will not produce tolerance. In addition, by continuity, there exists an open ball, B, around $(\tilde{x}_0, \tilde{y}_0)$, such that $(\tilde{x}_b, \tilde{y}_b)$ will not produce tolerance for all $(\tilde{x}_b, \tilde{y}_b) \in B$. Thus, the set of points which might produce tolerance is a subset of S_{T_2}. As a result, the range of $(\tilde{x}_0, \tilde{y}_0)$ points that will possibly produce tolerance has been further narrowed.

Since this is an example of a linear system, the methods from Section 4.5 can be used to precisely pinpoint the set of $(\tilde{x}_0, \tilde{y}_0)$ points which are guaranteed to produce tolerance. In Figure 50, the regions $\mathbb{I} - \mathbb{IV}$, defining the relationship between the coefficients of the analytic solution of $\phi(t)$ and $\psi(t)$, are shown. According to the results of Section 4.5, tolerance is produced by those $(\tilde{x}_0, \tilde{y}_0)$ points in region $\mathbb{IV}$, where the only inclusive boundary is the x-axis. (See the case for Region 2a in Section 4.5.5 for more information.)

In Figure 51, the lines that form the regions $\mathbb{I} - \mathbb{IV}$ are overlaid on the content shown in Figure 49. It can be seen in Figure 51 that there is a collection of $(\tilde{x}_0, \tilde{y}_0)$ points which are elements of S_{T_2}, but which are not elements of Region $\mathbb{IV}$. Hence, this technique of using inhibition and isoclines to detect tolerance does not sharply define the subset of points on the $(\tilde{x}_0, \tilde{y}_0)$-curve which do produce tolerance. However, for this example, the estimation is not grossly far off from the actual set of points $(\tilde{x}_0, \tilde{y}_0)$ that do produce tolerance.

Now, we look at examples of specific 2D nonlinear ODE systems and apply this approach to determine when the possibility of tolerance exists.

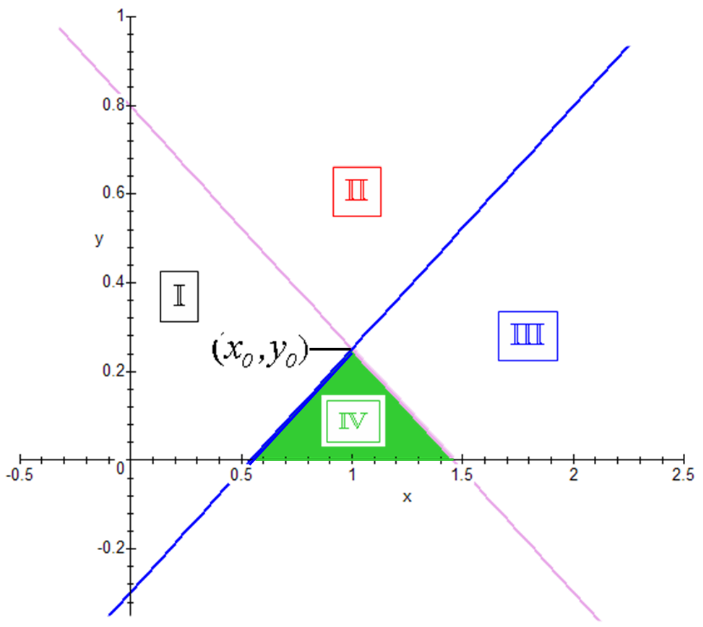


Figure 50: *Regions* $\mathbb{I}-\mathbb{IV}$ *defining the relationship between the coefficients of the analytic solution of* $\phi(t)$ *and* $\psi(t)$ for Example 8. *According to the results of Section 4.5, tolerance is produced by those* $(\tilde{x}_0, \tilde{y}_0)$ *points in region* $\mathbb{IV}$, *where the only inclusive boundary is the* x*-axis.*

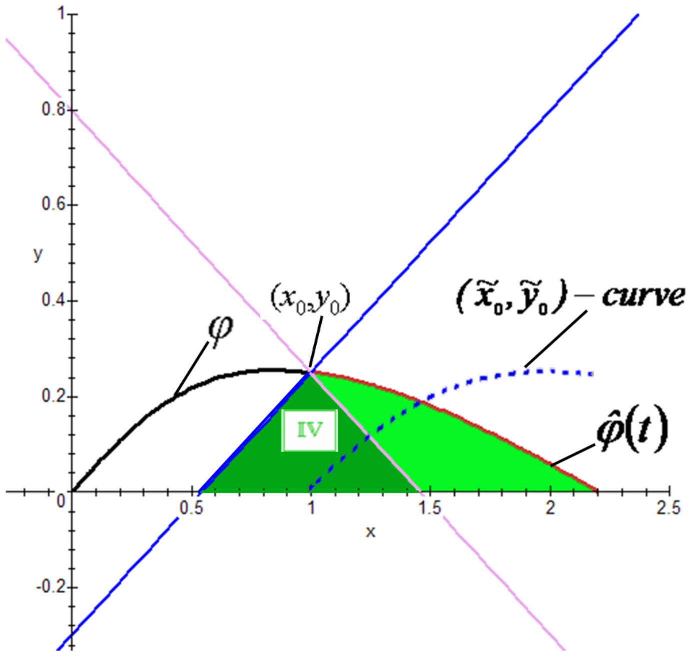


Figure 51: The lines that form the regions $\mathbb{I} - \mathbb{IV}$ are overlaid on the content shown in Figure 49. Specifically, for the $(\tilde{x}_0, \tilde{y}_0)$ points in region $\mathbb{IV}$ (dark green area), tolerance will be produced. In addition to these points, the inhibition and isocline method also included the $(\tilde{x}_0, \tilde{y}_0)$ points in the light green region as possible points that produce tolerance, but which do not.

Example 9. *Consider the nonlinear system:*

$$\left.\begin{array}{rcccc} \dot{x} & = & f(x,y) & = & (.5x-y)\left(\frac{0.1x}{1+y}-1\right) \\ \dot{y} & = & g(x,y) & = & 0.4x-y. \end{array}\right\}. \tag{4.29}$$

The isoclines for this system are the family of curves given by the equations:

$$y_1 = \frac{3}{10}x - \frac{1}{2} + \frac{1}{2}C + \frac{1}{10}\sqrt{4x^2+20x+30xC+25+50C+25C^2} \text{ and} \tag{4.30}$$

$$y_2 = \frac{3}{10}x - \frac{1}{2} + \frac{1}{2}C - \frac{1}{10}\sqrt{4x^2+20x+30xC+25+50C+25C^2}, \tag{4.31}$$

where $C \in R$. In Figure 52, the isoclines are drawn for various values of $C_1 \in [-1.5, 0.0]$, in increments of 0.05. For $C \in \mathbb{R}$, the two curves defined by equations 4.30-4.31 form a continuous parabola-like curve and the apparent discontinuity is simply due to numerical issues when graphing. However, it does nicely draw attention to the fact that the portion of the first quadrant containing the top curves of the parabolas given by equation 4.30 between the x-nullcline and the $C = -1.5$ isocline is not a region of inhibition. However, the portion of the first quadrant containing the bottom curves of the parabolas given by equation 4.31 between the x-nullcline and the $C = -1.5$ isocline is a region of inhibition.

Note that as C ranges from -1.5 up to 0, for the curves defined by equation 4.30, the speed of the isoclines in the x-direction decreases monotonically going from right to left, toward the origin. The value of (x_0, y_0) that we will be considering in this example will be such that $\phi(t)$ and $\psi(t)$ will only pass through these isoclines and not the other ones generated by equation 4.31. Figure 53 shows a specific solution, $\phi(t)$, that will be considered for this example and also only shows the portions of the isoclines that are of relevance here. As was done with Example 8, the following features are also a part of Figure 53:

- $\phi(t)$ is the curve shown in solid black for initial condition $\phi(0) = (x_0, y_0) = (2, 0.5)$.
- The orange curve, denoted as $\hat{\phi}$, is the curve of points obtained by essentially integrating $\phi(t)$ in backward time from $t = 0$ to $t \approx -0.85$, at which time it intersects the x-axis at $\hat{x} \approx 2.5$. In other words:

$$\hat{\phi} \equiv \{\phi(-t) | t \in (0.0, 0.85) \text{ and } \phi(-0.85) = (2.5, 0.0)\}.$$

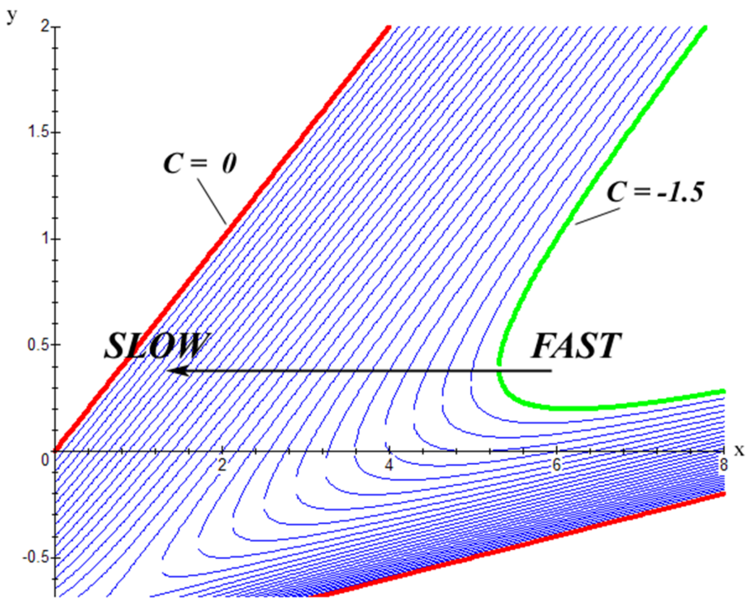


Figure 52: Isoclines for Example 9, drawn for various values of $C_1 \in (-1.5, 0.0)$, in increments of 0.05.

- The blue dotted curve denotes the $(\tilde{x}_0, \tilde{y}_0)$-curve. Formally, define this set of points as

$$P \equiv \{(\tilde{x}_0, \tilde{y}_0) | (\tilde{x}_0, \tilde{y}_0) = \phi(s) + (x_0, 0), s \geq 0\}$$

- Let R_1 be the region shown in light green which is bounded inclusively by the black vertical line $y = x_0$, the orange curve, $\hat{\phi}$, and the x-axis. Note that every point $(\tilde{x}_0, \tilde{y}_0)$ will lie to the right of the line $y = x_0$, by the definition of $\tilde{x}_0$.
- Define the set $S_{T_1} \neq \emptyset$ to be the intersection of the $(\tilde{x}_0, \tilde{y}_0)$-curve with the complement of R_1 in the first quadrant:

$$S_{T_1} \equiv \left(R^{2+} \backslash R_1\right) \cap P.$$

- Define the set $S_{T_2} \neq \emptyset$ to be the intersection of the light green region and the $(\tilde{x}_0, \tilde{y}_0)$-curve:

$$S_{T_2} \equiv R_1 \cap P.$$

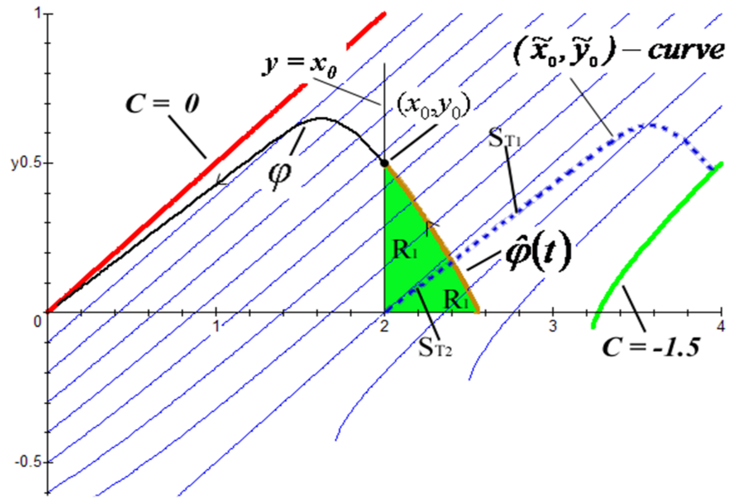


Figure 53: Two regions in which the points in the $(\tilde{x}_0, \tilde{y}_0)$-curve lie for Example 9. The points on the $(\tilde{x}_0, \tilde{y}_0)$-curve fall into one of two regions defined by (1) the light green area and (2) the complement of this area with respect to $\mathbb{R}^{2+}$. The light green area is inclusive of the orange curve, the x-axis, and the line $y = x_0$.

Remark 9. *In this example, because (1) $\phi(t)$, $\psi(t) \to (0,0)$ as $t \to \infty$ and (2) $\dot{x} > 0$ in the region of the first quadrant bounded by the y-axis and the $C = 0$ isocline (x-nullcline), neither $\phi(t)$ nor $\psi(t)$ cross the x-nullcline. Thus, the isoclines for $C \geq 0$ do not need to be considered in the example.*

Consider the isoclines for $C \in [-1.5, 0.0)$. Also, assume $0 \leq x_1 < 3.5$, which more than encompasses the x-intercepts of the isoclines under consideration. The orientation of the isoclines are very similar to that of the linear Example 8 in that for any subset of the first quadrant, containing the trajectories of $\phi(t)$ and $\psi(t)$, there exist no regions of inhibition. (See Definition 8.) The sets S_{T_1} and S_{T_2} are formed so that for $(\tilde{x}_0, \tilde{y}_0) \in S_{T_1}$, ψ will be bounded below by ϕ and for $(\tilde{x}_0, \tilde{y}_0) \in S_{T_2}$, ψ will be bounded above by ϕ. The the graph of $\hat{\phi}$, in orange, creates a natural boundary (by uniqueness of solutions) with which to divide the $(\tilde{x}_0, \tilde{y}_0)$-curve.

Case 1: Let $(\tilde{x}_0, \tilde{y}_0) \in S_{T_1}$. Then, ψ will be bounded below by ϕ. Thus, by Theorem 21, since there are no regions of inhibition present for $\psi(t)$ and $\phi(t)$, any $(\tilde{x}_0, \tilde{y}_0) \in S_{T_1}$ cannot produce tolerance.

Case 2: Let $(\tilde{x}_0, \tilde{y}_0) \in S_{T_2}$. Then, ψ will be bounded above by ϕ. Thus, by the contrapositive of Theorem 22, since there are no regions of inhibition present for $\psi(t)$ and $\phi(t)$, $(\tilde{x}_0, \tilde{y}_0)$ might produce tolerance, although tolerance is not guaranteed. Furthermore, consider the $(\tilde{x}_0, \tilde{y}_0)$ point that lies on orange curve, where the $(\tilde{x}_0, \tilde{y}_0)$-curve intersects $\hat{\phi}$. For this $(\tilde{x}_0, \tilde{y}_0)$, $\psi(t)$ and $\phi(t)$ are subsets of the same larger solution curve of the vector field 4.28, and, in this particular example, both $\phi_1(t)$ and $\psi_1(t) \to (0,0)$ monotonically as $t \to \infty$. By Proposition 7, this particular $(\tilde{x}_0, \tilde{y}_0)$ will not produce tolerance. In addition, by continuity, there exists an open ball, B, around $(\tilde{x}_0, \tilde{y}_0)$, such that $(\tilde{x}_b, \tilde{y}_b)$ will not produce tolerance for all $(\tilde{x}_b, \tilde{y}_b) \in B$. Thus, there exists an $S \subset S_{T_2}$, the points of which might produce tolerance. As a result, the range of $(\tilde{x}_0, \tilde{y}_0)$ points that will possibly produce tolerance has been further narrowed.

Although the methods from Section 4.5 cannot be applied to this nonlinear system, we can confirm our results by implementing a numerical approach discussed in the next section. An algorithm was generated to numerically find when tolerance occurs for a given (x_0, y_0). Figure 54 shows the results of the numerical findings, with $\phi_1(t)$ denoted by the red curve and $\psi_1(t)$ for various $\tilde{x}_0$ are denoted by the blue curves. The time τ at which a blue ψ_1-curve falls below the red ϕ_1-curve indicates that tolerance has occurred: $\psi_1(\tau) < \phi_1(\tau)$.

Note that all the $\tilde{x}_0$-values are all within the range $\tilde{x}_0 \in [2, 2.5)$ (See panel 3 of Figure 54), which correspond to points $(\tilde{x}_0, \tilde{y}_0)$ that are members of the set S_{T_2}, in which tolerance was a possibility. The first panel of Figure 54 shows a denser set of curves of $\psi_1(t)$ for $(\tilde{x}_0, \tilde{y}_0)$ points that are closer together. The second panel of this figure shows less curves of $\psi_1(t)$ for $(\tilde{x}_0, \tilde{y}_0)$ points spread further apart.

Example 10. *Using the system just given by 4.29 in Example 9, consider the initial condition for $\phi(t)$: $(x_0, y_0) = (2, 0)$. Figure 55, illustrates this example. There are two cases to consider:*

Case 1: $(\tilde{x}_0, \tilde{y}_0)$ lies on an isocline defined by equation 4.30, which defines the top portions of the parabolas, for $C \in [-1.5, 0]$. In this case, both the graphs of ϕ and ψ will be completely contained in this region of the first quadrant where the top portions of the parabolas lie. As mentioned in the previous example, this is not a region of inhibition.

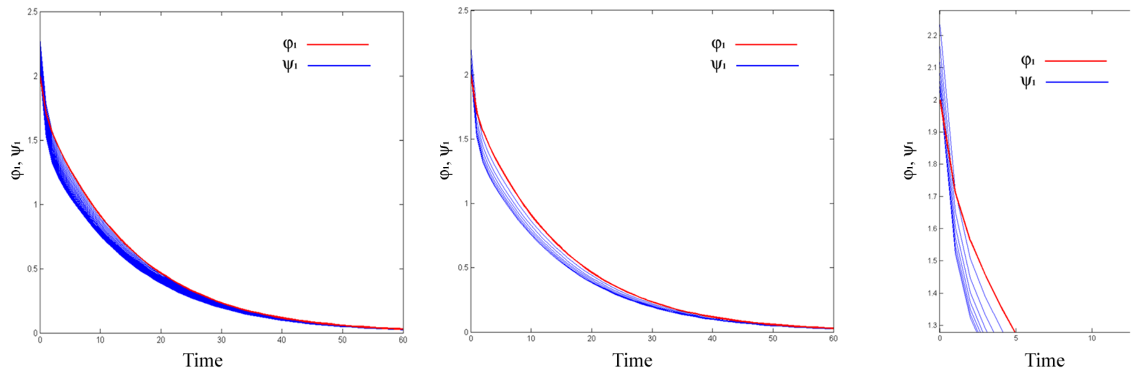


Figure 54: Numerical evidence of tolerance in Example 9. The red curve in all the panels is the time course of $\phi_1(t)$ for $(x_0, y_0) = (2.0, 0.5)$. First Panel: a dense set of curves (blue) of the time course of $\psi_1(t)$ for various $(\tilde{x}_0, \tilde{y}_0)$ points that are close together. Second Panel: a few curves (blue) of the time course of $\psi_1(t)$ for $(\tilde{x}_0, \tilde{y}_0)$ points that are spread further apart. Third Panel: A closeup of the $\tilde{x}_0$-values for the various time courses of $\psi_1(t)$ for $(\tilde{x}_0, \tilde{y}_0)$ points that are spread further apart. It can be seen that these (blue) time course where tolerance is evident have values of $\tilde{x}_0 \in [2, 2.5)$, corresponding to $(\tilde{x}_0, \tilde{y}_0)$ points in the set of points (S_{T_2}) for which tolerance is possible.

Furthermore, in this case $\psi(t)$ is bounded below by ϕ. Thus, by Theorem 21, $(\tilde{x}_0, \tilde{y}_0)$ cannot produce tolerance. Now for Case 2:

Case 2: $(\tilde{x}_0, \tilde{y}_0)$ lies on an isocline defined by equation 4.31, which defines the bottom portions of the parabolas, for $C \in [-1.5, 0]$.

This case is quite subtle for this example, but if (x_0, y_0) was further to the right on the x-axis, this becomes more of an issue. Nevertheless, however slight the occurrence, it still needs to be considered. In this case, $(\tilde{x}_0, \tilde{y}_0)$ does lie in a region of inhibition and portions of ψ also are contained in this region. However, no portion of the region contains ϕ, much less both ψ and ϕ. Thus, from Theorem 21, if these $(\tilde{x}_0, \tilde{y}_0)$ points produced tolerance than there would exist a region, R, of inhibition and $s_1, s_2 \in \mathbb{R}^+$, such that $\psi_1(s_1) = \phi_1(s_2)$ and $\psi_1(s_1), \phi_1(s_2) \in R$. However, such a region does not exist for this example. Thus, these $(\tilde{x}_0, \tilde{y}_0)$ points do not produce tolerance.

The algorithm used in the previous example was used here as well, and numerical calculations confirmed that tolerance was not exhibited in this system for $(x_0, y_0) = (2, 0)$.

The previous examples had similar isocline structures, with respect to the direction in which the speed of the isoclines increased/decreased. This final example looks at a system

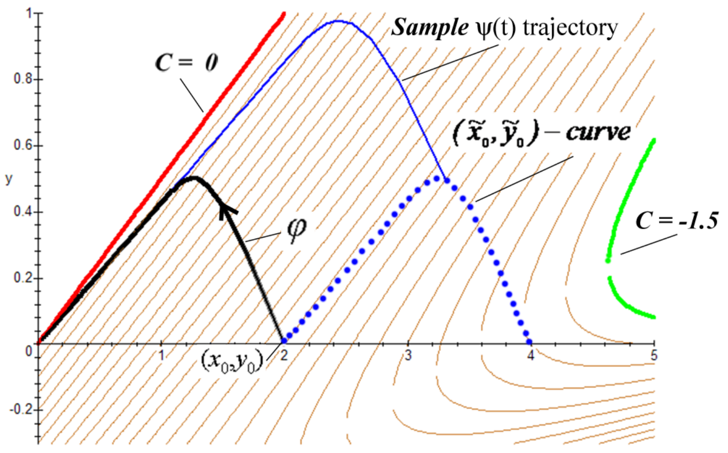


Figure 55: Important features of Example 10.

with a different isocline configuration.

Example 11. *Consider the system given by*

$$\left.\begin{aligned} \dot{x} &= f(x,y) &= \tfrac{x^2}{1+y} - x \\ \dot{y} &= g(x,y) &= x^2 - 0.5y \end{aligned}\right\}. \tag{4.32}$$

The isoclines for this system are the family of curves given by the equation:

$$y = \frac{x^2 - x - C}{x + C} \tag{4.33}$$

for $C \in \mathbb{R}$. Figure 56 shows the isoclines for $C \in [-6.0, 0]$ and denotes the change in speed from the $C = 0$ isocline (nullcline) to the $C = -6.0$ isocline. The isoclines for $C > 0$ are not drawn or considered here, since, in this region, the flow in the x-direction of the vector field is in the positive direction, and inhibition is defined for points at which the flow in the x-direction is negative. Note that for each $C \in [-6.0, 0)$, the $C = -x$ isocline is undefined at $x = -C$; however the asymptote is not drawn, and the other portion of the isocline which is on the other side of the asymptote below the positive x-axis is also not drawn. Note that if a trajectory is passing through a point with an x-value of $-C$ where the $C = -x$ isocline is approaching infinity, it does not mean that the speed of the trajectory at that point is

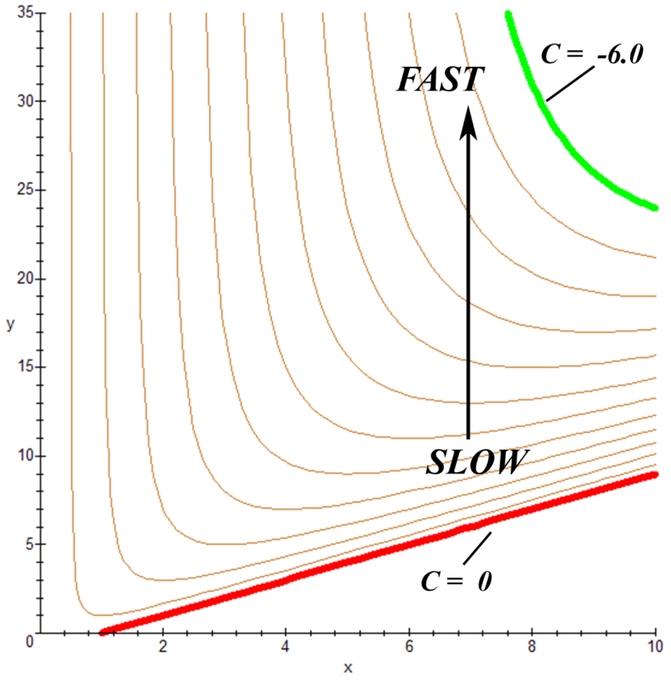


Figure 56: Isoclines for Example 11, drawn for various values of $C \in (-6.0, 0.0)$.

infinite. Instead, at that particular x-value, the trajectory is simply on a different isocline that *is* defined for $x = -C$.

Also note, that the sign of the slope of the isoclines in the limit as $x \to -C$ is

$$\begin{aligned} sign\left(\lim_{x \to -C} \frac{d}{dx}\left(\frac{x^2 - x - C}{x + C}\right)\right) &= sign\left(\frac{2x-1}{x+C} - \frac{x^2 - x - C}{(x+C)^2}\right) \\ &= -\left(sign(C)\right)^2 \end{aligned}$$

which is negative for all C. Define R_I to be the region bounded by (1) the x-nullcline, $y = x - 1$, (2) the $C = -6.0$ isocline, $y = \frac{x^2-x+6}{x-6}$, (3) the positive x-axis, and (4) the positive y-axis. Then, for $x_1 > 0$ and $y_1,\, y_2 \in R_I$, if $y_2 > y_1$, then

$$f(x_1, y_2) < f(x_1, y_1).$$

Hence, R_I is a region of inhibition. Now consider the initial condition $(x_0, y_0) = (4.0, 3.0)$ of $\phi(t)$. Figure 57 displays the following features:

- $\phi(t)$ is the curve shown in solid black for initial condition $\phi(0) = (x_0, y_0) = (4.0, 3.0)$.
- The orange curve, denoted as $\hat{\phi}$, is the curve of points obtained by essentially integrating $\phi(t)$ in backward time from $t = 0$ to $t \approx -1.0$, at which time it intersects the x-axis at $\hat{x} \approx 3.4$. In other words:

$$\hat{\phi} \equiv \left\{\phi(-t) | t \in (0.0, 1.0) \ and \ \phi(-1.0) = (3.4, 0.0)\right\}.$$

- The blue dotted curve denotes the $(\tilde{x}_0, \tilde{y}_0)$-curve. Formally, define this set of points as

$$P \equiv \{(\tilde{x}_0, \tilde{y}_0) | (\tilde{x}_0, \tilde{y}_0) = \phi(s) + (x_0, 0), s \geq 0\}$$

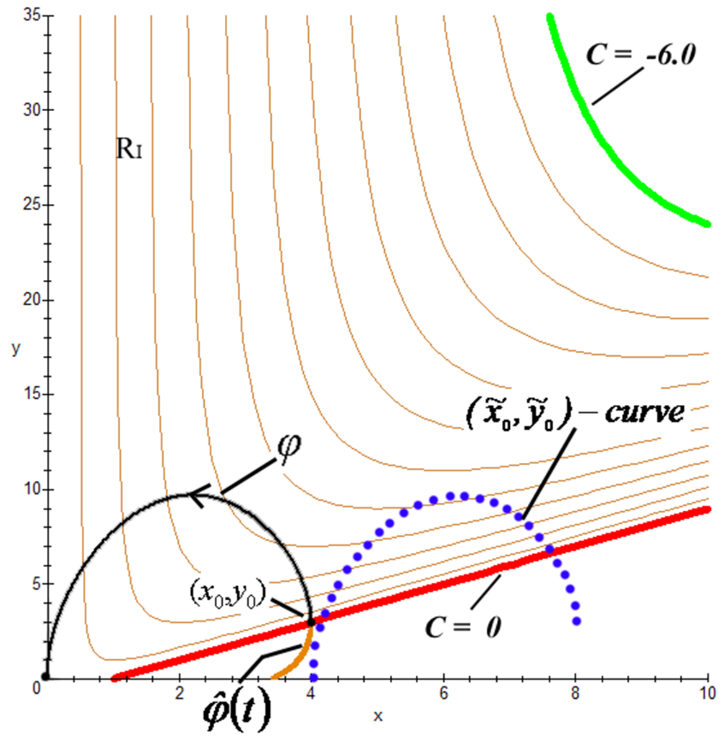


Figure 57: Isoclines and $(\tilde{x}_0, \tilde{y}_0)$ points for Example 11.

For this example, the curve $\hat{\phi}(t)$ (orange curve in Figure 57) shows that for every $(\tilde{x}_0, \tilde{y}_0) \in P$, the corresponding graph of ψ will eventually be bounded below by the graph of ϕ since (1) ψ will not be allowed to cross the trajectory defined as $\hat{\phi} \cup \phi$ and (2) $\phi, \psi \to (0,0)$. Thus, for all the $(\tilde{x}_0, \tilde{y}_0) \in P$ not in region R_I, $\psi(t)$ must enter this region eventually. To reiterate, the graph ψ will be bounded below by the graph of ϕ when $\psi_1(\tau) = \phi_1(0) = \max(\phi_1)$ for some $\tau > 0$.

Now consider the various ψ trajectories shown in Figure 58 for various values of $(\tilde{x}_0, \tilde{y}_0)$. For each ψ, there exists a $\tau > 0$, after which the graph of ψ is bounded below by the graph of ϕ. All of the ψ trajectories enter the region R_I and $\phi \subset R_I$. Hence, after $\tau > 0$, there exists a region $\bar{R} \subset R_I$ and $s_1, s_2 \in R^+$ (not necessarily equal), such that $\psi_1(s_1) = \phi_1(s_2)$ and $\psi_1(s_1), \phi_1(s_2) \in \bar{R}$. Hence, by Theorem 21, it is possible that tolerance can be exhibited by any $(\tilde{x}_0, \tilde{y}_0) \in P$, although not guaranteed since the existence of this inhibition region is only a necessary condition for the existence of tolerance.

Running the numerical tolerance algorithm on this example for initial condition $(x_0, y_0) = (4.0, 3.0)$, we see that tolerance does exist. However, the reduction noted is quite small when it does occur. Figure 59 shows results from the numerical simulations. The left panel gives the time courses for $\phi_1(t)$ (red curve) and $\psi_1(t)$ for the various $(\tilde{x}_0, \tilde{y}_0)$ points (blue curves) which produce tolerance. Note the $\tilde{x}_0$-values for these points, given on the "y-" axis of Figure 59. Then, look at where the corresponding $(\tilde{x}_0, \tilde{y}_0)$ points are in Figure 58. The right panel gives a close up of the portion of the left panel, showing that the time courses for the various blue ψ_1-curves do indeed fall below the red ϕ_1-curve, thereby confirming the existence of tolerance.

The next section discusses two numerical approaches for finding tolerance, one of which was used above in confirming the existence (or lack thereof) tolerance.

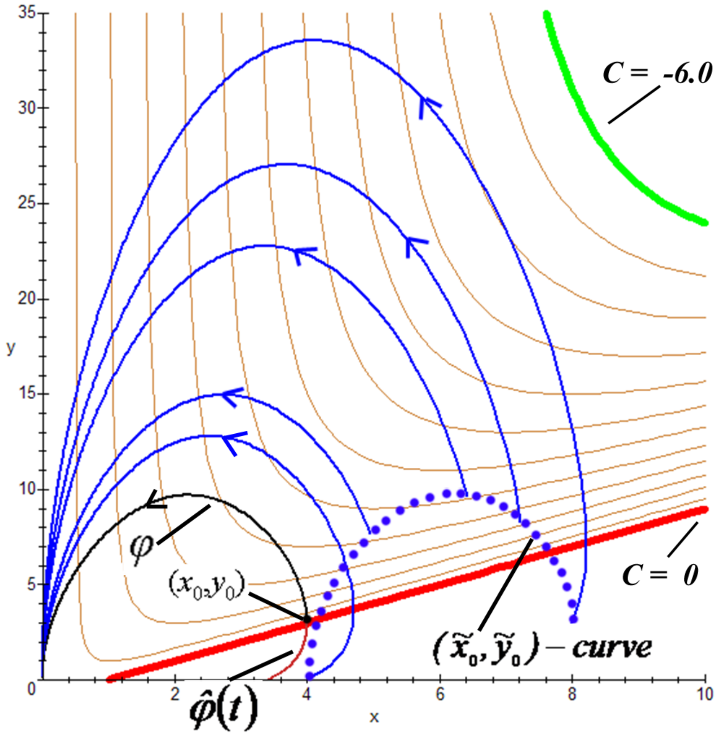


Figure 58: The isoclines and various examples of trajectories, $\psi(t)$, originating from the $(\tilde{x}_0, \tilde{y}_0)$-curve of system 4.32 of Example 11 for $(x_0, y_0) = (4.0, 3.0)$.

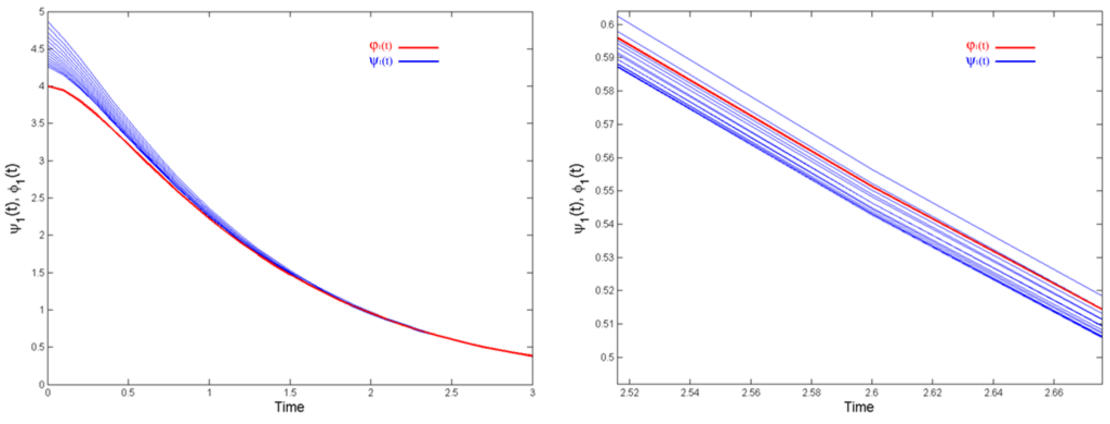


Figure 59: Numerical results from the application of the tolerance algorithm to Example 11. Left Panel: Time courses for $\phi_1(t)$ (red curve) and $\psi_1(t)$ for the various $(\tilde{x}_0, \tilde{y}_0)$ points which produce tolerance. Right Panel: A close up of the portion of the left panel, showing that the time courses shown for the various blue ψ_1-curves do indeed fall below the red ϕ_1-curve, thereby implying the existence of tolerance.

4.6.2 Numerical Approaches for Finding Tolerance

The primary difficulty in determining the presence of tolerance in a given ODE system is that closed form solutions of the system are usually not available and hence, the ability to determine the amount of time that it takes for a trajectory to traverse from its initial condition to another point on its path cannot be calculated explicitly. In this section we discuss the numerical algorithm for finding tolerance in a given system for a specified initial condition or a set of initial conditions. Also discussed is a method that uses isoclines to determine tolerance. This method directly deals with estimating the time it takes for a trajectory to reach a certain point, starting from a given initial condition.

4.6.2.1 Tolerance Algorithm A purely numerical approach to identifying the existence of tolerance in a given dynamical system has been to create an algorithm that tests the system for tolerance, given several input parameters. The algorithm is not restricted to two dimensional systems, so for now this is the only method for finding tolerance in systems of dimension greater than two. This algorithm was written as an m-file in MatLab. [72] First, the input that the algorithm requires is explained.

INPUT

- `ODEfile`: MatLab m-file giving the ODE system (2-d, 3-d, ..., n-d) with equations ordered so that the tolerance variable is first
- `X0`: Matrix whose rows contain the various initial condition vectors, x_0, for an original trajectory
- `HitIndex`: Column index (for all the x_0) in the `X0` matrix to which the hit is administered (note: could be different from the Tolerance Variable.)
- `HitSize`: size of hit; default value is the value of the variable found in the `HitIndex`th entry of the row currently being considered in the `X0` matrix.
- `TolVarIndex`: Column index (for all the x_0) in the `X0` matrix for which tolerance is being checked. This is the "Tolerance Variable."
- `MinTolPercent`: minimum amount of reduction accepted (in percentage form)
- `tstop`: Duration of integration

- `s`: vector of time points at which the hit is given; This also serves as the sequence of times at which solutions are compared.
- `JStepSize`: step size used to divide up the interval [`0 tstop`] to create the vector, s

There are a number of practical issues to consider with respect to the presence of tolerance. Since the curve of $(\tilde{x}_0, \tilde{y}_0)$ points is a continuous curve parameterized by $s > 0$, numerically, it is impossible to test every $(\tilde{x}_0, \tilde{y}_0)$ to see if tolerance is produced. However, it is important to try a sampling of s-values that range from $s = 0$ to a later time point corresponding to $\phi(s) \approx (0, 0)$. Thus, a step size, `JStepSize`, is specified to divide this interval of times, denoted [`0 tstop`], into a discrete set of points, where tolerance can be checked. This is sufficient to ensure that the entire $(\tilde{x}_0, \tilde{y}_0)$-curve is represented well enough.

Another practical issue has to do with the *degree* of tolerance that is exhibited. Recall in Chapter 2, when we noted that in some of the endotoxin tolerance simulations, there was a percentage decrease in the pro-inflammatory mediator, sometimes 40% or 70% of the original amount at a particular time point. Again, because of continuity, the amount of tolerance that is exhibited can be arbitrarily small. Hence, in order to numerically address this, the minimum percentage amount of reduction accepted must be specified. This is done by specifying a value for the variable, `MinTolPercent`. For Example 11 in the last section, the `MinTolPercent` was set to 1% and for each of the blue curves shown, all had a maximum reduction of less than 2%. Hence, if the `MinTolPercent` was given, for example, a value of `2`, then the algorithm would have returned:

```
"There was no reduction with a percentage greater than the minimum tolerance percent."
```

The algorithm is set up to consider a number of options beyond what was presented in this chapter. For example, recall that $(\tilde{x}_0, \tilde{y}_0)$ is a shifting of the point $\phi(s)$, $s > 0$, by the amount x_0. This was assumed to be the case throughout every section. However, it might be the case that the hit sizes are different as they were in the endotoxin tolerance experiments of Chapter 2. Thus, the algorithm allows for the separate specification of the variable in which tolerance is being checked and the variable to which the extra "hit" amount is given.

A host of data can be returned by the algorithm. Note that the equation for the variable in which tolerance is being checked is always the first equation in the ODE file. This simplifies the discussion and makes the code simpler. As it stands now, the following information is returned in a table format, if and when the algorithm finds tolerance:

- Row number for the initial condition in X_0 for which tolerance occured. Since X_0 is a matrix, the rows of which contain various initial condition vectors (e.g. (x_0, y_0)) from which to check for tolerance, it is important to know to which initial condition the presence of tolerance refers.
- Jump Time. Recall that s was defined to be the "*jump time*" – the time for which an $(\tilde{x}_0, \tilde{y}_0)$ is defined.
- Time when maximum degree of tolerance occurred. i.e. the time τ at which $\psi_1(\tau)-\phi_1(\tau)$ was greatest.
- First time when tolerance occurred. Again, because numerically it is virtually impossible to find the "first" time, τ^*, for which $\psi_1(t) < \phi_1(t)$, this is an estimate.
- Value of $\phi_1(t)$ at the time when the maximum degree of tolerance occurred.
- Value of $\psi_1(t)$ at the time when the maximum degree of tolerance occurred.
- The amount (as a percentage) of the maximum degree of tolerance between $\phi_1(t)$ and $\psi_1(t)$.
- Graph of time courses of $\phi_1(t)$ and corresponding $\psi_1(t)$, for every initial condition given in X_0 matrix (This is all put in one figure, which can get cumbersome, but the graphing feature was intended for displaying the results from running the algorithm on one initial condition. The table format displaying output is better suited for a multiple initial condition run.)

In order to make sure that the presence of tolerance is not being missed, it is important to carefully consider the amount of integration time, since this essentially determines which $(\tilde{x}_0, \tilde{y}_0)$ points are checked. As $s \to \infty$, $\phi(s) \to (0, 0)$ and $(\tilde{x}_0, \tilde{y}_0) \to \phi(0)$. However, if the integration time is too short, then, for the last specified s value, $\phi(s)$ many not yet be close enough to $(0, 0)$ and $(\tilde{x}_0, \tilde{y}_0) \approx \phi(0)$ may not be included in the points under consideration. In addition, a reasonable array of s values should be considered, i.e. `JStepSize` should not be too big. However, if tolerance is produced by many $(\tilde{x}_0, \tilde{y}_0)$ points, then `JStepSize` can be reduced so that more of the individual time courses $\psi_1(t)$ can be seen on the graph that is generated. As a good example to compare results, Figure 60 shows the results of running the algorithm on the linear system 4.28 given in Example 8 for the initial condition $(x_0, y_0) = (1, 0.25)$.

4.6.2.2 Isoclines Another approach deals with using isoclines to estimate the time it takes for a trajectory to get from point A to point B. We pose this problem for the 2D system (4.1). The main idea of this approach is to use isoclines to estimate the speed of a trajectory in a certain direction (or variable) and then use the estimate to determine the amount of time it takes for the trajectory to reach a certain value in the variable. Since the speed of a trajectory is time dependent we will look for upper bounds and/or lower bounds on speed in a particular direction. For simplicity, assume the direction we are interested in is the first component of the system and call it the *tolerance variable.*

We compare the speed estimates for two trajectories, one which we call the original trajectory since all other trajectories will be compared to it, and the other trajectory we call the competing trajectory, which comes from a family of trajectories defined from the original trajectory. We include this approach under numerical approaches since many times isoclines are too difficult to analyze analytically and one must numerically graph the isoclines and trajectories to estimate the supremum and infimum that are needed below. In addition, we restrict this to a two dimensional case, since isoclines become isoplanes in dimensions greater than two, and beyond the third dimension, this approach breaks down completely.

As done in previous work, consider two trajectories: $\phi(t) \equiv (x_0, y_0) \cdot t$ for some (x_0, y_0) and for $t \geq 0$, which we label as the original trajectory, and $\psi(t) \equiv (\tilde{x}_0, \tilde{y}_0) \cdot t$, where $(\tilde{x}_0, \tilde{y}_0) \equiv \phi(s_0) + (x_0, 0)$, for some $s_0 \geq 0$ and where $t \geq 0$, which is the competing trajectory. Note that by taking different values of s_0, we form a family of competing trajectories. Let x_f be a given x-value. Also, we will assume that ϕ and ψ remain in the first quadrant for all time and that as $t \to \infty$, both approach $(0, 0)$. We wish to estimate the time it takes for $\phi_1(t)$ and $\psi_1(t)$ to travel from x_0 and $\tilde{x}_0$, respectively, to x_f and then determine conditions under which $\psi_1(t)$ will arrive at x_f before $\phi_1(t)$. Assume that $x_f < x_0 < \tilde{x}_0$.

Consider the family of isoclines $f(x, y) = C$, where $C \in \mathbb{R}$. Each C-value represents the

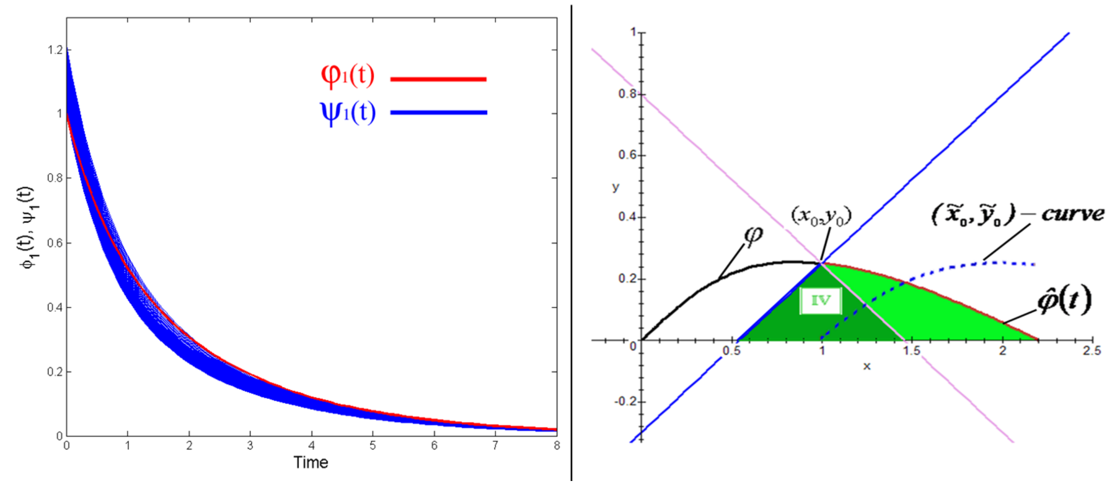


Figure 60: Numerical results showing the presence of tolerance for a host of different $(\tilde{x}_0, \tilde{y}_0)$ points for the system 4.28 given in Example 8 compared to the other methods for pinpointing tolerance. The initial condition for $\phi(t)$ is $(x_0, y_0) = (1, 0.25)$. Left Panel: The red curve is the time course for $\phi_1(t)$. The various blue time courses of $\psi_1(t)$ for different $(\tilde{x}_0, \tilde{y}_0)$ points. The $\tilde{x}_0$-value associated with each of the $(\tilde{x}_0, \tilde{y}_0)$ points can be clearly seen. Right Panel: This figure is reproduced from Example 8, showing both the results from the linear methods from section 4.5.5 and the isocline method from Section 4.6.1. Compare the $\tilde{x}_0$-values for the $\psi_1(t)$ curves on the left to the $\tilde{x}_0$-values of the $(\tilde{x}_0, \tilde{y}_0)$ points that fall in the dark green region shown on the right panel.

speed of $\dot{x}$ along each isocline $f(x,y) = C$. Let $\underline{c}_\phi = inf_{t\in[0,t_\phi)}\{\dot{x}(\phi_1(t),\phi_2(t))\}$, where $[0,t_\phi)$ is such that $\phi_1(0) = x_0$ and $\phi_1(t_\phi) = x_f$, so that t_ϕ is defined to be the time it takes for $\phi_1(t)$ to go from x_0 to x_f. Also, let $\overline{c}_\psi = sup_{t\in[0,t_\psi)}\{\dot{x}(\psi_1(t),\psi_2(t))\}$, where $[0,t_\psi)$ is such that $\psi_1(0) = \tilde{x}_0$ and $\psi_1(t_\psi) = x_f$, so that t_ψ is the time it takes for $\psi_1(t)$ to go from $\tilde{x}_0$ to x_f. Assume $\underline{c}_\phi < 0$ and $\overline{c}_\psi < 0$ which makes $\underline{c}_\phi$ the smallest or most negative isocline that the trajectory $\phi(t)$ passes through. Likewise, $\overline{c}_\psi$ is the largest or least negative isocline that the trajectory $\psi(t)$ passes through. Let

$$\begin{aligned} d_\phi &\equiv \phi_1(0) - \phi_1(t_\phi) \\ &\text{and} \\ d_\psi &\equiv \psi_1(0) - \psi_1(t_\psi) \end{aligned}$$

If C_ϕ and C_ψ are the actual speeds (both dependent on time) of $\phi(t)$ and $\psi(t)$, respectively, then we know that

$$\begin{aligned} d_\phi &= |C_\phi(t)|t_\phi \\ &\text{and} \\ d_\psi &= |C_\psi(t)|t_\psi \end{aligned}$$

since distance is the product of speed and time. Using this along with $|\underline{c}_\phi|$ and $|\overline{c}_\psi|$, we have the following inequalities:

$$\begin{aligned} d_\phi &\leq |\underline{c}_\phi|t_\phi \\ &\text{and} \\ d_\psi &\geq |\overline{c}_\psi|t_\psi \end{aligned}$$

Thus, if

$$(t_\phi \geq)\frac{d_\phi}{|\underline{v}_\phi|} > \frac{d_\psi}{|\overline{c}_\psi|}(\geq t_\psi) \text{ then } t_\psi < t_\phi$$

and this implies tolerance. Although this condition is sufficient for tolerance, it is also quite restrictive because it underestimates t_ψ and overestimates t_ϕ. In addition, the method is tedious in practice since a separate estimate for each competing trajectory would have to be calculated. Also, since we assumed that $\underline{c}_\phi < 0$ and $\overline{c}_\psi < 0$ then $\dot{x} < 0$ at x_0 and $\tilde{x}_0$ and all along the respective trajectories. The case where $\dot{x} > 0$ at x_0 can be approached in a similar way by estimating the speed of the original trajectory after it intersects the $\dot{x}$ nullcline and $\dot{x} < 0$ ensues. This again will overestimate t_ϕ, even more so than the other case.

5.0 USING NONLINEAR MODEL PREDICTIVE CONTROL TO FIND OPTIMAL THERAPY STRATEGIES TO MODULATE INFLAMMATION

5.1 INTRODUCTION

Controlling inflammation has become a key focal point in the treatment of critically ill patients. Much of the theoretical work regarding this has been with the objective to unravel the inner workings of systemic inflammation and understand how the mediators interact with one another with respect to their different time scales. [20, 112, 65] Significant insight has been acquired from these approaches and implications have been made regarding types of treatment that may be effective against persistent inflammation. A main result coming from this research confirms that the timing of events, such as the production and decay of both pro- and anti-inflammatory mediators, is critical to finding and implementing appropriate therapies [28, 98].

When the issue of controlling inflammation was initially pursued, the approach was to target a sole inflammatory mediator. It is now known, however, that there is no one mediator which stands as the source for persistent inflammation [13, 71, 15]. Instead, a cascade of inflammation occurs, which is, perhaps, started by a few key mediators but persists as a result of a complicated feedback process involving mediators that are produced later than the initial inflammatory instigator. In addition, anti-inflammatory mediators may be present in elevated levels during prolonged inflammation, but their effect on the pro-inflammatory mediators may be small or negligible due to the relative amounts of inflammation present in the system.

There is still much to be done in the area of identifying proper biological targets in order to develop therapies to combat excessive and pervasive inflammation. However, equally necessary is determining a strategy for delivering therapies, in the correct amount, at the right time. One of the tools that can help determine this complex dose regimen is Nonlinear Model Predictive Control (NMPC). This area of research has mainly been applied to industrial operations involving industrial systems that can be well described with a mathematical model, usually a system of ordinary differential equations.

The advantage of this procedure over other control algorithms is its ability to (1) predict the real system state at a future time, using a mathematical model, (2) receive actual feedback from the system and (3) use both the prediction of the model and the feedback from the system to suggest a control move that will help to optimize the desired outcome for a specific plant variable (e.g. minimize temperature). In section 5.2, a more detailed description of this approach is given. More recently, MPC has been used in biological applications involving the regulation of glucose supply in diabetic patients and in the exploration of optimal dosing of Tamoxofin for treating breast cancer [38, 89].

The application of NMPC discussed here, in the context of the inflammatory response, stretches the capability of this tool further than previous applications. The model that we consider is a highly nonlinear system, which cannot be approximated well by any linear system[1], nor can the various rate coefficients be identified as easily from existing data. It is also a model that is not as robust as those available for predicting the effect of insulin on glucose levels or the dynamics of tumor growth. In other words, because the inflammatory response is a very complex process involving positive and negative feedback, it is extremely difficult to predict the response of the various mediators to perturbations (i.e. to therapy) made to one or more of the variables.

In this current exploration of NMPC, we chose to use a small (four equation) ordinary differential equations (ODE) model, the dynamics of which have been thoroughly explored in [98]. There are two essential entities in an NMPC scheme: the process to be predicted and the model predicting the process. The current exposition is completely simulation based, meaning that it is necessary for the actual process, i.e. a patient's immune response, to be emulated by a model. Thus, the ODE model that we use will serve the dual purpose of not

[1]Many NMPC applications transform the nonlinear model describing a process into a linear model that approximates the dynamics, symplifying the model complexity to optimize controller performance.

only predicting patient data but generating the patient data as well. Hence, there are two copies of the model equations.

Initially, it is assumed that the predictive model and the patient (model) are identical with respect to equations, parameter values, and initial conditions. This is referred to as the absence of patient-model mismatch. Later patient-model mismatch will be introduced in some of the parameter values and initial conditions. The model we use is not sophisticated enough to predict quantitative data in actual patients; however, there is currently much work being done in this area, regarding the generation of models that predict quantitative measurements of specific inflammatory mediators, such as Tumor Necrosis Factor (TNF) and Interleukin-6 (IL-6), or anti-inflammatory cytokines, like Interleukin-10 (IL-10) and Transforming Growth Factor-β (TGF-β), among others. The extension of NMPC to such a model holds promise for suggesting optimal therapies and dosing profiles in actual patients.

The process of getting from the initial results to the current results was a rather involved one, including many different paths which cannot all be explained here. However, it is instructive to see a portion of the different strategies that were implemented along the way and the corresponding results and explanations for the modifications that were made from one change to the next. Indeed, the process will still continue as the method is refined and becomes better understood in the context of this problem. While this chapter is only based on *in silico* simulation studies, it is an ambitious and enthusiastic effort toward bringing model-based immunomodulation strategies closer to the bedside of the critically ill.

5.2 METHODS

Nonlinear Model Predictive Control is a methodology for creating a class of control algorithms which encompasses several principles. These principles include the use of a model that describes a certain process to make predictions about future process behavior, in order for recommendations to be made regarding a corrective action to direct the predicted performance closer toward a preferred outcome [86]. As such, schemes of this nature are ideal for industrial processes that can typically be well described by a set of equations. In fact, NMPC has been used on industrial applications since the 1970's.

Recently, NMPC has also been used in the somewhat less concrete area of medicine, where it is more difficult to develop accurate models that describe biological processes. The use of an underlying model for prediction is the key feature that makes the use of this class of control schemes so appealing. In addition, because the algorithm takes into account the differences that are expected to exist between the model and the actual process, the underlying model may not need to make perfect predictions. However, the question "How accurate does the core model need to be?" remains open. In this chapter, we explore this question in the setting of therapy administration in a reduced model of the acute inflammatory response to pathogen. It will be shown that when mismatch between the model and the patient exist, mechanisms need to be put into place in order to maintain the predictive accuracy of the underlying model.

In every NMPC algorithm, there are essential elements that must exist [86]. Below, we outline and describe them, using examples from our NMPC setup and the ordinary differential equation model [98] given below by equations 5.1-5.4.

$$\frac{dP}{dt} = k_{pg}P(1-\frac{P}{p_\infty}) - \frac{k_{pm}s_mP}{\mu_m + k_{mp}P} - k_{pn}f(NA)P, \tag{5.1}$$

$$\frac{dN^*}{dt} = \frac{s_{nr}R}{k_{nr}+R} - \mu_n N^*, \tag{5.2}$$

$$\frac{dD}{dt} = k_{dn}\frac{f(N^*)^6}{x_{dn}^6 + f(N^*)^6} - \mu_d D, \tag{5.3}$$

$$\frac{dC_A}{dt} = s_c + k_{cn}\frac{f(N^* + k_{cnd}D)}{1 + f(N^* + k_{cnd}D)} - \mu_c C_A \tag{5.4}$$

where

$$R = \frac{(k_{np}P + k_{nd}D + k_{nn}N^*)}{1 + (C_A/c_\infty)^2},$$

$$f(x) = \frac{x}{1 + (C_A/c_\infty)^2},$$

I. *The Specification of a Reference Trajectory*

The reference trajectory defines the target level that we would like our process output variables to eventually achieve. For instance, in our model, we would like damage to eventually decrease back down to zero, if it is currently at an elevated state. So, our reference trajectory for damage might be the constant function, $R_D(t) = 0$, or it might be a function gradually decreasing to the target value, perhaps even in a step-wise fashion. The reference trajectory can be thought of as the outcome goal that is desired. It is used in the objective function to define what is to be optimized, or more specifically, in our case, what is to be minimized.

II. *The Prediction of Process Output*

Prediction of process behavior (i.e. patient data) is accomplished via the underlying model. In our case, this is a reduced ordinary differential equation model, given by 5.1-5.4, describing the process of the acute inflammatory response to pathogen.

III. *The Definition of an Objective Function*

The objective function defines the goals to be achieved by an optimizer routine, which is an algorithm for locating the values at which the objective function reaches a minimum (or maximum) given constraints on the function variables. (See IV below.) The objective function(s) used in this chapter have the typical two norm squared form, $\|\cdot\|_2^2$ [78]. In addition, the objective function penalizes the *change* in doses, Δu, as well as the actual dose amount, u. Ultimately, the minimization of the objective function is achieved by selecting only Δu, since u is uniquely specified by the sequence of dose changes.

IV. *The Computation of a Sequence of Control Actions*

Using the model to predict future values of the system's response to changes in input, an optimal sequence of control moves (i.e. input changes) are sought that will bring the specified output variables as close as possible to the reference trajectory. The control is simply an input into the ODE system, calculated in such a way as to achieve the desired goal. For example, in our model, there is a positive control term in the anti-inflammatory equation which represents an anti-inflammatory therapy. Thus, the goal might be to find the right amount of anti-inflammatory therapy which will minimize the distance between the predicted levels of damage and the reference trajectory for damage, over a specified prediction window, h, given constraints on the dosing.

Practically, this is accomplished via the use of the *fmincon* algorithm made available by MatLab [72]. This algorithm finds a minimum of a constrained nonlinear multivariable function. In other words, it solves what is commonly known as a nonlinear programming problem. In particular, *fmincon* uses a sequential quadratic programming method. For more information, see [73]. The number of control moves, m, that are allowed during the prediction horizon, h, can also be specified. If the algorithm is set so that the model predicts out to $h = 24$ hours and therapy is administered on an hourly basis, m is number of control actions (doses) to be computed in order to minimize the objective function over 24 hours. If $m < h$, which is typically the case, then the m^{th} control move is held constant for the remainder of the prediction window, in determining the system's response to the input given over this time frame. See the MPC schematic given on page 997 of [86]. Thus, in our simulations, the objective function is minimized over all possible values of Δu for m control moves over a prediction window, h. Only the first control move is then implemented as the dose for the current hour, after which the algorithm moves on to find the dose for the next hour.

V. *Error Prediction Update*

A very important element of the algorithm is in this error prediction step. Error prediction is implemented to correct the imbalance that may exist between patient data and the current state of the model (i.e. when there is patient-model mismatch). After the current control action is implemented in both the process and the model, a current measurement, $M(k)$, of the process is taken and compared to the current model state, $p(k)$, where k is the current time step in the algorithm. This error quantity, $M(k) - p(k)$, is then used to update upcoming predictions used to calculate the dose for the next hour. It is minimized as a part of the objective function terms pertaining to the output variables being measured. (Note that while it is technically feasible to have all the model variables designated as output variables that can be measured, in reality this is typically not the case.)

In our NMPC scheme, when a mismatch exists, updating is done differently. This is due to the fact that in these particular instances, the variables which can be realistically measured are not those that appear in the objective function. For instance, we may wish to minimize damage (D) and pathogen (P), while only having access to measurements of the levels of activated phagocytes (N^*) and the anti-inflammatory mediator (C_A). Hence, the difference in the measurements between the patient data for N^* and C_A and the current model values for these two mediators cannot be worked into an objective function that contains only terms for the damage and pathogen variables. The strategies we explore for this situation are discussed later in the context of specific configuration types where it applies.

On page 997 of [86], a schematic is given that aptly illustrates some of the elements of an MPC algorithm. For equations 5.1-5.4, there are three states (outcomes) that are possible given certain parameter values and initial pathogen levels [98]:

1. Healthy: Healthy is defined as the state in which pathogen has been eliminated and the mediators have returned to their baseline levels at the end of the simulation time.
2. Aseptic: Aseptic is defined as the state in which pathogen has been eliminated, yet both the pro-inflammatory and anti-inflammatory mediators are at elevated levels at the end of the simulation time.
3. Septic: Septic is defined as the state in which all mediator levels, as well as pathogen levels, are elevated at the end of the simulation time.

Through the use of the NMPC control algorithm, proper therapy dosing profiles are identified in order to correct inflammatory responses that would result in either the aseptic or septic scenarios in the absence of any controller based therapy. The resulting therapy found by the control algorithm is referred to as *targeted therapy*. In addition, the aim also includes not harming those patients whose inflammatory response resolves to the healthy state in the absence of targeted therapy. It is also assumed that basic therapy, including the administration of antibiotics, resuscitation of fluids, and so forth, are implicitly modeled in system (5.1)-(5.4). This means that the various outcomes mentioned above can occur despite administration of basic treatment. This assumption is made due to the small size of the model.

The algorithm we use is a modified version of that used by Florian et al. in [38], acquired through [37]. The initial algorithm underwent many customized modifications throughout the different stages of this NMPC exploration for inflammation, which are discussed throughout this chapter. Much thanks are owed to Jeff Florian and Robert Parker of the University of Pittsburgh's Chemical Engineering Department for the suggestion to apply NMPC, specifically, to the problem at hand, for making an algorithm available, and for their very helpful assistance in the beginning stages of becoming familiar with the NMPC methodology and the specific algorithm. In all of the simulations that we discuss, the total simulation time is 168 hours (1 week). In addition, k is an hourly step, so doses are given on an hourly basis and administered as instantaneous injections.

5.3 RESEARCH AND DEVELOPMENT PHASE

Having established the general components of the NMPC scheme and the basic behaviors present in the model, we now discuss the metamorphosis of the algorithm through the various exploratory stages of finding therapeutic strategies to correct the septic and aseptic scenarios previously discussed. In the initial investigations, it is assumed that the process, i.e. the patient, and the model describing the process/patient are one and the same. Later, a mismatch between the two is introduced, since such an incongruity is more realistic. There are other differences between the various configurations and each will be explained separately in detail.

5.3.1 Configuration 1: No Mismatch; 1 Therapy

In this setup, there is no patient-model mismatch. Hence, we will generically refer to the "system" instead of the model and the patient, separately, until a later setup when we introduce differences between the two. The model equations (5.1)-(5.4) are numerically integrated from a given initial condition for 6 hours. After this time, therapeutic intervention is initiated by way of the NMPC algorithm. We assume that at this time point the system is not yet in a state of "mortality" and that therapeutic intervention is a feasible option.

For this setup, anti-inflammatory therapy is explored. It is modeled in the equation of the anti-inflammatory mediator, C_A, as a positive source term, $AIDose$:

$$\frac{dC_A}{dt} = s_c + k_{cn}\frac{f(N^* + k_{cnd}D)}{1 + f(N^* + k_{cnd}D)} - \mu_c C_A + AIDose.$$

Furthermore, there are constraints which prevent dosing from going negative, meaning that therapy can be infused into the system but not extracted. Later, we explore the possibility of "extraction therapy." The maximum dose amount of therapy allowed at a given step is calculated as the difference between the current level of C_A and $C_AMax = 0.6264$. The value for C_AMax was chosen based on the analysis done in [98] on the subsystems of equations (5.1)-(5.4), namely the Damage-Pathogen subsystem. Values of C_A higher than 0.6264 cause the system to exhibit nonbiological behavior. Thus, this maximum level of C_A is imposed to ensure that the NMPC results remain plausible.

The objective function in this setup contains terms to penalize the following: damage levels (D), pathogen levels (P), changes in dosing (Δu), and total amount of drug delivered (u). As mentioned, the objective function uses the standard two norm squared:

$$\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u}\Delta u\|_2^2 + \|\Gamma_u u\|_2^2 \tag{5.5}$$

The Γ-parameters are weighting constants, which can be used to emphasize the importance of one term over another.

The variables D and P are also the output variables that are measured from the patient. However, because there is no patient-model mismatch in this case, there is no difference between the current measurement, $M(k)$, of the damage and pathogen levels in the patient and the current model state for damage and pathogen. Thus, the predicted error is zero. For this case, we experimented with a range of values for m and h, deciding on $m = 2$ and $h = 24$. In general, there is no systematic way to choose the various controller "tuning parameters" which include m, h, and the weighting parameters present in the objective function.

Summary for Configuration 1:

- No patient-model mismatch
- One Therapy: Anti-inflammatory therapy (no extraction) given on an hourly basis
- Measured patient output variables: D and P
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u}\Delta u\|_2^2 + \|\Gamma_u u\|_2^2$
- $m = 2$ and $h = 24$
- Γ weighting constants: various values explored
- $C_AMax = 0.6264$
- Pathogen growth rate values: Aseptic scenario: $k_{pg} = 0.514$; Septic scenario: $k_{pg} = 0.52$

- Initial conditions for simulations: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$

To arrive at the specific configuration given in Setup 1, there were a number of prior configurations that led us to this one. For instance, for a time, we only considered one output variable (measurable), namely D, which was also the only model variable appearing in the objective function. However, it became apparent that if minimizing pathogen was not also a part of the objective, then the algorithm would do its best to minimize damage by administering the anti-inflammatory therapy; however, this would only prevent the inflammation from trying to eradicate pathogen on its own. Instead, pathogen would grow uncontrolled because inflammation was being suppressed by treatment. This, obviously, was not ideal, so there needed to be a modification to let the algorithm know that it was important to minimize both damage and pathogen.

This, however, introduces a difficult challenge: maintaining a balance between these two objectives. An emphasis on minimizing damage, as mentioned, might lead to unrestricted pathogen growth. On the other hand, an emphasis on minimizing pathogen might lead to an overzealous immune response bent on eliminating pathogen as soon as possible whatever the costs, after which it might be too late to bring the inflammation back down. The latter difficulty becomes more apparent in Configuration 3 on page 209 where both pro- and anti-inflammatory therapies are considered.

The term in the objective function, $\|\Gamma_u u\|_2^2$, responsible for penalizing total therapy administered, was also something not initially included. In preliminary experiments, the control was essentially overdosing the system with therapy. In other words, the therapy would continue to be given even after it was clear that the model dynamics were responding favorably to treatment (i.e. the reference trajectory was being reached) and would only need time, not more therapy, to resolve to the healthy state. This problem occurred because of the third term, $\|\Gamma_{\Delta u}\Delta u\|_2^2$, in the objective function 5.5 that penalizes the change in dose, Δu, from one step to the next. Once the reference trajectory was reached, the controller's strategy to minimize the objective function focussed on not making any more dose *changes*, (i.e. $\Delta u = 0$) and so the current dose would stay at its current level for the remainder of the simulation. Hence, it became necessary to add a fourth term, $\|\Gamma_u u\|_2^2$, in 5.5, which penalizes the amount of drug delivered. This way, once the reference is reached, there will be competing objective function penalties: one hindering alterations in dose level and one attempting to reduce how much drug is delivered. This allows the system to taper off to zero administered drug [37].

In addition, the maximum level on C_A was not implemented until some of the results we were getting did not appear to be in line with normal model behaviors. Our early experiments led to the specification of an upper bound on the amount of therapy allowed at the current dosing step. This maximum allowable dose is calculated based on the current level of the anti-inflammatory mediator. For the model parameter values established in [98], the amount of C_A in the system can reach a maximum level, specifically, $C_AMax = 0.6264$, before a break down occurs in the model dynamics established in the subsystems used to build the model. See [98] for more details.

The manner in which therapeutic intervention was initiated was another aspect that evolved in the process of settling on Configuration 1. In prior simulations, the starting point for initiating therapy was quite different. The initial pathogen level and pathogen growth rate were chosen so that in the *absence of any therapy*, the system would end up aseptic. A second pathogen load was introduced as the system approached the aseptic state – simulating a secondary infection. It was assumed that at the onset of the secondary infection, the system was not yet in a state of "mortality" and hence therapeutic intervention was a feasible option. Immunomodulation was then initiated after the onset of the secondary infection instead of simply after a specified amount of time (6 hours), as is the case for Configuration 1.

Initiating immunomodulation in the previous manner was

1. more complicated than what we wanted an initial exploratory simulation to be, and
2. would not apply in the future, when patient-model mismatch is introduced.

For reason (2), there would be a need for treatment to be initialized, other than at the onset of an infection, either initial or secondary. Hence, it was determined that for the setup in Configuration 1, 6 hours was enough time for the system to evolve "naturally" from its initial state so that treatment is not started at time zero, an unlikely scenario. However, 6 hours is soon enough after the onset of infection for treatment to still be a possibility and not a time when the patient is in a physiologic state beyond the point of intervention. Since there are outward signs such as fever, elevated heart rate, etc, that can alert physicians to the presence of an infection, we equated semi-elevated levels of N^* at 6 hours with being "sick enough" to warrant the initiation of targeted therapy. Again, we reiterate our assumption that model dynamics incorporate basic care, such as antibiotics, fluid resuscitation, etc, and that the patient's condition at 6 hours is in spite of (or, perhaps, even because of) this earlier basic treatment.

5.3.2 Configuration 2: No Mismatch; 2 Therapies

In Configuration 2, there is again no patient-model mismatch. Thus, again, we refer to the "system" instead of the model and the patient, separately. Like Config. 1, after 6 hours of running (numerically integrating) the system, therapeutic intervention is initiated by way of the NMPC algorithm.

Although the original NMPC algorithm was written to handle more than one control input, it had only been used and tested for one control input. Thus, there were some errors that needed to be worked through in order for the code to work properly in the case of two inputs, which this setup considers. The therapy for this setup includes both an anti-inflammatory therapy, present as a source term ($+AIDose$) in the equation of the anti-inflammatory mediator, C_A, as well as a pro-inflammatory therapy, present as a source term ($+PIDose$) in the equation for activated phagocytes, N^*:

$$\begin{aligned}\frac{dN^*}{dt} &= \frac{s_{nr}R}{k_{nr}+R} - \mu_n N^* + PIDose \\ \frac{dC_A}{dt} &= s_c + k_{cn}\frac{f(N^*+k_{cnd}D)}{1+f(N^*+k_{cnd}D)} - \mu_c C_A + AIDose\end{aligned}$$

As was the case for Config. 1, constraints are defined which prevent dosing from going negative, meaning that therapy can be infused into the system but not extracted. Our objective function contains terms to minimize the following: damage levels, D, pathogen levels, P, changes in dosing (Δu_1 and Δu_2) and total therapy given (u_1 and u_2) for both C_A and N^*, respectively:

$$\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u1}\Delta u_1\|_2^2 + \|\Gamma_{\Delta u2}\Delta u_2\|_2^2 + \|\Gamma_{u1}u_1\|_2^2 + \|\Gamma_{u2}u_2\|_2^2 \quad (5.6)$$

Again, the Γ-parameters are the weighting constants. The maximum dose amount of anti-inflammatory therapy allowed at a given step is calculated as the difference between the current level of C_A and $C_AMax = 0.6264$. The maximum dose amount of proinflammatory therapy allowed at a given step is calculated as the difference between the current level of N^* and $N^*Max = 2.0$. The maximum was chosen based on average levels of N^*; however, in later simulations we will see that this maximum level is too high.

The variables D and P are also the output variables that are measured from the patient. However, because there is no patient-model mismatch, there is no difference between the current measurement, $M(k)$, of the damage and pathogen levels in the patient and the current model state for damage and pathogen. Thus, the predicted error is zero. For this case, we experimented with a range of values for m and h, deciding on $m = 2$ and $h = 24$ for the results shown.

Summary for Configuration 2:

- No patient-model mismatch
- Two Therapies: Anti-inflammatory and proinflammatory therapy (no extractions)
- Measured patient output variables: D and P
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u1}\Delta u_1\|_2^2 + \|\Gamma_{\Delta u2}\Delta u_2\|_2^2 + \|\Gamma_{u1}u_1\|_2^2 + \|\Gamma_{u2}u_2\|_2^2$
- $m = 2$ and $h = 24$
- Γ weighting constants: various values explored
- $C_AMax = 0.6264$
- $N^*Max = 2.0$
- Pathogen growth rate values: Aseptic scenario: $k_{pg} = 0.514$; Septic scenario: $k_{pg} = 0.52$
- Initial conditions for simulations: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$

5.3.3 Configuration 3: Patient-Model Mismatch; 2 Therapies

Unlike that of Configurations 1 and 2, here, in Configuration 3, patient-model mismatch is introduced, with respect to several model parameters and two initial conditions. Patients are "generated" with varying profiles, meaning that some of the parameters used in the model representing Patient X are different from those of the underlying predictive model and are different from profiles of other generated patients. Several parameters in the equations 5.1-5.4 are identified as reasonable parameters for which variability may exist from patient to patient. These parameters include those given in Table 9. Note that initial conditions P_0 and C_{A0} are treated as parameters here.

The values in [98] used for the parameters shown here in Table 9 (except for the values of P_0 and kpg) are considered here as the "mean" values. Furthermore, variability for the ranges was chosen to be $+/-25\%$ of the mean, for those parameters that had "estimated" values in [98]. For these parameters, the range of variability is calculated so that $+/-25\%$ of the mean value is two standard deviations on either side of the mean. Then, when a patient profile is generated, values from this range are chosen randomly with a normal distribution. For the parameter k_{pg}, where a range is available (see [98]), values from this range are chosen randomly with a normal distribution, where the lower and upper bounds on k_{pg} are assumed to be two standard deviations from the mean.

Any randomly generated values that fall outside of the range of variability are discarded and another value is generated, until a value is found that falls in the range of variability. Hence, the parameter values are chosen with a "normal-like" distribution because of the upper and lower bounds of the variability ranges. In addition, some parameters are chosen to co-vary. This means that if paramters p_1 and p_2 co-vary and the value chosen for p_1 is $+n\%$ of its mean value, then the generated value for p_2 should also be $+n\%$ of its mean value. For example, the variability in k_{cnd}, the production of the anti-inflammatory mediator (C_A) by damaged tissue (D), is to vary by the same percentage as k_{cn}, the production of C_A by activated phagocytes (C_A), so that one is not relatively producing more or less (C_A) than the other.

The simulation results for this configuration are not viewed in graph format as those done for Configurations 1 and 2 above. Instead, the individual outcomes (healthy, aseptic, septic) from each simulation are tallied and reported as percentages of the total number of patients treated with targeted therapy. The first step in this process is generating the patient population, which is accomplished by selecting parameters in the way described above. This creates a "profile" for an individual patient. This is repeated for the total number of patients to be considered. In this case, there will be 1000 patients.

Table 9: Model parameters in which variability was assumed in the patient-model mismatch case given by Configuration 3. Patient parameters are generated by choosing a normally (normal-like) distributed random value from the given ranges.

Name	Patient Parameter Ranges	Mean	Description
P_0	0.0-1.0	0.5	Initial condition of pathogen (P)
C_{A0}	0.0938-0.1563	0.125	Initial condition of the anti-inflammatory mediator (C_A)
k_{pg}	0.021-1.0	0.5105	Growth rate of pathogen (P)
k_{cn}	0.03-0.05	0.04	Maximum production of anti-inflammatory mediator (C_A)
k_{nd}	0.015-0.025	0.02	Activation of phagocytes by tissue damage (D)
k_{np}	0.075-0.125 (Co-varies w/ k_{nd})	0.1	Activation of phagocytes (N^*) by pathogen (P)
k_{cnd}	36.0-60.0 (Co-varies w/ k_{cn})	48.0	Controls relative effectiveness of activated phagocytes (N^*) versus damage (D) in the production of the anti-inflammatory mediator (C_A)
k_{nn}	0.0075-0.0125 (Co-varies w/ k_{nd})	0.01	Activation of phagocytes (N^*) by already activated phagocytes (N^*) (or the cytokines that they produce)

As discussed in Configuration 2 on page 208, the time at which intervention is initiated becomes an issue for consideration when there is patient-model mismatch. This is because in the case of mismatch, many patients are considered, and it no longer seems realistic to initiate therapy at the exact same time in each patient. In fact, for some patients, 6 hours into the infection might not be long enough for signs of the infection to show, whereas in others, 6 hours might be too late. Therefore, the intervention time in Configuration 3 is based on the level of N^*. If a patient's N^* level rises above a certain threshold, then the patient is deemed "sick enough" to receive targeted therapy. This implies a biomarker driven approach to initiating therapeutic intervention. This implementation, however, is not without its caveats. For instance, how should the threshold be chosen? This question is likely to be answered differently depending on the physician one talks to. In the current exploration, an N^* threshold of 0.1 was selected, since this amount was a considerable elevation for this variable in a simulation having a septic or aseptic endpoint.

Like Configuration 2, the therapy for Configuration 3 includes both an anti-inflammatory therapy, present as a source term ($+AIDose$) in the equation of the anti-inflammatory mediator, C_A, as well as a pro-inflammatory therapy, present as a source term ($+PIDose$) in the equation for activated phagocytes, N^*:

$$\begin{aligned}\frac{dN^*}{dt} &= \frac{s_{nr}R}{k_{nr}+R} - \mu_n N^* + PIDose\\ \frac{dC_A}{dt} &= s_c + k_{cn}\frac{f(N^*+k_{cnd}D)}{1+f(N^*+k_{cnd}D)} - \mu_c C_A + AIDose\end{aligned}$$

As was the case for Config. 1, constraints are defined which prevent dosing from going negative, meaning that therapy can be infused into the system but not extracted. As before, the objective function, given by (5.6), contains terms to minimize the following: damage levels, D, pathogen levels, P, changes in dosing (Δu_1 and Δu_2) and total therapy given (u_1 and u_2) for both C_A and N^*, respectively.

The maximum dose amount of anti-inflammatory therapy allowed at a given step is calculated as the difference between the current level of C_A and $C_AMax = 0.6264$. The maximum dose amount of proinflammatory therapy allowed at a given step is calculated as the difference between the current level of N^* and $N^*Max = 1.0$. The maximum was chosen based on average levels of N^* during a moderate to severe infection and the fact that the value used in Config. 1 ($N^*Max = 2.0$) was too high. However, in later simulations, we will see that even this reduced maximum level is probably too high.

For this configuration, it is N^* and C_A that are designated as the output variables to be measured, instead of the variables D and P. This change was incorporated because damage is a difficult, if not impossible, variable to quantify and measure in clinical settings, and it is unlikely that a measurement of the pathogen population could be made at all, much less made every hour. However, we still want the algorithm to focus on minimizing damage and pathogen levels. Because of this, typical error prediction and updating procedures cannot be employed here, as was briefly discussed in point *V: Error Prediction Update* on page 202. The difference in the measurements, between the patient data for N^* and C_A and the current model values for these two mediators, cannot be worked into an objective function that only contains mediator prediction terms for the damage and pathogen variables. This becomes very important, now that we have introduced patient-model mismatch, and the underlying model is now different from the patient model, with respect to parameter values and starting conditions.

Thus, to address this, the model state for N^* and C_A is synchronized with the current patient data measurements for N^* and C_A, right after the prescribed dose for the current hour has been implemented in the model and in the patient and before the next measurements are taken. Thus, in essence, hourly measurements are taken of the patient's levels of N^* and C_A and the model is updated with these values.

The underlying model drives the design of the therapy that is chosen to be implemented in the patient, with measurements taken from the patient to correct for mismatch. Therefore, the accuracy of the underlying model is important. Note, however, the updating scheme described in the last paragraph does not address any possible discrepancies between the model and patient with respect to levels of damage and pathogen, both of which are the cause for pushing the system toward an unhealthy endpoint.

Thus, in the initial exploration of this configuration, it was determined that the model parameters for those given in Table 9 on page 211, specifically P_0 and k_{pg}, were not chosen well enough to accurately represent those patients who were on the path to a septic or aseptic outcome. In other words, the therapy regimens that the controller was finding were being based on an underlying model that normally would resolve to healthy, being able to eliminate pathogen and damage even in the absence of therapy. So, in many cases, the levels of pathogen were still elevated in the patient, but not in the model. Such a mismatch, between model and patient, was not adequately being taken care of with the synchronization of the N^* and C_A values in the model with the patient measurements.

As a first attempt at addressing this issue, the model was conditioned to be more representative of a sicker patient, by increasing the model's pathogen growth rate, k_{pg}. This makes the pathogen more powerful in the model, and thus, the resulting therapy would be more applicable to patients which experienced a stronger, more persistent infection. However, this change only indirectly addresses the problem when pathogen levels in the patient and model are vastly different. It became apparent from the results of this configuration that some kind of updating involving pathogen would need to be included, without directly synchronizing the model pathogen levels with a measurement of pathogen levels in the patient. This is addressed in the next configuration. Finally, for this configuration, we experimented with a range of values for m and h, deciding on $m = 2$ and $h = 24$ for the results shown.

Summary for Configuration 3:

- Patient-model mismatch
- 1000 patients considered, with differing profiles (i.e. differing values for the parameters shown in Table 9 on page 211
- Patient profiles are generated by choosing a normally (normal-like) distributed random value from a specified range for six selected parameters and two initial conditions.
- Therapy is initiated in the patient when N^* levels reach a threshold of 0.1.
- Two Therapies: Anti-inflammatory and proinflammatory therapy (no extractions), given on an hourly basis
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u1} \Delta u_1\|_2^2 + \|\Gamma_{\Delta u2} \Delta u_2\|_2^2 + \|\Gamma_{u1} u_1\|_2^2 + \|\Gamma_{u2} u_2\|_2^2$
- Γ-weighting constants: various values explored
- $C_A Max = 0.6264$
- $N^* Max = 1.0$
- Measured patient output variables: N^* and C_A

- Model $k_{pg} = 0.8$
- Model initial conditions: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$
- Patient initial conditions: $(P_0, N^*, D, C_A) = (P_0\text{-random}, 0, 0, C_{A0}\text{-random})$
- $m = 2$ and $h = 24$

5.3.4 Extraction Therapy Considered

Before explaining the various changes made for the next configuration, we take a brief moment to discuss a side path that was explored between Configuration 3 and Configuration 4. This involved the consideration of "extraction therapy" with respect to both the pro- and anti-inflammatory therapies. Each could be administered not only in a positive way (addition to the system), but also negatively (extraction from the system). Technically, this means that the algorithm was allowed to find changes in the doses (Δu_1 and Δu_2) that caused the doses (u_1 and u_2) to be a negative quantity, implying that C_A and/or N^* were being removed from the patient. Hence, because each therapy can be given and taken out, there are essentially four therapies that can be employed.

However, the consideration of four therapies is more unrealistic than the previous configurations. In the initial experimentation with the use of four therapies, it quickly became apparent that the extra "knobs" available for turning the levels of C_A and N^* up and/or down were too powerful for the size of model being used. The usual strategy the controller would exercise would be to promote inflammation by giving pro-inflammatory therapy and extracting some of the anti-inflammatory mediator to ensure pathogen was eliminated. Then, when pathogen was under control, the controller would extract the pro-inflammatory mediator and give anti-inflammatory therapy in order to bring down the inflammation previously created, thereby helping the patient resolve to healthy.

This is illustrated in the Figure 61, showing a patient treated with this "4-knob" therapy (targeted therapy shown in blue). The patient would have otherwise ended up in a septic state (placebo shown in red). The therapy dosing strategy described above can be seen in the last row of Figure 61. The left panel of the last row is the pro-inflammatory doses and the right panel is the anti-inflammatory doses. The therapy is hugely successful in preventing the patient from ending up in the septic case.

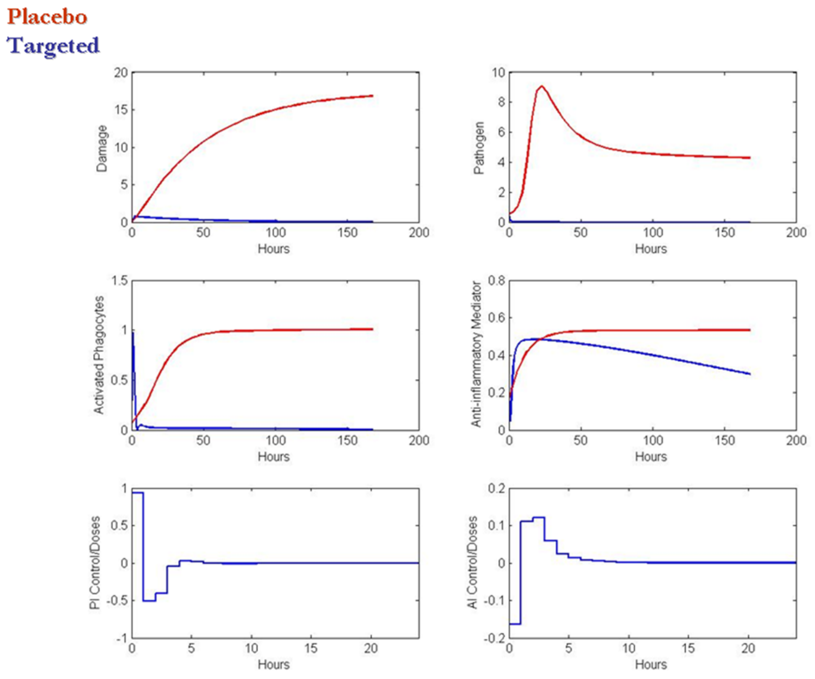


Figure 61: A sample simulation from a simulation incorporating four therapies. These include the addition as well as the extraction of pro- and anti-inflammatory therapy, PI and AI, respectively. A difficult scenario, that would otherwise end up septic (red curves) is managed very well by the algorithm (blue curves, first 4 panels) with the use of the four therapies. The PI doses are shown in the left panel of the last row. The AI doses are shown in the right panel of the last row. Both are shown over a 24 hour period, after which no more doses are given.

Although it was interesting to explore this configuration and encouraging to know that the therapy regimen generated by the algorithm made sense, a four therapy strategy like this one is not one that can realistically be discussed at this time. Methods for extracting cytokines and other molecules from a patient's bloodstream are still in the very early stages of development. Furthermore, in a model the size of the one we are using, although the dynamics are complex, there is only so much that can happen in response to input. In a larger model, input to one variable might not imply a global-like affect on the system dynamics. However, in a small model, inputs tend to affect the entire dynamics, so it is somewhat artificial to allow the consideration of too many therapies. Thus, we return to the two therapy configuration, but make additional changes, specifically to address the mismatch regarding pathogen and also to add some realism with respect to the length of time that the anti-inflammatory mediator can stay elevated from the addition of therapy.

5.3.5 Configuration 4: Patient-Model Mismatch; 2 Therapies; Pathogen Update; C_A therapy cap

In Configuration 4, the patient parameters are generated slightly differently than in Configuration 3. During the many simulations conducted with Configuration 3, it became clear that the patient placebo outcomes were primarily being driven by the patients' values of the pathogen growth rate, k_{pg}, and the initial pathogen load, P_0. To address this problem, we first modified the patient generator scheme so that the selected parameters are chosen randomly with a *uniform* distribution rather than a normal distribution from the specified variability ranges.

This allows more variability in parameter selection, since, with a uniform distribution, all the values in a range are equally likely to be chosen. In addition, the range for the pathogen growth rate, k_{pg}, was modified so that the placebo outcomes in the patient population are not driven primarily by k_{pg} and P_0, but by the parameter profile as a whole. Also modified was the N^* threshold for intervention: from 0.1 down to 0.05. The scatterplot shown in Figure 62 depicts patient placebo outcomes with respect to corresponding k_{pg} and P_0 levels. The scatterplot shows that the new range for k_{pg}, along with the reduced N^* threshold, allows a significant degree of overlap between outcome possibilities in the k_{pg}-P_0- plane, meaning that parameters other than P_0 and k_{pg} are driving the outcome in a significant number of individuals. Table 10 on page 219 shows the patient parameter ranges again with an updated range for k_{pg}.

Another significant change made for Configuration 4 is with respect to the issue mentioned in Configuration 3, regarding the case when pathogen levels in the patient and model are vastly different. An updating strategy is necessary; however, directly measuring pathogen levels in the patient and synchronizing the model with this value is unrealistic and artificial. Hence, we added an update that notes the level of pathogen

1. in the patient when it increases above a specified threshold, and
2. in the model when it is under a certain small threshold.

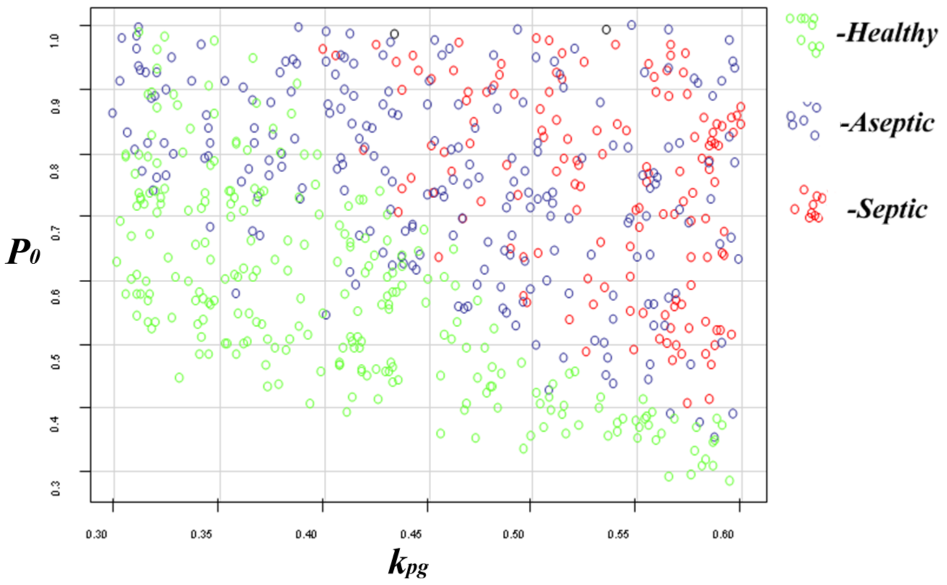


Figure 62: The scatterplot showing the distribution of patient outcomes with respect to k_{pg} and P_0. For the given ranges of k_{pg} and P_0 the various outcomes (healthy, aseptic, and septic) of the patient profiles in the placebo case are well mixed in the P_0-k_{pg} plane. Thus, P_0 and k_{pg} are not the primary driving force of patient outcome.

Table 10: Model parameters in which variability was assumed in the patient-model mismatch case given by Configuration 4. Patient parameters are generated by choosing a uniformly (uniform-like) distributed random value from the given ranges.

Name	Patient Parameter Ranges	Description
P_0	0.0-1.0	Initial condition of pathogen (P)
C_{A0}	0.0938-0.1563	Initial condition of the anti-inflammatory mediator (C_A)
k_{pg}	0.3-0.6	Growth rate of pathogen (P)
k_{cn}	0.03-0.05	Maximum production of anti-inflammatory mediator (C_A)
k_{nd}	0.015-0.025	Activation of phagocytes by tissue damage (D)
k_{np}	0.075-0.125 (Co-varies w/ k_{nd})	Activation of phagocytes (N^*) by pathogen (P)
k_{cnd}	36.0-60.0 (Co-varies w/ k_{cn})	Controls relative effectiveness of activated phagocytes (N^*) versus damage (D) in the production of the anti-inflammatory mediator (C_A)
k_{nn}	0.0075-0.0125 (Co-varies w/ k_{nd})	Activation of phagocytes (N^*) by already activated phagocytes (N^*) (or the cytokines that they produce)

If 1. and 2. above occur at the same time, then the pathogen level of the *model* state is reset to a higher level compared to the pathogen state at which it is currently. This can be looked at as a type of re-initialization of the pathogen value in the model. Although this strategy does use a measurement of the patient's pathogen values, we note that this is the only way that the current simulation setup can detect the presence of a persistent infection. In a clinical setting, a physician can see outward signs that an infection is still raging, i.e. fever, elevated heart rate, etc. Thus, because we are not directly setting the model's pathogen state to the value "measured" in the patient, the updating strategy is a reasonable way to alert the algorithm of the case when pathogen levels in the patient are high and those in the model are not. Practically, the levels of pathogen are checked every four hours and if the criteria are met, the model state for pathogen is reset to its initial value of 0.5.

As we worked through the process of developing one configuration to the next, making the algorithm more realistic was one of the main goals. Therefore, when it was noticed that the amount of anti-inflammatory therapy given to patients would sometimes cause the levels of the anti-inflammatory mediator to stay elevated for very long periods of time, a mechanism to prevent this needed to be put in place. This problem usually happens in scenarios when inflammation is high after the eradication of pathogen, and the production of anti-inflammation by inflammatory mediators causes the levels of C_A to be elevated as well. Then, when the anti-inflammatory therapy is given, pushing the level of C_A up to the maximum allowable level, it does not have an immediate effect on bringing inflammation down. Thus, the therapy would have to be continually given in an attempt to essentially saturate the system with as much C_A as possible for as long as possible.

This, however, is not a realistic treatment regimen, since clinicians are careful not to purposely induce a state of immunosuppression in patients, which would make them susceptible to secondary infections. In order to address this issue, a mechanism was put into place so that if the level of C_A remains constant for more than 48 hours, the maximum allowable amount of C_A is reduced by half. In some cases, this means that patients who fall into this category might end up aseptic instead of healthy since the amount of inflammation in their system is decreasing, and the amount of inflammation, that may have been decreasing with the elevated C_A levels, will once again rise. However, as mentioned, the alternative is not a realistic one.

The last change made from the previous configurations is the reduced value of the prediction horizon, h, from 24 to 8 hours. It was determined that an 8 hour prediction window was long enough to capture essential model dynamics and it also decreased the computational time for running the algorithm on 1000 patients. The value of h (as well as m) may have to be reconsidered in future configurations, especially since computational time has become less of an issue, due to the resources now available.

This configuration, while the most sophisticated of those presented, is not without its difficulties, as will be shown in the Results section. Hence, also included in the Results section after the outcomes for Config. 1-4 are presented, are several ideas and a few preliminary results for future configurations that address some of the issues brought up by Configuration 4.

Summary for Configuration 4:

- Patient-model mismatch
- 1000 patients considered, with differing profiles (i.e. differing parameter values)
- Patient profiles generated from choosing a uniformly distributed random value from a specified range (ranges shown in Table 10 on page 219)
- N^* threshold $= 0.05$ (for determining when to initiate therapeutic intervention)
- Two Therapies: Anti-inflammatory and proinflammatory therapy (no extractions), given on an hourly basis
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u1} \Delta u_1\|_2^2 + \|\Gamma_{\Delta u2} \Delta u_2\|_2^2 + \|\Gamma_{u1} u_1\|_2^2 + \|\Gamma_{u2} u_2\|_2^2$, where the following are being minimized: damage levels, D, pathogen levels, P, changes in dosing (Δu_1 and Δu_2) and total therapy given (u_1 and u_2) for both C_A and N^*, respectively.
- Γ-weighting constants: various values explored
- $C_A Max = 0.6264$
- $N^* Max = 1.0$
- Measured patient output variables: N^* and C_A

- Model $k_{pg} = 0.8$
- Pathogen update mechanism
- 48 hour C_A cap
- Model initial conditions: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$
- Patient initial conditions: $(P_0, N^*, D, C_A) = (P_0\text{-random}, 0, 0, C_{A0}\text{-random})$
- $m = 2$ and $h = 8$

5.4 RESULTS

The configurations of the last section detailed the setup of each individual simulation or set of simulations. In this section, specific results for each configuration will be shown, keeping in mind the discussions and points made in the previous sections for the corresponding configuration types. For the configurations in which the simulation sets included 1000 patients, not only are the results from one configuration compared to another, the results within an individual configuration are compared to alternative therapies that we now describe.

5.4.1 Alternative Therapies for Multi-Patient Simulations

The therapy strategies found by the NMPC algorithm are referred to as "targeted" therapy. In the simulation sets that include 1000 patients, all with differing profiles, the algorithm generates a therapy specific to that patient's particular dynamics. Many times, the dosing regimens are similar among patients, but nonetheless, the therapy specifically targets an individual patient. In order to get an idea of how well the targeted therapy does, it is compared to the results from the administration of alternative therapies.

These alternate therapies include, of course, the *Placebo Therapy*, where no control-based treatment is given. In addition, two other alternative therapies are explored. The first of these is known as *Standard Therapy.* Standard therapy is calculated in the following way: using the underlying model (and corresponding parameters) as *both* the patient and the model and starting from the usual model initial conditions of $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$, with the setup given in one of the configurations, the control algorithm is run on this "patient" and a dosing profile is generated, as usual. For example, in Config. 3, standard therapy is calculated by implementing all the different specifications of that configuration in the algorithm and running the algorithm on a patient with exactly the same parameter values as the model. In other words, the model becomes one of the patients and a targeted therapy is found for the model patient. This one dosing profile then becomes a standard therapy to administer to all the patients. An example of a standard therapy dosing profile is given in Figure 63:

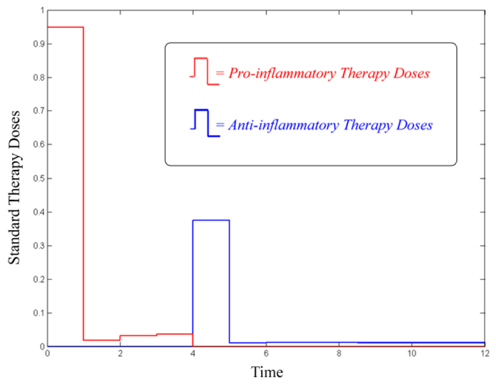


Figure 63: An example of a standard therapy dose profile. This standard dose regimen is administered to all patients within a simulation set, for the purpose of comparing the resulting outcomes with the outcomes from targeted therapy generated specifically for each patient.

The third and final alternate therapy is referred to as *Uber Standard Therapy*, meaning that it is *very* standard, since it does not employ any control-based methods to generate. In fact, this therapy is designed to represent the therapy regimen currently given to critically ill patients with severe inflammatory disorders in the intensive care unit: a consistent dosing regimen of an anti-inflammatory therapy known as *Activated Protein C*. Practically, a dosing profile is created that gives a small dose of the anti-inflammatory therapy (via instantaneous injections, as usual) each hour over a period of 72 hours.

Once an entire therapy dosing profile has been administered to a patient, it is necessary to have some methodology for determining the outcome of an individual simulation. This proved to be another rather technical issue. At the end of the simulation time, how are the outcomes of 1000 different simulations systematically determined and tallied? Initially, these results were based on the values of the variables at the end of 168 hours; however, sometimes this was ambiguous. For example, damage which might be elevated beyond the threshold that we designate as a "healthy" level might also be decreasing toward equilibrium. In such a case, the system might evolve to the healthy steady state, given some more time (in the absence of any more therapeutic intervention).

Thus, it was decided to create a post processing algorithm that takes the ending values of the variables in the patient at 168 hours and integrates the system for another 300 hours, when it would be highly likely that the solution has, by then, settled to a state, whether healthy, aseptic, or septic. A patient outcome is labeled "septic" if the pathogen levels are above a threshold of 1.0, and damage and activated phagocytes are also above their designated thresholds, 1.0 and 0.05, respectively. If pathogen levels are not above threshold, yet damage and activated phagocyte levels are, then the patient outcome is labeled "aseptic," in accordance with the definitions of these physiologic states mentioned earlier. Otherwise, a patient is labeled "healthy." (Note, we do have a check for inconclusive results, however, there have never been any outcomes that fall into this category.)

Hence, in the upcoming results presented for multiple patient simulations, the outcomes are not given graphically, but rather in table format showing percentages of the total treated patient population that fell into the different outcome types. The multiple patient simulations are run on a network of computers known as *PittGrid*, which is a University of Pittsburgh project networking computers from all around campus. Account users submit jobs into a queue, which then sends copies of the files for a specific job to any unused nodes in the network. A special thanks to Senthil Natarajan for making it possible for these NMPC simulations to be run on *PittGrid*. Next, we present the specific results from Configurations 1-4, beginning with Config. 1, in which there is no patient-model mismatch and where only one therapy is considered.

5.4.2 Results: Configuration 1

Recall that in Configuration 1 the following specifications are made:

- No patient-model mismatch
- One Therapy: Anti-inflammatory therapy (no extraction) given on an hourly basis

- Measured patient output variables: D and P
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u} \Delta u\|_2^2 + \|\Gamma_u u\|_2^2$
- $m = 2$ and $h = 24$
- Γ weighting constants: various values explored
- $C_A Max = 0.6264$
- Pathogen growth rate values: Aseptic scenario: $k_{pg} = 0.514$; Septic scenario: $k_{pg} = 0.52$
- Initial conditions for simulations: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$

Figure 64 shows the placebo outcome for an aseptic scenario, where $k_{pg} = 0.514$ and $P_0 = 0.5$. Then, in Figure 65 the NMPC results are shown for weighting constants shown as follows: $\Gamma_D = 1.0; \Gamma_p = 1.0; \Gamma_{\Delta u} = 1.0; \Gamma_u = 1.0$. The control-based therapy that the NMPC algorithm generates for this case is able to turn the aseptic scenario, shown in Figure 64, into a healthy outcome. However, notice that when the damage weight, Γ_D, is significantly increased with respect to the other weights, the result, shown in Figure 66, is not favorable. In this case, the therapy that the algorithm generates, while successful in suppressing damage at least in the beginning stages, turns the aseptic scenario, shown in Figure 64, into septic. Thus, it is important to be aware of the affect that the weights can have on outcome.

Next consider the case when the placebo outcome is a septic scenario (Figure 67) generated with $k_{pg} = 0.52$ and $P_0 = 0.5$. Looking at both Figure 68 and Figure 69, the anti-inflammatory therapy is irrelevant for this scenario, since the system cannot overcome the pathogen. In the first case (Figure 68), the objective function is weighted so that minimizing damage over pathogen is not stressed. Here, the algorithm tries to eliminate pathogen by letting inflammation grow unsuppressed by any additional anti-inflammatory therapy (note the flat line at zero on the C_A dose profile). However, if minimizing damage is a priority (second case: Figure 69), then the algorithm attempts to minimize damage; however, pathogen is then allowed to grow without restriction, and eventually, the inflammation must respond to this, causing the system to arrive (albeit in a slightly delayed fashion) at the septic state.

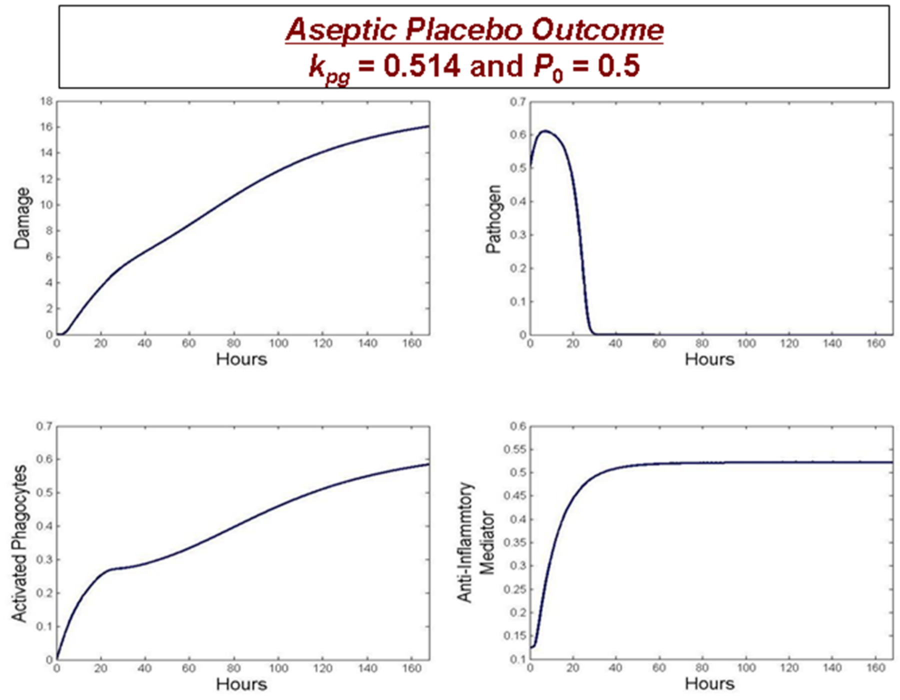


Figure 64: An example of a simulation under Configuration 1 with an aseptic outcome for the placebo case (i.e. without any therapeutic intervention from the NMPC algorithm).

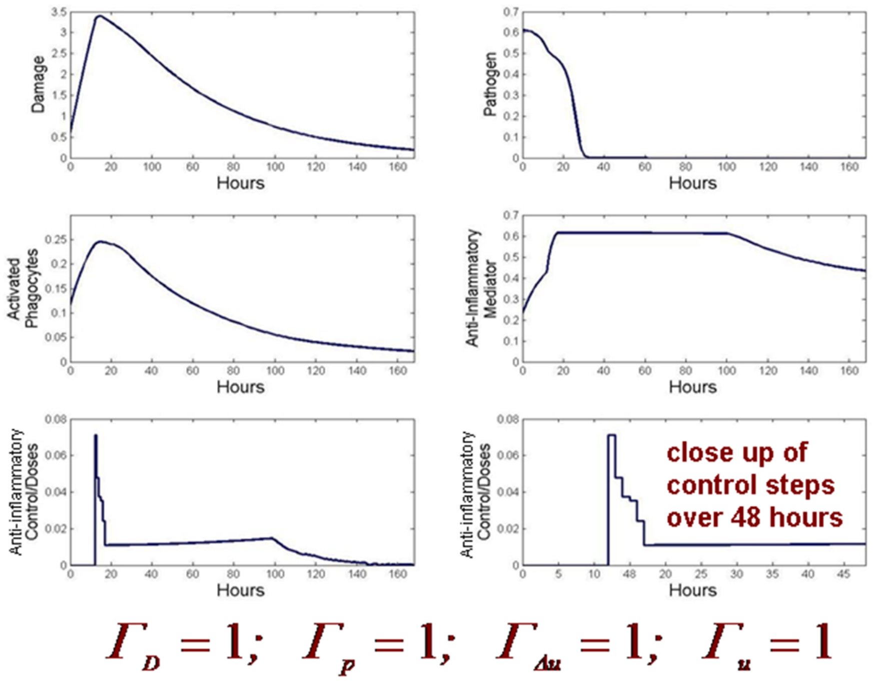


Figure 65: An example of a simulation under Configuration 1, where the aseptic response presented in Figure 64 has been modulated with therapy from the NMPC algorithm. The therapy regimen found is successful to change an otherwise aseptic case into a healthy outcome. The time at which therapy is initiated by the NMPC algorithm was set at 6 hours after the onset of the infection.

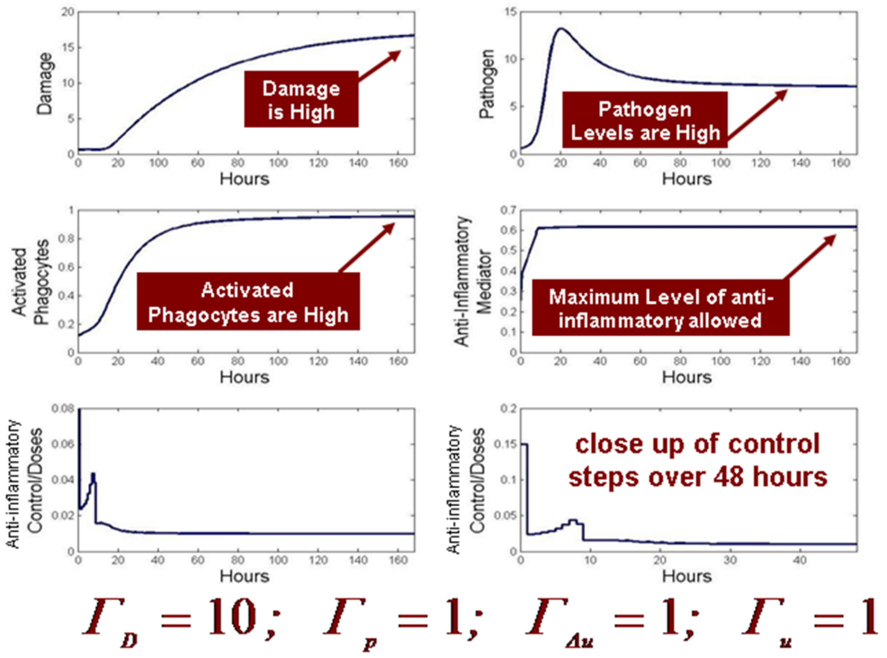


Figure 66: The simulation shown in Figure 65 for the aseptic Placebo case of Figure 64 is rerun with a heavy weight on Damage. The therapy regimen found is unsuccessful to change an otherwise aseptic case into a healthy outcome. Instead the simulation results in a septic outcome, showing that the weighting parameters can be influential in the outcome of a simulation. The heavy weight on damage causes the algorithm to prescribe the anti-inflammatory therapy to supress damage; however, this only postpones the onset of inflammation and the situation is made worse. The time at which therapy is initiated by the NMPC algorithm was set at 6 hours after the onset of the infection.

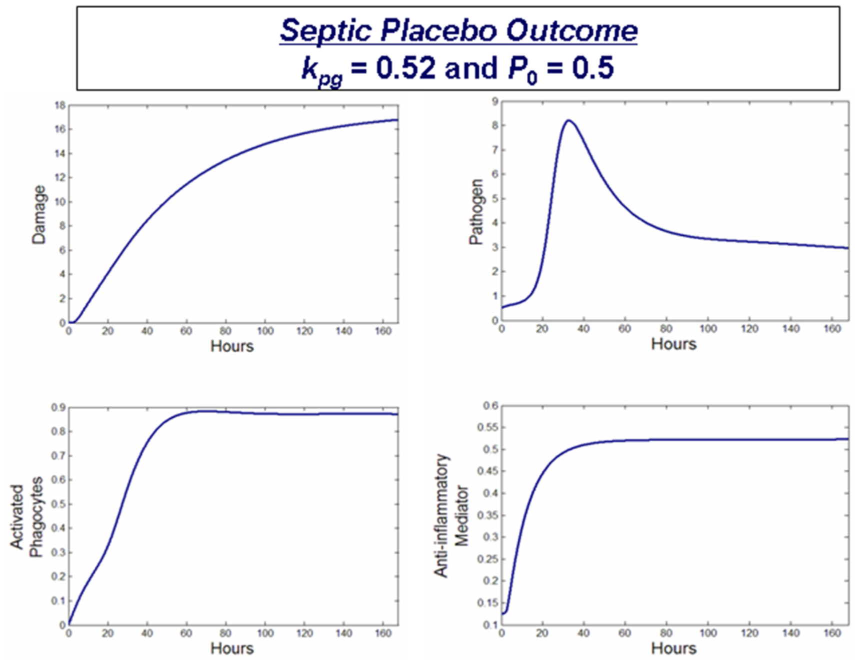


Figure 67: An example of a simulation under Configurations 1 and 2 with a septic outcome for the placebo case (i.e. without any therapeutic intervention from the NMPC algorithm).

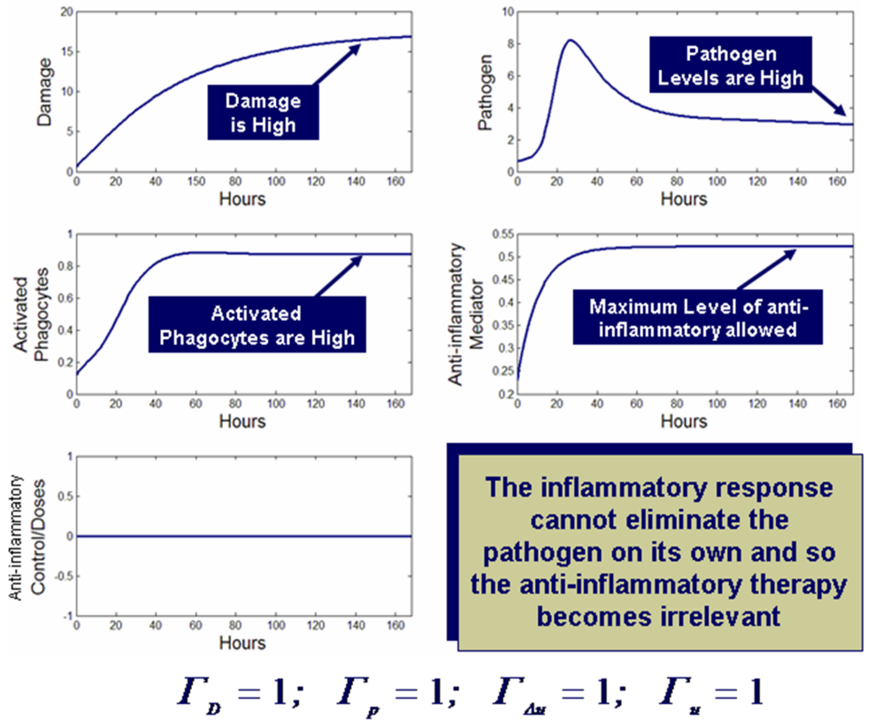


Figure 68: For the septic scenario under Configuration 1 shown in Figure 67, therapeutic intervention was intiated 6 hours after the onset of infection. However, since Configuration 1 only includes the ability to administer anti-inflammatory therapy, the algorithm is unable to find a suitable treatment to resolve the septic scenario to healthy. Essentially, in this case the anti-inflammatory therapy become irrelevant because it does nothing to help the eradication of the pathogen and can only hurt it more if administered, as Figure 69 shows.

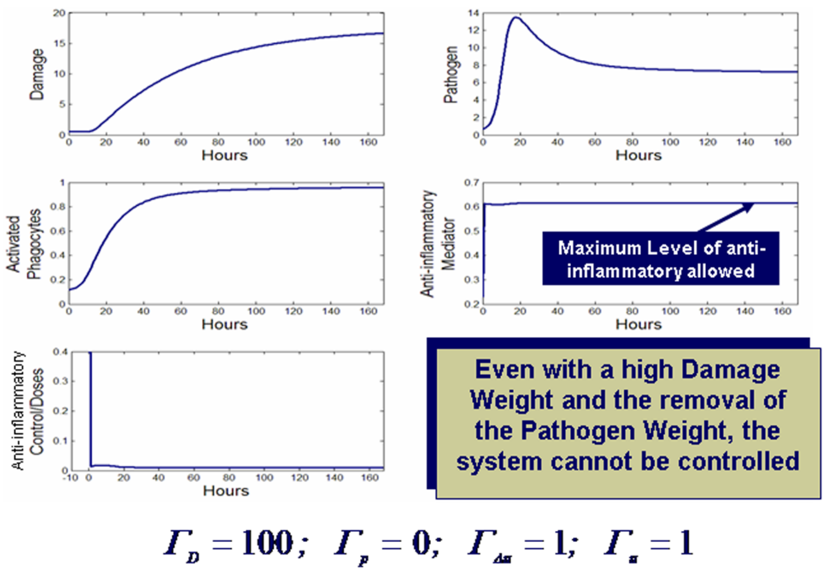


Figure 69: Similar to the simulation run in Figure 68 for the septic scenario under Configuration 1 shown in Figure 67, therapeutic intervention was intiated 6 hours after the onset of infection. However, in this simulation the weight on damage is increased significanlty. This causes the NMPC algorithm to attempt to minimize damage with the anti-inflammatory therapy available. While successful for a while, eventually inflammation grows out of control in response to the unrestricted pathogen growth. This shows the need for another therapy besides the anti-inflammatory therapy.

5.4.3 Results: Configuration 2

Recall that in Configuration 2 the following specifications are made:

- No patient-model mismatch
- Two Therapies: Anti-inflammatory and proinflammatory therapy (no extractions), given on an hourly basis
- Measured patient output variables: D and P
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u1}\Delta u_1\|_2^2 + \|\Gamma_{\Delta u2}\Delta u_2\|_2^2 + \|\Gamma_{u1} u_1\|_2^2 + \|\Gamma_{u2} u_2\|_2^2$
- $m = 2$ and $h = 24$
- Γ weighting constants: various values explored
- $C_A Max = 0.6264$
- $N^* Max = 2.0$
- Pathogen growth rate values: Aseptic scenario: $k_{pg} = 0.514$; Septic scenario: $k_{pg} = 0.52$
- Initial conditions for simulations: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$

Configuration 2 applies two therapies instead of one, but otherwise the configuration is the same as Configuration 1. Thus, the results from the previous section for Configuration 1 can be compared to the results that are now presented for Configuration 2. Only the septic scenario is considered, since the anti-inflammatory therapy alone could not help rectify this case in Configuration 1 above. Figure 67, shows the placebo outcome that results in Configuration 2 in a septic scenario with $k_{pg} = 0.52$ and $P_0 = 0.5$. Figure 70 shows that a two therapy strategy is able to successfully curb the potentially septic scenario and allows the system to resolve to healthy. Since this configuration can restore both septic and aseptic scenarios, it is likely that this setup would restore to health most, if not all, patients who have differing profiles like those generated in subsequent configurations. However, the fact that there is no patient-model mismatch in this configuration makes these potentially nice results less impressive, considering that the presence of mismatch is to be expected in a more real world setting. Hence, in simulations involving a larger patient population, a patient-model mismatch is necessary to better emulate reality.

All the simulations so far initiated therapy 6 hours after the onset of infection. We also take a quick look at what might happen if therapeutic intervention is initiated much later,

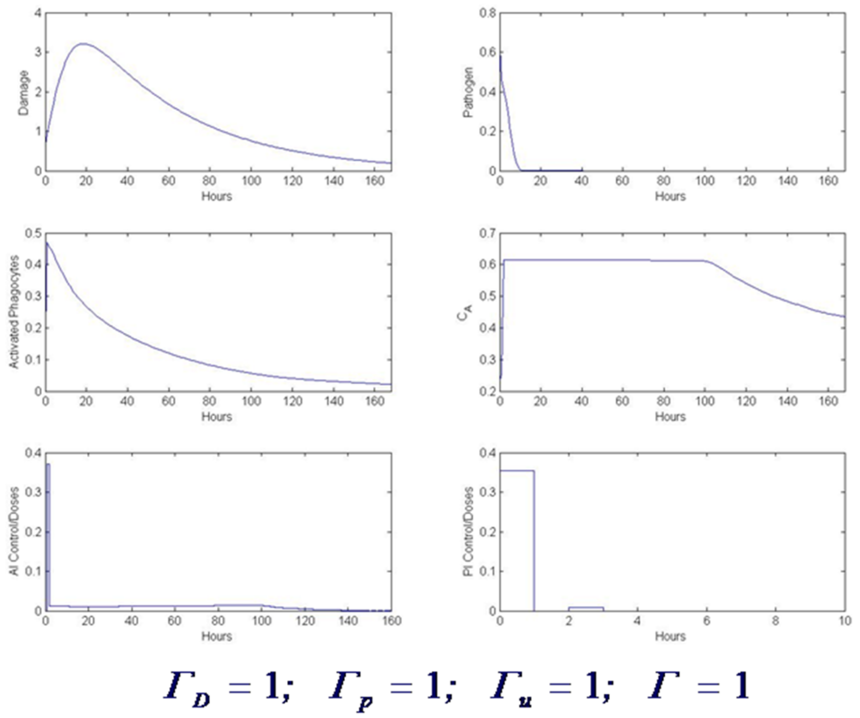


Figure 70: An example of a simulation under Configuration 2, where the septic response presented in Figure 67 has been modulated with therapy fromt the NMPC algorithm. The therapy regimen found is successful to change an otherwise septic case into a healthy outcome. The time at which therapy is initiated by the NMPC algorithm was set at 6 hours after the onset of the infection.

say 40 hours. Figure 71 shows that intervention initiated at this much later time point is able to eradicate pathogen, but inflammation becomes too great by this time for the system to resolve, even with the maximum amount of C_A therapy given over a very lengthy amount of time. Therefore, the success of the algorithm to find appropriate therapies also depends on when intervention is initiated.

5.4.4 Results: Configuration 3

Remark 10. *As a shorthand for referring to therapies and particular details of the different schemes, the following abbreviations will be used for simulation sets of 1000 patients:*

- **NE**: *Both pro- and anti-inflammatory therapies are applied but with No Extraction*
- **WE**: *Bot pro- and anti-inflammatory therapies With Extraction*
- $\mathbf{W \equiv 1}$: *All the weighting parameters of the objective function have a weight equal to* 1.
- **ST**: *Standard Therapy*
- **UST**: *Uber Standard Therapy*
- *[α; β; δ; γ; $\cdots$, ε] : Specifies the weighting parameters when they are are not all equal to* 1. *The order matches the way in which they appear in the objective function used in the particular configuration.*

Recall that in Configuration 3 a multi-patient simulation is carried out and the other following specifications are made:

- Patient-model mismatch
- 1000 patients considered, with differing profiles (i.e. differing values for the parameters shown in Table 9 on page 211)
- Patient profiles are generated from choosing a normally (normal-like) distributed random value from a specified range
- Therapy is initiated in the patient when N^* levels reach a threshold of 0.1
- Two Therapies: Anti-inflammatory and proinflammatory therapy (no extractions), given on an hourly basis
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u1} \Delta u_1\|_2^2 + \|\Gamma_{\Delta u2} \Delta u_2\|_2^2 + \|\Gamma_{u1} u_1\|_2^2 + \|\Gamma_{u2} u_2\|_2^2$

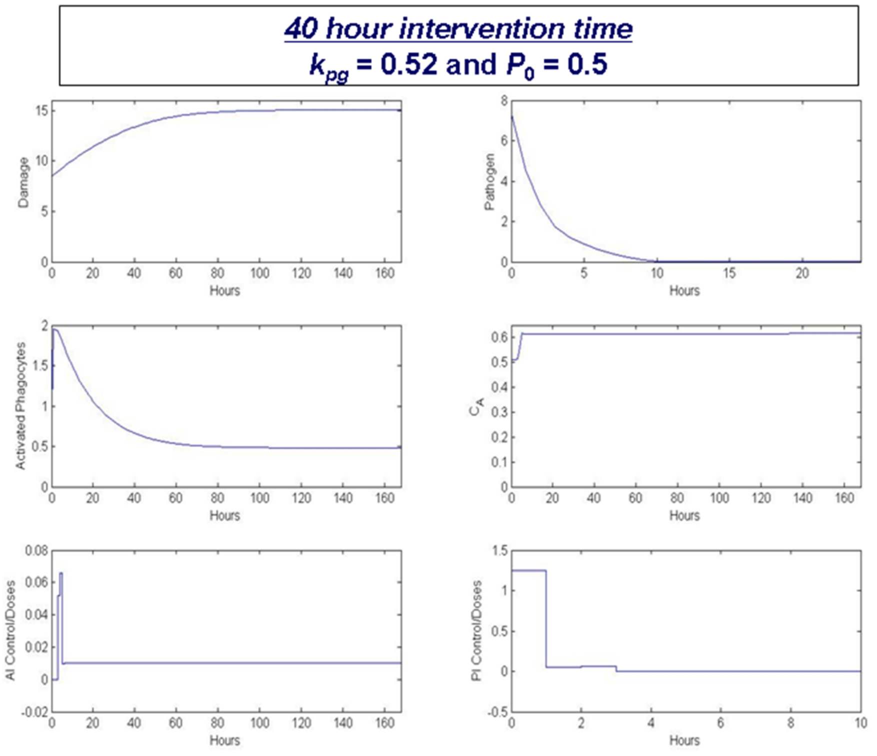


Figure 71: The same simulation that was run for Figure 70, under Configuration 2 in the Placebo septic case, is shown for a different intervention time. Instead of the NMPC algorithm initiating therapy at 6 hours after the onset of infection, the algorithm is configured to wait till 40 hours after the onset of infection. The result is unfavorable, with the patient now ending up with an aseptic condition as opposed to septic.

- Γ-weighting constants: various values explored
- $C_A Max = 0.6264$
- $N^* Max = 1.0$
- Measured patient output variables: N^* and C_A
- Model $k_{pg} = 0.8$
- Model initial conditions: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$
- Patient initial conditions: $(P_0, N^*, D, C_A) = (P_0\text{-random}, 0, 0, C_{A0}\text{-random})$
- $m = 2$ and $h = 24$

The results shown for this configuration include three comparison data sets, in addition to the targeted therapy results. The first two alternative data sets, shown on the same table as the targeted therapy results, are for standard therapy and "4 knob" therapy (i.e. therapy that includes extraction capability). The third alternative therapy data set, shown later and separately, is that of the uber standard therapy. At this stage, we were experimenting with finding proper doses to be administered for the 72 hour period in such a way as to not violate the $C_A Max$ constraint. Hence these results are presented in a separate table, but can still be compared to the other results.

Out of the 1000 patients in the population considered for this configuration, 525 of them reached the N^* threshold of 0.1, after which therapeutic intervention was initiated. The remaining 425 did not receive any treatment; however, percentages are out of the total 1000 patients. Table 11 compares the results of different treatment strategies: 4 therapies (WE), 2 therapies with damage weight emphasized (NE), and 2 therapies with all weights equal to one (NE; W$\equiv$ 1). Also shown with respect to the underlying model's values for k_{pg} and P_0 is the number of patients not rescued with a therapy that

1. had both a higher k_{pg} and P_0 value,
2. had a higher k_{pg} but a lower P_0 value,
3. had a higher P_0 but a lower k_{pg} value, or
4. had both a lower k_{pg} and P_0 value.

Data such as this drew our attention to the fact that perhaps these two parameters were driving the outcomes too much, and that it would be better if the patient profiles were

Table 11: Comparison of Treatment Schemes for Configuration 3

Configuration 3:
Comparison of Treatment Schemes and k_{pg} and P_0 values
No Pathogen Update

575/1000 patients receive treatment; Percentages are out of 1000

	Placebo	*WE* $W \equiv 1$	*WE/ST*	*NE* [1;100 ;1 ;1]	*NE/ST* [1;100 ;1 ;1]	NE $W \equiv 1$	NE/ST $W \equiv 1$
Percentage Rescued	7.3%	57.3%	57.5%	9.8%	20.9%	15.0%	21.3%
Percentage Not Rescued	50.2%	0.2%	0.0%	47.7%	36.6%	42.5%	36.2%
Both Higher k_{pg} and P_0	n/a	0	0	35	31	39	39
Higher k_{pg}	n/a	2	0	36	28	40	38
Higher P_0	n/a	0	0	308	235	264	215
Both Lower	n/a	0	0	98	72	82	70

such that all of the parameters that varied from patient to patient had more significance in determining outcome. Hence, changes are made to the way patient profiles are generated in the next configuration. Also note that, surprisingly, standard therapy does better than the targeted therapy in all of the different simulation types. At the time, it was not yet clear as to why this was the case, since targeted therapy was supposed to be designed for the individual patient; however, in light of recent work, this could have been the result of patient-model mismatch not being dealt with very well. Also, at this stage, the post processing technique was not employed here, and outcomes were only sorted as "rescued" or "not rescued", instead of by the three possible outcomes: septic, aseptic, and healthy. Thus, it may be the case that some of the patient outcomes were mislabeled due to the inaccuracy of the sorting method used at the time.

Nevertheless, these results for Configuration 3 show that the extraction therapy works unrealistically well and that once again the weighting parameters can have a (perhaps unexpected) negative effect. Increasing the pathogen weight was done with the intention that this might help more patients. However, the effect was actually detrimental, with more patients ending up in the "not rescued" category.

In addition to the standard therapy and "4 knob" therapy results, we also administered several uber standard therapies to the patient population. In Table 12, an array of differing doses given every hour over 72 hours is shown. Only one of these did not violate the C_AMax constraint for any of the patients. It is interesting to note that the dose regimen of 0.0008 C_A-therapy/hr for 72 hours was able to rescue all patients, although it was not safe for all patients (in fact, it was only unsafe for 1 patient). So, the dose regimen of 0.0006 C_A-therapy/hr for 72 hours was taken as the safe therapy to administer and, surprisingly, this therapy does just as well as the "4 knob" therapy. However, the fact that this uber standard therapy does better than either of the "2 knob" therapy scenarios and just as well as the "4 knob" therapy gives reason for pause. By this juncture, the need to modify the profiles of the patient population was apparent, as well as the need for a pathogen update and a cap on C_A levels. Thus, we essentially moved into Configuration 4, without troubleshooting the ill-equipped Configuration 3 setup.

Table 12: Uber Standard Therapy Exploration for Configuration 3

Configuration 3: *Uber Standard Therapy (U.S.T.) Exploration*							
575/1000 patients receive treatment; Percentages are out of 1000							
U.S.T. dose per hour for 72 hours	0.0006 Safe for all	0.0008 Too High	0.001 Too High	0.002 Too High	0.0025 Too High	0.005 Too High	0.2 Too High
Percentage Rescued	57.3%	57.5%	7.6%	7.6%	7.6%	8.5%	8.6%
Percentage Not Rescued	2%	0%	49.9%	49.9%	49.9%	49.0%	48.9%

However, before doing so, a direct pathogen update was tested on the patient population from Configuration 3. "Direct" means that the model is synchronized with the patient's pathogen levels at the time it is checked (i.e. measured). We explored a range of time intervals at which to check pathogen levels: every 1, 5, 10 or 20 hours. Table 13 shows the results for these different cases, compared against Standard Therapy results in the first column and Targeted therapy results without a pathogen update in the second column. The larger the time interval (i.e. every 20 hours compared to every hour), the less the direct pathogen update helps, which makes sense. As nice as the direct pathogen update is, it is not realistic and so, as was discussed previously, Configuration 4 includes an indirect pathogen update.

5.4.5 Results: Configuration 4

Recall that in Configuration 4 a multi-patient simulation is carried out, patient profiles are generated differently, a pathogen update is included, a 48 hour cap on elevated C_A levels is enforced, and the following other specifications are made:

- Patient-model mismatch
- 1000 patients considered, with differing profiles (i.e. differing parameter values)
- Patient profiles generated from choosing a uniformly distributed random value from a specified range (ranges shown in Table 10 on page 219)
- N^* threshold = 0.05 (for determining when to initiate therapeutic intervention)

Table 13: Direct Pathogen Update Exploration for Configuration 3

Configuration 3: *Direct Pathogen Update: every 1, 5, 10, or 20 hours* *575/1000 patients receive treatment; Percentages are out of 1000*							
Therapy Type:	Placebo	NE/ST	NE	NE	NE	NE	NE
Pathogen Update?:	n/a	no	no	yes	yes	yes	yes
PU Check Time every x hrs:	n/a	n/a	n/a	1	5 hrs	10hrs	20hrs
Percentage Rescued	7.3%	52.5%	45.4%	57.5%	52.8%	46.5%	45.4%
Percentage Not Rescued	50.2%	5.9%	12.1%	0.0%	4.7%	11.0%	12.1%

- Two Therapies: Anti-inflammatory and proinflammatory therapy (no extractions), given on an hourly basis
- Objective function: $\|\Gamma_D D\|_2^2 + \|\Gamma_p P\|_2^2 + \|\Gamma_{\Delta u1}\Delta u_1\|_2^2 + \|\Gamma_{\Delta u2}\Delta u_2\|_2^2 + \|\Gamma_{u1}u_1\|_2^2 + \|\Gamma_{u2}u_2\|_2^2$, where the following are being minimized: damage levels, D, pathogen levels, P, changes in dosing (Δu_1 and Δu_2) and total therapy given (u_1 and u_2) for both C_A and N^*, respectively.
- Γ-weighting constants: various values explored
- $C_A Max = 0.6264$
- $N^* Max = 1.0$
- Measured patient output variables: N^* and C_A
- Model $k_{pg} = 0.8$
- Pathogen update mechanism
- 48 hour C_A cap
- Model initial conditions: $(P_0, N^*, D, C_A) = (0.5, 0, 0, 0.125)$
- Patient initial conditions: $(P_0, N^*, D, C_A) = (P_0\text{-random}, 0, 0, C_{A0}\text{-random})$
- $m = 2$ and $h = 8$

It would be nice to say that all the new features implemented in this configuration solved the previous problems encountered, and that the results acquired were superb. However, this case is really just another stepping stone in the development and custom-tailoring of the NMPC algorithm. What can be said, is that at this point in the process the algorithm

Table 14: Indirect Pathogen Update and 48 hours C_A Cap Results for Configuration 4

Configuration 4: *INDIRECT Pathogen Update; 48 Hour C_A* maximum Cap				
*620/1000 patients receive treatment; Percentages are out of **620*** *Number of patients out of 620 given in parenthesis.*				
Therapy Type:	Placebo	NE	NE/ST	UST:
Indirect Pathogen Update?:	no	yes	n/a	0.001/hr
PU Check Time every x hrs:	n/a	4 hrs	n/a	for 72 hrs
Percentage Septic:	22.7% (141)	0.0% (0)	0.0% (0)	23.7% (147)
Percentage Aseptic:	36.8% (228)	81.1% (503)	56.6% (351)	35.0% (217)
Percentage Healthy:	40.5% (251)	18.9% (117)	43.4% (269)	41.3% (256)

and subprograms have been refined from previous steps and many of them streamlined to make running and compiling the results more efficient than in the past. Also, many of the technical issues that were a part of the previous configurations, especially #3, have been ironed out. For instance, a post processing routine is implemented so that the outcomes can be clearly classified and divided into categories (healthy, aseptic, septic) that give more information about the results than just "rescued" or "not rescued." Though several of the things mentioned seem minor, altogether they are important for producing and presenting solid results that can be explained well and, if need be, reproduced.

The results for Configuration 4 are perhaps the most disappointing out of all the results, simply because the expectations for this configuration were high. However, it once again shows areas of the algorithm that need attention. Recall that these results are for a new patient population whose parameters are generated differently than before. Table 14 shows the results for this configuration, as well as the placebo outcomes in the first column, the standard therapy results in the third column, and uber standard therapy results in the fourth column.

Targeted therapy does really poorly, compared to all the other treatments, especially placebo. Apparently, the therapy is converting all the septic patients, and a number of the otherwise healthy patients, into aseptic cases. Looking graphically at several patients and the dosing profile generated from them, it was seen that the algorithm was prescribing rather large amounts of the pro-inflammatory therapy at the beginning, in order to eliminate pathogen rapidly; however, inflammation was being profusely generated and the level of the anti-inflammatory mediator even with the addition of anti-inflammatory therapy could not adequately suppress the inflammation. In addition, the 48 hour C_A cap that is enforced in this configuration further prevents the system from controlling the inflammation. Thus, even those patients, who would otherwise be healthy, are harmed by this aggressive pathogen elimination strategy. There are several reason why the algorithm is choosing to do this and these are addressed in the next section where current and future configuration modifications are discussed.

Standard therapy does not do too much better than placebo or uber standard therapy, but once again, it does better than targeted therapy. Still, in this case, it is likely that all the septic cases turned into aseptic cases and some of the aseptic cases were converted to healthy. In fact, consider the sum of the number of septic and aseptic patients in the Placebo column: $141 + 228 = 369$. Also, consider the difference between the number of healthy patients in the Standard Therapy column and the number of healthy patients in the Placebo column: $269 - 251 = 18$. Then the difference between these two quantities is $369 - 18 = 351$, which is the number of aseptic patients in the Standard Therapy column. Thus, it appears that 18 of the aseptic patients became healthy with standard therapy, and all 141 of the septic patients became aseptic with standard therapy. Hence, standard therapy is not a really good therapy, since no septic patients are helped.

The results of Uber Standard Therapy are pretty close to the placebo results. It might be the case that the uber standard therapy that was implemented was not strong enough to exact any major differences from the placebo results. This is another aspect that is in need of modification and refinement.

5.4.6 Preliminary Results: Current/Future Configurations

The results for Configuration 4 brought up still other issues that were of concern. The first modification to be considered after Configuration 4 deals with the pathogen update mechanism and N^*Max. In the strategy described in Configuration 3, an update only occurs if the pathogen levels are high in the patient and low in the model. However, just as important is the opposite case: when pathogen levels are high in the model and low in the patient. Thus, as a new feature in the this current configuration, the model's pathogen levels are set to zero when pathogen levels are elevated in the model and low in the patient. Also, the algorithm previously seemed to be applying the maximum amount of pro-inflammatory therapy, in order to quickly eradicate pathogen. This occurs even in cases where the patient might be able to naturally take care of pathogen or when only a little therapy is needed to boost inflammation enough to eliminate pathogen. An initial step taken simply reduced N^*Max.

These two changes did meet with some success; however, even with this success, if the number of patients who are septic in the placebo scenario are considered, not many of these patients are rescued. Table 15 shows the results in the same format as was presented for Configuration 4. It is mainly the aseptic patients who are helped. In addition, the level at which N^*Max should be set is something that needs further exploration. Here, results for an N^*Max value of 0.1 is shown. Preliminary results suggest that a value of 0.5 would be better, however, this has not been tested on the entire patient population. This was tested on 6 patients that had a variety of placebo outcomes: 1 septic patient, 3 aseptic patients, and 2 healthy patients. (Note that all of these patients did reach the specified N^* threshold of intervention.) With an N^*Max of 0.5, 5 out of the 6 patients resolved to healthy cases.

Another possible angle, with which we slightly experimented, deals with changing the objective function weights on dosing for N^*. While adjusting the weights on dosing for N^* seems like a reasonable course of action to follow, it did not produced the desired effect. First of all, penalizing the changes made to N^* doses, $\Gamma_{\Delta u2}$, only means that drastic changes to dosing are avoided. While this might help the initial N^* dose from being too large, it usually means that the dosing occurs gradually. This, however, turns out to be a

Table 15: Work in Progress Configuration featuring results from the implementation of an indirect 2-way pathogen update

Work in Progress Configuration: *INDIRECT 2-way Pathogen Update; 48 Hour* C_A maximum Cap Reduced N^*Max (0.1)				
620/1000 patients receive treatment; Percentages are out of ***620*** *Number of patients out of 620 given in parenthesis.*				
Therapy Type:	Placebo	NE	NE/ST	UST:
2-Way Pathogen Update?:	no	yes	n/a	0.005/hr
PU Check Time every x hrs:	n/a	4 hrs	n/a	for 72 hrs
Percentage Septic:	22.7% (141)	21.9% (136)	21.1% (131)	23.1% (143)
Percentage Aseptic:	36.8% (228)	17.7% (110)	46.1% (286)	31.5% (195)
Percentage Healthy:	40.5% (251)	60.3% (374)	32.7% (203)	45.5% (282)

worse therapeutic strategy, since in some patients, the pathogen might be strongly driving inflammation, but the response is not strong enough in the beginning stages to be able to overcome it. Instead, the gradual addition of an immune booster only adds to the already accumulated inflammation at a time when the anti-inflammatory levels are also elevated. Preliminary results show that a heavy $\Gamma_{\Delta u2}$ weight cause all 6 of the test patients to end up septic or aseptic, for $N^*Max = 0.5$, for which there was previously success for this group of patients.

The heavy weight not only restricts the treatment from being given too sharply, but also from being sharply cut off. By this time, even though pathogen might be eliminated, both the inflammatory and anti-inflammatory levels are very high, so the addition of an anti-inflammatory therapy puts the patient in the situation described above in Configuration 4 with respect to the 48 hour C_A cap: patients end up aseptic instead of healthy since the amount of inflammation in their system is decreasing due to the C_A cap and the amount of inflammation, which may have been decreasing with the elevated C_A levels, is on the rise once again.

Lastly, the underlying model under the current configuration is such that the algorithm is actually unable to find a successful therapy for it, meaning that the standard therapy generated from the model does not even help the model itself to resolve to healthy, but instead to an aseptic case. However, because of this, the treatment is really aggressive toward eliminating pathogen first and giving much anti-inflammatory therapy afterward, in an attempt to curb the inflammation, at which it is actually unsuccessful. This is the reason why in Table 15 the standard therapy converts septic patients into aseptic patients. Additionally, for those patients who would be aseptic otherwise, the copious amounts of anti-inflammatory therapy prescribed by the standard therapy actually help some aseptic patients to transfer to the healthy camp. Therefore, it is probably the case that the underlying model should at least generate a therapy from which the model itself actually benefits. In order for this to happen under the current configuration, the model's k_{pg} and P_0 values will have to be modified. In fact, the current k_{pg} value of 0.8 in the model is outside of the range chosen for the individual patients' k_{pg} values, which was an oversight in this process.

Overall, the differences in the results between the cases where patient-model mismatch does not exist and the cases where it does exist are strikingly apparent. There are an entirely different set of issues and difficulties that arise because of mismatch. Hence, there are still other configurations to be found that are more effective at dealing with the balance between minimizing damage and pathogen levels and making the algorithm more sensitive to patient-model mismatch.

5.5 DISCUSSION

The various configurations that have been explored thus far show that there are many facets to the application of NMPC for finding proper therapies and dosing regimens for correcting immune dysfunction. It remains to be seen whether the changes mentioned above in the current/future configuration section will bring about more favorable results when applied to a large patient population. There are also other things that need to be reexamined, such as the size of the move horizon, m, and prediction horizon, h, as well as the objective function weights (Γ_*) in order to make sure that these tuning parameters are chosen thoughtfully in light of the many changes that the algorithm has undergone.

It was seen that only the four therapy module (briefly discussed in Configuration 4) was really able to help significantly in converting both potentially aseptic and septic patients into healthy ones. On the other hand, it appears that the two therapy cases are limited in their ability to balance the objectives between minimizing damage and pathogen, objectives that are at odds with one another. It might be the case that the options given to the algorithm with respect to therapy types are either too many (e.g. the four therapy case) or are too limited for the current model, meaning that the system, in some sense, might be too simple either way for the application of NMPC. More success might be found if it is applied to a larger model where there is a one-to-one correspondence between variables in the model and mediators of the inflammatory response. For example, in such models, there is more than one anti-inflammatory mediator and so changes to one may not have such drastic effects on the entire system as is seen in this smaller model. On the other hand, it might be much more difficult to troubleshoot any issues of the algorithm for such a large model.

There are many sub programs in addition to the main NMPC algorithm that were developed to take care of the various tasks along the way. These include programs to generate patient profile data, process and compile patient outcomes after the NMPC data is calculated, and to administer alternate therapies, along with other routines for visualization and troubleshooting. The development phase and corresponding results discussed in the previous sections were the culmination of a year long process which brought the idea of a controller based therapy from a conceptual phase into an experimental design phase. The results presented here are not as successful as what is desired and many challenges of this approach have been exposed. Nonetheless, the idea of using a control-based algorithm to generate appropriate therapy regimens only makes more sense in light of the fact that finding appropriate, successful therapies under precarious conditions is a nontrivial and difficult venture.

6.0 CONCLUSION

The research presented in this manuscript covers a wide array of topics pertaining to the process of understanding and controlling the acute inflammatory response. Even the work presented in Chapters 3 and 4, which can be considered apart from the context of inflammation, was motivated by this area of research. The chapters are strung together with this common thread, and many interesting results as well as challenging problems have been reached.

The work in Chapter 2 gives a strong argument supporting the viewpoint that endotoxin tolerance and potentiation are characteristics of the dynamics of the acute inflammatory response. It highlights the importance of the timing of the different mediators involved in the response and how preconditioning can shift the system into a very different state compared to the state of the non-preconditioned system.

The desire to graphically illustrate these differing states, with respect to the threshold that exists between the healthy and unhealthy states, led to the stable manifold code implementation in Chapter 3. Although the method to generate the manifold mesh is due to Krauskopf and Osinga [63], many parts of the implementation of it are not explicitly a part of their paper. Hence, the task of creating the computer code is quite a formidable one, with many challenges.

Although the manifold program presented has some shortcomings, as far as we know, there is not yet a program that exists for use in the academic community. Therefore, this MatLab implementation might be the first published code of a program for generating 2D (un)stable manifolds of 3D ODE systems. The topic of numerically generating a manifold is interesting enough to, in the future, continue refinement of the current program or to implement other methods that have been published in the literature, which do not have a workable code available.

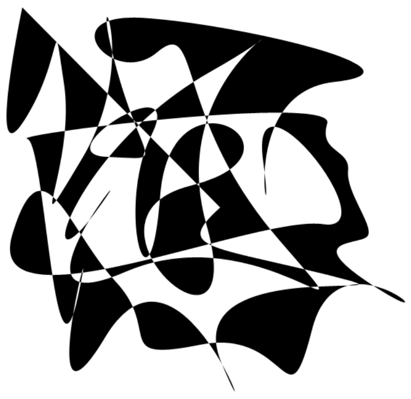
In addition to Chapter 2 inspiring the work on manifolds, it also brought to light the transient nature of tolerance. The magnitude of the tolerance reduction is very much dependent on the levels of the various mediators and the timing of the precondition dose(s). Figure 13 nicely shows this transient behavior. This aspect of the endotoxin tolerance research led to the work in Chapter 4, where tolerance-like behavior is studied from a purely dynamical systems point of view.

Chapter 4, which contains perhaps the most interesting results (at least mathematically speaking), presents a novel research topic, regarding when the tolerance behavior can and cannot occur in a given differential equations system. A plethora of original results are presented and in the case of 2D linear systems, in particular, the results give a completely systematic way to identify exactly where tolerance will occur. In the case of 2D nonlinear systems, the methods for identifying tolerance are not as exact as the linear case. However, the methods for 2D nonlinear systems are a creative use of the combination of isoclines and the concept of inhibition and can narrow down the possibilities of where tolerance can occur in a system.

Most of the results are restricted to the two-dimensional case, so this area of research has room for expansion and further generalization. For example, knowing for which initial conditions tolerance will exist does not imply that the time at which tolerance is exhibited is known. For instance, in the linear case, it can be determined for a given $<(x_0, y_0), s>$ whether or not there exists a $\tau > 0$, such that $\psi_1(\tau) < \phi_1(\tau)$, but the value of τ remains unknown. Also, results are based on shifting the graph of $\phi(t)$ by the amount x_0 in the x-direction and not shifting the graph at all in the y-direction. A more generic "hit" size in both directions could be considered. In particular, the linear results rely on the assumption that the shift was only in the x-direction and only by the amount x_0. Hence, there are several different angles from which to further explore this area of research in the future.

Lastly, Chapter 5 presents a preliminary, but thorough, exposition on the use of nonlinear model predictive control (NMPC) as a tool for generating and implementing immunomodulation strategies and therapies. A majority of the work regarding modeling the acute inflammatory response has focussed on understanding the dynamics of the interactions between mediators and how they relate to the different outcomes. Here, the application of NMPC is explored for the purpose of *controlling* the inflammatory response. NMPC has not before been applied for the purpose of controlling systemic inflammation via immunomodulation strategies. As was discussed in Chapter 5, there are many challenges that exist, and this research area is still very much an ongoing one.

As a whole, this dissertation contains a breadth of results that are both interesting and insightful for the mathematical community, as well as for those in systems biology. Bridging the gap between the creation and use of mathematical/engineering tools and their application in the clinical realm is an immense challenge. The work presented here and that to be conducted in the future is focussed on continuing this bridging process, keeping in mind that today's ideas are tomorrow's realities.



APPENDIX

GLOSSARY OF ABBREVIATIONS

- **ABM**: Agent Based Model
- **AIDose**: Anti-inflammatory Dose
- **C$_A$**: Anti-inflammatory mediator (Cytokine - Anti-inflammatory)
- **D**: Damage
- **EVC**: Eigenvector Configuration
- **HIV**: Human Immunodeficiency Virus
- **HMG-1**:
- **IL-10**: Interleukin-10
- **IL-12**: Interleukin-12
- **IL-6**: Interleukin-6
- **LPS**: Lipopolysaccharide
- **MPC**: Model Predictive Control
- **N***: Activated Phagocytes (i.e. Neutrophils)
- **NE**: No Extraction
- **NMPC**: Nonlinear Model Predictive Control
- **ODE**: Ordinary Differential Equations
- **P**: Pathogen
- **PE**: Pathogen Endotoxin
- **PIDose**: Proinflammatory Dose
- **PU**: Pathogen Update

- $\mathbb{R}^+$: The set of positive real numbers
- **RHS**: Right Hand Side
- **ST**: Standard Therapy
- **TGF-**β1: Transforming Growth Factor-β1
- **TNF**: Tumor Necrosis Factor
- **UST**: Uber Standard Therapy
- **WE**: With Extraction
- **W**$\equiv 1$: All Weighting parameters of the objective function are identically equal to 1
- **WLOG**: Without Loss Of Generality
- $\mathbf{W}_s(x_0)$: Global stable manifold of the fixed point, x_0
- $\mathbf{W}^s_{loc}(x_0)$**:** Local stable manifold of the fixed point, x_0
- $\mathbf{W}_u(x_0)$: Global unstable manifold of the fixed point, x_0
- $\mathbf{W}^u_{loc}(x_0)$: Local unstable manifold of the fixed point, x_0

BIBLIOGRAPHY

[1] Abraham, R., and Shaw, C. *Dynamics - the Geometry of Behavior, Part Three: Global Behavior*. Aerial Press, Santa Cruz, 1982-1985.

[2] Alexander, C., and Rietschel, E. T. Bacterial lipopolysaccharides and innate immunity. *J.Endotoxin.Res. 7*, 3 (2001), 167–202.

[3] Alves-Rosa, F., Vulcano, M., Beigier-Bompadre, M., Fernandez, G., Palermo, M., and Isturiz, M. A. Interleukin-1beta induces in vivo tolerance to lipopolysaccharide in mice. *Clin.Exp.Immunol. 128*, 2 (2002), 221–228.

[4] An, G. Agent-based computer simulation SIRS: Building a bridge between basic science and clinical trials. *Shock 16* (2001), 266–273.

[5] An, G. In-silico experiments of existing and hypothetical cytokine-directed clinical trials using agent based modeling. *Crit Care Med 32* (2004), 2050–2060.

[6] Andersson, U., Wang, H., Palmblad, K., Aveberger, A. C., Bloom, O., Erlandsson-Harris, H., Janson, A., Kokkola, R., Zhang, M., Yang, H., and Tracey, K. J. High mobility group 1 protein (hmg-1) stimulates proinflammatory cytokine synthesis in human monocytes. *J.Exp.Med. 192*, 4 (2000), 565–570.

[7] Babior, B. Phagocytes and oxidative stress. *Am J Med 109* (2000), 33–44.

[8] Bacon, G. E., Kenny, F. M., Murdaugh, H. V., and Richards, C. Prolonged serum half-life of cortisol in renal failure. *Johns.Hopkins.Med.J. 132*, 2 (1973), 127–131.

[9] Balkhy, H. H., and Heinzel, F. P. Endotoxin fails to induce ifn-gamma in endotoxin-tolerant mice: deficiencies in both il-12 heterodimer production and il-12 responsiveness. *J.Immunol. 162*, 6 (1999), 3633–3638.

[10] Beeson, P. B. Tolerance to bacterial pyrogens: I. factors influencing its development. *J.Exp.Med. 86* (1947), 29–38.

[11] Berg, D. J., Kuhn, R., Rajewsky, K., Muller, W., Menon, S., Davidson, N., Grunig, G., and Rennick, D. Interleukin-10 is a central regulator of the response to lps in murine models of endotoxic shock and the shwartzman reaction but not endotoxin tolerance. *J.Clin.Invest 96*, 5 (1995), 2339–2347.

[12] Bocci, V. Interleukins. clinical pharmacokinetics and practical implications. *Clin.Pharmacokinet. 21*, 4 (1991), 274–284.

[13] Bone, R. The search for a magic bullet to fight sepsis. *JAMA 269*, 17 (1993), 2221–2227.

[14] Bone, R. Towards a theory regarding the pathogenesis of the systemic inflammatory response syndrome: What we do and do not know about cytokine regulation. *Crit Care 24*, 1 (1996), 163–172.

[15] Bone, R. Why sepsis trials fail. *JAMA 276* (1996), 565–566.

[16] Buchman, T., Cobb, J., Lapedes, A., and Kepler, T. Complex systems analysis : A tool for shock research. *Shock 16*, 4 (2001), 248–251.

[17] Bumiller, A., Gotz, F., Rohde, W., and Dorner, G. Effects of repeated injections of interleukin 1beta or lipopolysaccharide on the hpa axis in the newborn rat. *Cytokine 11*, 3 (1999), 225–230.

[18] Cavaillon, J. M. The nonspecific nature of endotoxin tolerance. *Trends Microbiol. 3*, 8 (1995), 320–324.

[19] Cavaillon, J. M., Pitton, C., and Fitting, C. Endotoxin tolerance is not a lps-specific phenomenon: partial mimicry with il-1, il-10 and tgf-beta. *J Endotoxin Res 1* (1994), 21–29.

[20] Chow, C. C., Clermont, G., Kumar, R., Lagoa, C., Tawadrous, Z., Gallo, D., Betten, B., Bartels, J., Constantine, G., Fink, M. P., Billiar, T. R., and Vodovotz, Y. The acute inflammatory response in diverse shock states. *Shock 24*, 1 (2005), 74–84.

[21] Clermont, G., Bartels, J., Kumar, R., Constantine, G., Vodovotz, Y., and Chow, C. In silico design of clinical trials: a method coming of age. *Crit Care Med. 32*, 10 (2004), 2061–2070.

[22] Clermont, G., Chow, C., Constantine, G., Vodovotz, Y., and Bartels, J. Mathematical and statistical modeling of acute inflammation. In *Classification, Clustering, and Data Mining Applications* (2004), Springer, pp. 457–467.

[23] Copeland, S., Warren, H. S., Lowry, S. F., Calvano, S. E., and Remick, D. Acute inflammatory response to endotoxin in mice and humans. *Clin.Diagn.Lab Immunol. 12*, 1 (2005), 60–67.

[24] Council, N. R. Animal models for use in detecting immunotoxic potential and determining mechanisms of action. In *Biologic Markers in Immunotoxicology.* National Academy Press, Washington, 1992, p. 95.

[25] Coxon, A., Tang, T., and Mayadas, T. N. Cytokine-activated endothelial cells delay neutrophil apoptosis in vitro and in vivo. a role for granulocyte/macrophage colony-stimulating factor. *J.Exp.Med. 190*, 7 (1999), 923–934.

[26] Cross, A., and Opal, S. A new paradigm for the treatment of sepsis: Is it time to consider combination therapy? *Ann Intern Med. 138*, 6 (2003), 502–505.

[27] Cross, A. S. Endotoxin tolerance-current concepts in historical perspective. *J.Endotoxin.Res. 8*, 2 (2002), 83–98.

[28] Day, J., Rubin, J., Vodovotz, Y., Chow, C., Reynolds, A., and Clermont, G. A reduced mathematical model of the acute inflammatory response II. capturing scenarios of repeated endotoxin administration. *J Theor Biol 242*, 1 (2006), 237–256.

[29] Degryse, B., Bonaldi, T., Scaffidi, P., Muller, S., Resnati, M., Sanvito, F., Arrigoni, G., and Bianchi, M. E. The high mobility group (hmg) boxes of the nuclear protein hmg1 induce chemotaxis and cytoskeleton reorganization in rat smooth muscle cells. *J.Cell Biol. 152*, 6 (2001), 1197–1206.

[30] Dellnitz, M., and Hohmann, A. *The Computation of Unstable Manifolds Using Subdivision and Continuation.* Birkhäuser, Basel, 1996, pp. 449–459.

[31] Dellnitz, M., and Hohmann, A. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numer. Math. 75* (1997), 293–317.

[32] Doedel, E., Keller, H., and Kernévez, J. Numerical analysis and control of bifurcation problems:. *Int. J. Bifurcation and Chaos 1*, 3 (1991), 493–520.

[33] Doedel, E., Keller, H., and Kernévez, J. Numerical analysis and control of bifurcation problems: II. *Int. J. Bifurcation and Chaos 1*, 4 (1991), 745–772.

[34] Edelstein-Keshet, L. *Mathematical Models in Biology.* SIAM, 2004.

[35] Eng, R. H., Smith, S. M., Fan-Havard, P., and Ogbara, T. Effect of antibiotics on endotoxin release from gram-negative bacteria. *Diagnostic Microbiology and Infectious Disease 16*, 3 (1993), 185–189.

[36] Ermentrout, B. *Simulating, analyzing, and animating dynamical systems: A guide to XPPAUT for researchers and students.* Soc for Industrial and Applied Math, Philadelphia, 2002.

[37] Florian, J., June 2006. (Personal Communication).

[38] Florian, J. J., Eiseman, J., and Parker, R. Nonlinear model predictive control for dosing daily anticancer agents: A tamoxifen treatment of breast cancer case study. (to appear).

[39] Food and Drug Administration. Innovation or stagnation: Challenge and opportunity on the critical path to new medical products, 2004.

[40] Fuchs, A. C., Granowitz, E. V., Shapiro, L., Vannier, E., Lonnemann, G., Angel, J. B., Kennedy, J. S., Rabson, A. R., Radwanski, E., Affrime, M. B., Cutler, D. L., Grint, P. C., and Dinarello, C. A. Clinical, hematologic, and immunologic effects of interleukin-10 in humans. *J.Clin.Immunol. 16*, 5 (1996), 291–303.

[41] Ginè, J., and Grau, M. Characterization of isochronous foci for planar analytic differential systems. *Proceedings of the Royal Society of Edinburgh 135A* (2005), 985–998.

[42] Grutz, G. New insights into the molecular mechanism of interleukin-10-mediated immunosuppression. *J.Leukoc.Biol. 77*, 1 (2005), 3–15.

[43] Guckenheimer, J., and Homes, P. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Applied Mathematical Sciences Volume 42. Springer-Verlag, New York, 1983.

[44] Guckenheimer, J., and Vladimirsky, A. A fast method for approximating invariant manifolds. *SIAM J. Appl. Dyn. Sys 3*, 3 (2004), 232–260.

[45] Guckenheimer, J., and Worfolk, P. *Dynamical Systems: Some Computational Problems.* Kluwer Academic Publishers, 1993, pp. 241–277.

[46] Henderson, M. Computing invariant manifolds by integrating fat trajectories. Tech. Rep. RC22944, IBM Research, 2003.

[47] Hobson, D. An efficient method for computing invariant manifolds of planar maps. *J. Comput. Phys. 104*, 1 (1993), 14–22.

[48] Howard, M., Muchamuel, T., Andrade, S., and Menon, S. Interleukin 10 protects mice from lethal endotoxemia. *J.Exp.Med 177*, 4 (1993), 1205–1208.

[49] Huhn, R. D., Radwanski, E., Gallo, J., Affrime, M. B., Sabo, R., Gonyo, G., Monge, A., and Cutler, D. L. Pharmacodynamics of subcutaneous recombinant human interleukin-10 in healthy volunteers. *Clin.Pharmacol.Ther. 62*, 2 (1997), 171–180.

[50] Isler, P., de Rochemonteix, B. G., Songeon, F., Boehringer, N., and Nicod, L. P. Interleukin-12 production by human alveolar macrophages is controlled by the autocrine production of interleukin-10. *Am.J.Respir.Cell Mol.Biol. 20*, 2 (1999), 270–278.

[51] Iversen, M. H., and Hahn, R. G. Acute effects of vitamin a on the kinetics of endotoxin in conscious rabbits. *Intensive Care Med 25*, 10 (1999), 1160–1164.

[52] J., G., and Llibre, J. A family of isochronous foci with darbouz first integral. *Pacific journal of mathematics 218*, 2 (2005), 343–355.

[53] Jaeschke, H., and Smith, C. Mechanisms of neutrophil-induced parenchymal cell injury. *J Leukoc Biol 61* (1997), 647–653.

[54] Janeway, C. A., J., and Medzhitov, R. Innate immune recognition. *Annu.Rev.Immunol. 20* (2002), 197–216.

[55] Janeway, C. A., J., Travers, P., Walport, M., and Shlomchik, M. *Immunobiology: The immune system in health and disease.* Garland Publishing, New York, 2001.

[56] Jarrar, D., Chaudry, I., and Wang, P. Organ dysfunction following hemorrhage and sepsis: Mechanisms and therapeutic approaches. *Int J Mol Med 4* (1999), 575–583.

[57] Johnson, M., and Billiar, T. Roles of nitric oxide in surgical infection and sepsis. *World J Surg 22* (1998), 187–196.

[58] Johnson, M., Jolly, M., and Kevrekidis, I. Two-dimensional invariant manifolds and global bifurcations: Some approximation and visualization studies. *Num. Alg. 14*, 1-3 (1997), 125–140.

[59] Kariko, K., Weissman, D., and Welsh, F. A. Inhibition of toll-like receptor and cytokine signaling–a unifying theme in ischemic tolerance. *J.Cereb.Blood Flow Metab 24*, 11 (2004), 1288–1304.

[60] Keel, M., Schregenberger, N., Steckholzer, U., Ungethum, U., Kenney, J., Trentz, O., and Ertel, W. Endotoxin tolerance after severe injury and its regulatory mechanisms. *J.Trauma 41*, 3 (1996), 430–437.

[61] Kirschner, D., and Wigginton, J. A model to predict cell-mediated immune regulatory mechanisms during human infection with mycobacterium tuberculosis. *J Immunol 166*, 3 (2001), 1951–1967.

[62] Krauskopf, B., and Osinga, H. Growing 1d and quasi-2d unstable manifolds of maps. *Journal of Computational Physics 146* (1998), 404–419.

[63] Krauskopf, B., and Osinga, H. Two-dimensional global manifolds of vector fields. *Chaos 9*, 3 (1999), 768–774.

[64] Krauskopf, B., Osinga, H., Doedel, E., Henderson, M., Guckenheimer, J., Vladimirsky, A., Dellnitz, M., and Junge, O. A survey of methods for computing (un)stable manifolds of vector fields. *Int. J. Bifurcation and Chaos 15*, 3 (2005), 763–791.

[65] Kumar, R., Clermont, G., Vodovotz, Y., and Chow, C. C. The dynamics of acute inflammation. *J.Theor.Biol. 230*, 2 (2004), 145–155.

[66] Lagoa, C. E., Chow, C., Bartels, J., Barrat, A., Kumar, R., Day, J., Rubin, J., Constantine, G., Chang, S., Fink, M. P., Billiar, T. R., Clermont, G., and Vodovotz, Y. Mathematical models predict the course of the inflammatory response in rats subjected to trauma-hemorrhagic shock, and to anti-tumor necrosis factor alpha therapy in endotoxemia (abstract). *J Crit.Care 20*, 4 (2005), 393–394.

[67] Leon, P., Redmond, H. P., Shou, J., and Daly, J. M. Interleukin 1 and its relationship to endotoxin tolerance. *Arch.Surg. 127*, 2 (1992), 146–151.

[68] Letterio, J. J., Vodovotz, Y., Bogdan, C., Henderson, B., and Higgs, G. Tgf-beta and il-10: Inhibitory cytokines regulating immunity and the response to infection. In *Novel Cytokine Inhibitors*. Birkhäuser Verlag, Basel, 2000, pp. 217–242.

[69] Li, N. The dynamics of the vocal fold inflammatory and wound healing in responses to the biomechanical stresses associated with phonotrauma. (submitted).

[70] Marshall, J. Inflammation, coagulopathy, and the pathogenesis of multiple organ dysfunction syndrome. *Crit Care Med 29* (2001), S99–S106.

[71] Marshall, J., Deitch, E., Moldawer, L., Opal, S., Redl, H., and Poll, T. Preclinical models of shock and sepsis: What can they tell us? *Shock 24*, suppl 1 (2005), 1–6.

[72] MathWorks, I. MATLAB: The Language of Technical Computing, version 7.0.4.365 (r14) service pack, 2005.

[73] MathWorks, I. *Optimization Toolbox: Fmincon.* Natick, MA, 2005.

[74] Matzinger, P. The danger model: a renewed sense of self. *Science 296*, 5566 (2002), 301–305.

[75] Mayer, H., Zaenker, K. S., and an der Heiden, U. A basic mathematical model of the immune response. *Chaos 5*, 1 (1995), 155–161.

[76] Mendez, C., Kramer, A. A., Salhab, K. F., Valdes, G. A., Norman, J. G., Tracey, K. J., and Carey, L. C. Tolerance to shock: an exploration of mechanism. *Ann Surg. 229*, 6 (1999), 843–849.

[77] Mi, Q., Riviere, B., Clermont, G., Steed, D., and Vodovotz, Y. Agent-based modeling of inflammation and wound healing: Insights into diabetic foot ulcer pathology and the role of transforming growth factor-beta. *Wound Repair and Regeneration* (2007), (to appear).

[78] Morari, M., and Ricker, N. *Model Predictive Control Toolbox.* The MathWorks, Inc., Natick, MA, 1998.

[79] Morrison, D. C., and Ryan, J. L. Endotoxins and disease mechanisms. *Annu.Rev.Med. 38* (1987), 417–432.

[80] Murray, J. *Mathematical Biology: I. An Introduction*, vol. 1. Springer-Verlag, 2002.

[81] Murray, J. *Mathematical Biology: II. Spatial Models and Biomedical Applications*, vol. 2. Springer-Verlag, 2003.

[82] Nathan, C. Points of control in inflammation. *Nature 420*, 6917 (2002), 846–852.

[83] Nathan, C., and Hibbs, J. J. Role of nitric oxide synthesis in macrophage antimicrobial activity. *Curr Opin Immunol 3* (1991), 65–70.

[84] Nathan, C., and Sporn, M. Cytokines in context. *J Cell Biol. 113*, 5 (1991), 981–986.

[85] Nemzek, J., Xiao, H., Minard, A., Bolgos, G., and Remick, D. Humane endpoints in shock research. *Shock 21* (2004), 17–25.

[86] Ogunnaike, B., and Ray, W. *Process Dynamics, Modeling, and Control.* Oxford University Press, Oxford; New York, 1994.

[87] Osinga, H. Two-dimensional invariant manifolds in four-dimensional dynamical systems. *Computers and Graphics 29*, 2 (2005), 289–297.

[88] Panetta, J. A mathematical model of breast and ovarian cancer treated with paclitaxel. *Math. Biosci. 146* (1997), 89–113.

[89] Parker, R., Doyle, F. r., and N.A., P. The intravenous route to blood glucose control. *IEEE Eng Med Biol Mag 20* (2001), 65–73.

[90] Parker, S. J., and Watkins, P. E. Experimental models of gram-negative sepsis. *Br.J Surg. 88*, 1 (2001), 22–30.

[91] Parrillo, J. E. Pathogenetic mechanisms of septic shock. *N.Engl.J.Med. 328*, 20 (1993), 1471–1477.

[92] Perelson, A. Modeling the interaction of HIV with the immune system. In *Mathematical and Statistical Approaches to AIDS Epidemiology* (New York, 1989), C. Castillo-Chavez, Ed., vol. 83 of *Lect. Notes in Biomath.*, Springer-Verlag, pp. 350–370.

[93] Pinsky, M. R. Sepsis: a pro- and anti-inflammatory disequilibrium syndrome. *Contrib.Nephrol.*, 132 (2001), 354–366.

[94] Pinsky, M. R. Dysregulation of the immune response in severe sepsis. *Am.J Med Sci. 328*, 4 (2004), 220–229.

[95] Randow, F., Syrbe, U., Meisel, C., Krausch, D., Zuckermann, H., Platzer, C., and Volk, H. D. Mechanism of endotoxin desensitization: involvement of interleukin 10 and transforming growth factor beta. *J.Exp.Med 181*, 5 (1995), 1887–1892.

[96] Rayhane, N., Fitting, C., and Cavaillon, J. M. Dissociation of ifn-gamma from il-12 and il-18 production during endotoxin tolerance. *J.Endotoxin.Res. 5*, 5-6 (1999), 319–324.

[97] Reynolds, A., Clermont, G., and Ermentrout, G. Modeling the effect of acute inflammation on gas exchange. *J Crit Care 21*, 4 (2006), 354.

[98] Reynolds, A., Rubin, J., Clermont, G., Day, J., and Ermentrout, G. A reduced mathematical model of the acute inflammatory response: I. derivation of the model and analysis of anti-inflammation. *J Theor Bio 242*, 1 (2006), 220–236.

[99] Rubinow, S. *Introduction to Mathematical Biology.* John Wiley, 1975.

[100] Sabatini, M. Isochronus sections via normalizers. Matematica UTM 659, University of Trento, February 2004.

[101] Sanchez-Cantu, L., Rode, H. N., and Christou, N. V. Endotoxin tolerance is associated with reduced secretion of tumor necrosis factor. *Arch.Surg. 124*, 12 (1989), 1432–1435.

[102] Schade, F. U., Flach, R., Flohe, S., Majetschak, M., Kreuzfelder, E., Dominguez-Fernandez, E., Borgermann, J., Obertacke, U., Brade, H., Morrison, D. C., Opal, S., and Vogel, S. Endotoxin tolerance. In *Endotoxin in Health and Disease.* Marcel Dekker, New York, 1999, pp. 751–767.

[103] Schlag, G., and Redl, H. Mediators of injury and inflammation. *World J Surg 20* (1996), 406–410.

[104] Senaldi, G., Shaklee, C. L., Guo, J., Martin, L., Boone, T., Mak, T. W., and Ulich, T. R. Protection against the mortality associated with disease models mediated by tnf and ifn-gamma in mice lacking ifn regulatory factor-1. *J.Immunol. 163*, 12 (1999), 6820–6826.

[105] Sly, L. M., Rauh, M. J., Kalesnikoff, J., Song, C. H., and Krystal, G. Lps-induced upregulation of ship is essential for endotoxin tolerance. *Immunity. 21*, 2 (2004), 227–239.

[106] Spivak, M. *Differential Geometry*, 2nd ed. Publish or Perish, Houston, TX, 1979.

[107] Stoiser, B., Knapp, S., Thalhammer, F., Locker, G., Kofler, J., Hollenstein, U., Staudinger, T., Wilfing, A., Frass, M., and Burgmann, H. Time course of immunological markers in patients with the systemic inflammatory response syndrome: Evaluation of sCD14, sVCAM-1, sELAM-1, MIP-1 alpha and TGF-beta. *Eur J Clin Invest 28* (1998), 672–678.

[108] Strogatz, S. H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering.* Westview Press, Cambridge, 1994.

[109] Tjardes, T., and Neugebauer, E. Sepsis research in the next millenium: Concentrate on the software rather than the hardware. *Shock 17*, 1 (2002), 1–8.

[110] Vodovotz, Y. Deciphering the complexity of acute inflammation using mathematical models. *Immunol Res 36*, 1-3 (2006), 237–245.

[111] Vodovotz, Y., Barcellos-Hoff, M. H., Salvemini, D., and Billiar, T. R. Direct and indirect modulation of the inducible nitric oxide synthase by nitric oxide: Feedback mechanisms in inflammation. In *Nitric Oxide and Inflammation*. Birkhäuser-Verlag, Basel, 2001, pp. 41–58.

[112] Vodovotz, Y., Chow, C., Bartels, J., Lagoa, C., Prince, J., Levy, R., Kumar, R., Day, J., Rubin, J., Constantine, G., Billiar, T., Fink, M., and Clermont, G. In silico models of acute inflammation in animals. *Shock 26*, 3 (2006), 235–244.

[113] Vodovotz, Y., Clermont, G., Chow, C., and An, G. Mathematical models of the acute inflammatory response. *Curr Opin Crit Care 10* (2004), 383–390.

[114] Vodovotz, Y., Kim, P., Bagci, E., Ermentrout, G., Chow, C., Bahar, I., and Billiar, T. Inflammatory modulation of hepatocyte apoptosis by nitric oxide: In vivo, in vitro, and in silico studies. *Curr Mol Med 4*, 7 (2004), 753–762.

[115] Vogel, S. N., Kaufman, E. N., Tate, M. D., and Neta, R. Recombinant interleukin-1 alpha and recombinant tumor necrosis factor alpha synergize in vivo to induce early endotoxin tolerance and associated hematopoietic changes. *Infect.Immun. 56*, 10 (1988), 2650–2657.

[116] Wang, H., Bloom, O., Zhang, M., Vishnubhakat, J. M., Ombrellino, M., Che, J., Frazier, A., Yang, H., Ivanova, S., Borovikova, L., Manogue, K. R., Faist, E., Abraham, E., Andersson, J., Andersson, U., Molina, P. E., Abumrad, N. N., Sama, A., and Tracey, K. J. Hmg-1 as a late mediator of endotoxin lethality in mice. *Science 285*, 5425 (1999), 248–251.

[117] Warner, A. E., DeCamp, M. M., J., Molina, R. M., and Brain, J. D. Pulmonary removal of circulating endotoxin results in acute lung injury in sheep. *Lab Invest 59*, 2 (1988), 219–230.

[118] Weisstein, E. W. Basin of attraction, 2006. MathWorld–A Wolfram Web Resource.

[119] West, M. A., and Heagy, W. Endotoxin tolerance: A review. *Crit Care Med. 30*, 1 Supp (2002), S64–S73.

[120] Wysocka, M., Robertson, S., Riemann, H., Caamano, J., Hunter, C., Mackiewicz, A., Montaner, L. J., Trinchieri, G., and Karp, C. L. Il-12 suppression during experimental endotoxin tolerance: dendritic cell loss and macrophage hyporesponsiveness. *J.Immunol. 166*, 12 (2001), 7504–7513.

[121] Xing, Z., Gauldie, J., Cox, G., Baumann, H., Jordana, M., Lei, X. F., and Achong, M. K. Il-6 is an antiinflammatory cytokine required for controlling local or systemic acute inflammatory responses. *J Clin.Invest 101*, 2 (1998), 311–320.

[122] Yadavalli, G. K., Auletta, J. J., Gould, M. P., Salata, R. A., Lee, J. H., and Heinzel, F. P. Deactivation of the innate cellular immune response following endotoxic and surgical injury. *Exp.Mol.Pathol. 71*, 3 (2001), 209–221.

[123] Yoshida, M., Roth, R. I., and Levin, J. The effect of cell-free hemoglobin on intravascular clearance and cellular, plasma, and organ distribution of bacterial endotoxin in rabbits. *J.Lab Clin.Med 126*, 2 (1995), 151–160.

[124] Zamora, R., and Vodovotz, Y. Transforming growth factor-beta in critical illness. *Crit Care Med. (in press)* (2005).

[125] Zervos, E. E., Kramer, A. A., Salhab, K. F., Norman, J. G., Carey, L. C., and Rosemurgy, A. S. Sublethal hemorrhage blunts the inflammatory cytokine response to endotoxin in a rat model. *J.Trauma 46*, 1 (1999), 145–149.