universität wien

DISSERTATION

Titel der Dissertation

Optimization Techniques for Error Bounds of ODEs

Verfasserin

Mag. Qaisra Fazal

angestrebter akademischer Grad

Doktor der Naturwissenschaften (Dr.rer.nat)

Wien, June 2011

Studienkennzahl lt. Studienblatt:	A 091 405
Studienrichtung lt. Studienblatt:	Mathematik
Betreuer:	Prof. Dr. Arnold Neumaier

Abstract

Error bounds of initial value problems with uncertain initial conditions are traditionally computed by using interval analysis but with limited success. Traditional analysis only leads to asymptotic error estimates valid when the maximal step size tends to zero, while efficiency in the approximation requires that step sizes are as large as possible without compromising accuracy. Recent progress in global optimization makes it feasible to treat the error bounding problem as a global optimization problem. This is particularly important in the case where the differential equations or the initial conditions contain significant uncertainties. A new solver DIVIS (Differential Inequality based Validated IVP Solver) has been developed to compute the error bounds of initial value problems by using defect estimates and optimization techniques. The basic idea is to compute the defect estimates of initial value problems by using outer ellipsoidal approximation. The validated state enclosures are computed by applying differential inequalities. Convergence of the method depends upon a suitable choice of preconditioner.

The scheme is implemented in MATLAB and AMPL and the resulting enclosures are compared with VALENCIA-IVP, VNODE-LP and VSPODE.

Abstract

Fehlerschranken von Anfangswertproblemen mit unbestimmten Anfangsbedingungen werden herkmmlicherweise mit Hilfe von Intervallanalysis berechnet, allerdings mit migem Erfolg. Die traditionelle Herangehensweise fhrt zu asymptotischen Fehlerabschtzungen, die nur gltig sind, wenn die maximale Schrittweite gegen Null geht. Jedoch bentigt eine effiziente Approximation grtmgliche Schrittweiten, ohne die Genauigkeit zu mindern. Neue Entwicklungen in der globalen Optimierung ermglichen es, das Finden von Fehlerschranken als globales Optimierungsproblem aufzufassen. Das ist insbesondere wichtig im Fall, dass die Differentialgleichungen oder die Anfangsbedingungen bedeutende Unschrfen enthalten.

Es wurde ein neuer Solver - DIVIS (Differential Inequality based Validated IVP Solver) - entwickelt, um die Fehlerschranken fr Anfangswertprobleme mit Hilfe von Fehlerabschtzungen und Optimierungstechniken zu berechnen. Die Idee dabei ist, die Fehlerabschtzung von Anfangswertproblemen durch elliptische Approximation zu berechnen. Die validierten Zustandseinschliessungen werden mit Hilfe von Differentialungleichungen berechnet. Die Konvergenz dieser Methode hngt von der Wahl geeigneter Vorkonditionierer ab.

Das beschriebene Schema wurde in MATLAB und AMPL implementiert. Die Ergebnisse wurden mit VALENCIA-IVP, VNODE-LP und VSPODE verglichen.

DEDICATED TO

MY PARENTS

FAZAL DIN & **FAIZ ELLAHI**

&

MY SONS

MUHAMMAD SADIM SADIQ
&
MUHAMMAD AYAAN SADIQ

Acknowledgment

Perusing my research work is both painful and enjoyable experience. It is just like climbing a high peak, step by step, accompanied with bitterness, destitution, disappointment, encouragement, support, trust and with so many people's kind lending of my hand. When I found myself with a compiled copy of my thesis, I realized that it was in fact a teamwork that could make it complete. Though it will not be ample to convey my gratitude in words to all those who helped me, I would still like to give many many thanks to all of the following:

First of all, I am grateful to Allah Almighty, all praises be to Him Who gave me vision to advancement and showed me light to enlighten my mind and heart during this work. This effort will guide me to the pathway of my success throughout my life.

I feel great pride in being a Muslim and from the ummah of the Holy prophet Hazrat Muhammad PBUH. His life history is a perfect code of conduct for me to progress forward with sanity and conviction in this era of competitiveness.

The most important individual in this thesis is my honorific supervisor Prof. Arnold Neumaier who brought me where I am today. I hereby acknowledge with all depth of emotions that the road of success towards the completion of my project would never have been so clearly visible as it was with his precise, focused, sincere and untiring support.

I need to thank my mother Faiz Ellahi and my father Mian Fazal Din (May their souls rest in peace). It was due to my mother's kind, god fearing and humble nature that I could possibly develop a character as a strong women. I can never forget her praying hands for my success, prosperity and contentment. Without her prayers, it was not possible for me to reach up to this ladder of success. Of course, it was my father's worldly wisdom which enabled me to deal with any problem as a professional, in a rational way.

I am truly indebted to my husband Tanveer Iqbal, who while doing PhD at the same time had given me complete support in raising our kids and sharing responsibilities of our family. I can never forget the hard time that we had seen together, in which he always stood out and supported me.

I need to mention my kids especially my elder son Muhammad Sadim Sadiq whose time was always compromised due to my workload. His innocent smile used to refresh me after a complete day of work. My younger son Muhammad Ayaan Sadiq is almost passing through the similar life cycle.

I would like to give my sincere thanks to my brothers Arshad Naeem, Kamran Fazal, my sisters Saira Fazal and Humaira Fazal, my nephews Ali Raza, Shoaib Sarwar and Taimoor Sarwar who have always been there just a call away.

Many thanks to my teacher Prof. Hermann Schichl whom positive attitude would always be a source of inspiration for me to take along for all my professional life to come.

I am also thankful to my teacher Prof. Oleg Shcherbina for his guidance remarkably in AMPL modeling.

I would like to thank my M.Phil thesis supervisor Prof. Shahid Siddiqi for teaching me the theory of spline functions and the differential equations.

Special thanks to Dr. Johannes Ploss and our Dean Prof. Harald Rindler for their support concerning some administrative and organizational issues.

I am thankful to my referee Prof. John D. Pryce. His valuable comments and suggestions helped me to improve the presentation of final draft.

I would like to thank Prof. N.S. Nedialkov, Prof. Mark A. Stadtherr and Andreas Rauh for providing me their solvers VNODE-LP, VSPODE and VALENCIA-IVP and their prompt responses for my quiries. Special thanks to Andreas Rauh for his positive critisim and value able suggestion on my preliminary draft sent to him.

I owe a few but very important people, whose limitless gratitude for always being there for me in whatever capacity I needed, are my dear collegue Saptarshi Das, my near and dear friends: Ghazala Akram, Rozeia Mustafa, Sobia janjua, Moumita, Saadia, Astrid, Sigit, my very nice collegues from the Computational Mathematics Group Wien: Ferenc Domes, Mihaly Markot, Erich Dolejsi, Martin Fuchs, Peter Schodl and all other friends and colleagues from both countries for their kind support.

Last but not the least I have to mention Higher Education Commission Pakistan for generating all the resources which enabled me to fulfill this goal.

Qaisra Fazal
Vienna
June, 2011.

Contents

Abstract ii

Dedication iv

Acknowledgment v

1 Introduction 1

1.1 Literature Survey 2

1.2 Contributions 3

1.3 Introduction to Thesis 5

2 Existing methods 7

2.1 Preliminaries 7

2.2 AWA(Anfangswertaufgabe) 9

2.3 VNODE (Validated Numerical ODE) 11

2.4 VNODE-LP 14

2.5 VSPODE (Validating Solver for Parametric ODEs) 15

2.6 VALENCIA-IVP 17

2.7 Kühn's rigorous error bounds 19

3 Chernousko's error bounds for linear ODEs 23

3.1 Reachable sets 23

3.2 Ellipsoidal approximation of reachable sets 25

3.2.1 Affine transformation 26

3.2.2 Approximating the convex sets 27

3.3 Sum of ellipsoids 27

3.3.1 Simultaneous diagonalization of quadratic forms 28
3.4 Ellipsoidal approximations for dynamic systems 33

4 Ellipsoidal bounds using differential inequalities 39
4.1 Conditional differential inequalities 39
4.2 The generalized singular value decomposition 45
4.3 Ellipsoidal bounds without control 47
4.4 Ellipsoidal approximation with control 51

5 Error bounds by optimization 57
5.1 Error bounds by optimization 57
5.1.1 Approximating the solution 58
5.1.2 Error bounds for the preconditioned ODEs 59
5.2 Algorithms 61
5.3 Implementation 71
5.4 Graphical representation 73

6 Numerical results 75
6.1 Examples 76
6.1.1 Example 1 76
6.1.2 Example 2 76
6.1.3 Example 3 78
6.1.4 Example 4 79
6.1.5 Example 5 80
6.1.6 Example 6 81
6.1.7 Example 7 82
6.1.8 Example 8 84
6.1.9 Example 9 86
6.2 Conclusion 87

7 Perspectives and future work 93
7.1 Rounding error control 93
7.2 Solving the system of ODEs with uncertain parameters 94

7.3 BVPs 94
7.4 Generalization towards differential algebraic equations 95
7.5 Automatic choice of parameters 96

Appendix A: Samples of AMPL files **97**
AMPL model 97
AMPL data 100
AMPL run file 102

Appendix B: The DIVIS package **105**
Installation and use 105
Headers of MATLAB routines 108

Curriculum Vitæ **141**

Chapter 1

Introduction

While the approximate solution of differential equations has a long and successful history, much less is known about the accuracy achieved by the computed approximations. Traditional analysis only leads to asymptotic error estimates when the maximal step size tends to zero, while the efficiency in approximation requires step sizes as large as possible without compromising accuracy.

Computable error bounds are traditionally achieved by means of interval analysis, because it is often the case that the parameters and/or the initial values are not known with certainty but are given as intervals. Hence traditional methods do not apply to the resulting parametric ordinary differential equations since they would have to solve infinitely many systems and differential inequalities in simple (inverse monotone) cases, with limited success only.

Recent progress in global optimization makes it feasible to treat the error bounding problem as a global optimization problem. This is particularly important in the case where the differential equations or the initial conditions contain significant uncertainties.

Ordinary differential equations (ODEs) have many applications in engineering and science. If the solution of a system of ODEs satisfies certain conditions at one point of the independent variable, the problem is known as initial value problem (IVP) and if such conditions are satisfied at more than one places of independent variable, then it is known as boundary value problem (BVP). When these conditions are satisfied at two extreme points of independent variable, the problem is known as two-point BVP [48]. In this thesis, we will concentrates on computing the validated enclosures of IVP(see section 7.3 for some remarks about BVP).

1.1 Literature Survey

Bounding the solution of ordinary differential equations in the context of interval analysis was first discussed by MOORE [61, 63]. A computer program was designed to determine the intervals, containing the exact solution to ordinary differential equations [66, 58, 93, 94]. Then in [62], a technique was presented to reduce the wrapping effect produced by the successive expansions in Taylor series with interval remainder [61, 93]. See also [64, 59, 60, 65, 12] in this regard. KRÜCKEBERG [42] introduced a method that computed in each step an inclusion of the solution with arbitrary fixed initial value. Then he computed an inclusion of the perturbation of the solution due to the variation of the solution. HUNGER [34], first computed an approximate solution and then bounded a linear differential system for the error function. MARCOWITZ also computed an approximate solution. He used the system of non-linear differential inequalities in order to bound the error function. JACKSON [35] exposed some flaws in Moore's and Krückeberg's algorithms and also discussed some problems for which these algorithms worked well. EIJGENRAAM [26] developed an algorithm for the solution of IVP that did not have exact initial values but contained in a given initial value set. He gave the first rigorous overestimation analysis for an enclosure. See also [6, 99, 10, 9, 22, 24, 32, 68, 72, 70, 74, 75, 76, 81, 77, 78, 79, 82, 84, 83, 104, 106, 105, 100]

Moore's algorithms have been refined in subsequent years and implemented in software packages such as AWA [51, 53, 54, 101, 23], VNODE [69], VNODE-LP [67, 71], VSPODE [50, 49], VODESIA [25], VALENCIA-IVP [92, 90], ADIODES [102], COSY INFINITY [8, 55, 57, 56] etc.

Historically, AWA was one of the most important packages to compute guaranteed enclosures for the solution of ordinary IVP. This was written in Pascal-XSC, and was presented by LOHNER. This is a one step method and proves existence and uniqueness of the solution by using Picard iteration and gives a rough enclosure of the solution. In order to compute correct enclosures of the solution, a mean value method and the Taylor expansion on a variational equation is applied on global errors. Wrapping effect is reduced by applying coordinate transformations and intersecting different enclosures. The efficiency of AWA is affected by Algorithm I which is limited to the Euler step size.

VNODE (Validated Numerical ODE) is a C++ package for computing rigorous bounds on the solution of an IVP for an ordinary differential equation. This method is based on an interval Hermite-Obreschkoff method [73] for performing Algorithm II. VNODE-LP is a successor of the VNODE package and has been developed by using Literate programming and is easy to implement as compared to VNODE.

VSPODE (Validated Solver for Parametric Ordinary Differential Equations) a C++ package, is based on traditional interval methods but uncertainty in

parameters and initial data are dealt by using Taylor models.

VALENCIA-IVP (VALidation of state ENClosures using Interval Arithmetic for Initial Value Problems) is a C++ package that computes guaranteed state enclosures for IVPs of dynamical systems. The basic idea is first to calculate non-validated approximate solutions of initial value problems and then using these solutions, fixed-point iterative scheme is implemented to compute guaranteed error bounds. Later on, its performance was improved [91]. By using this solver, a template based tool SMARTMOBILE(Simulation and modeling of dynamics in mobile) [2, 3, 4] is introduced to choose suitable arithmetic for a certain model in multibody system. problems and set of criteria to make a fair comparison of

ADIODES (Automatic Differentiation Interval Analysis Ordinary Differential Equation Solver) proves the uniqueness and existence of periodic solutions of specific ordinary differential equations by using Picard iteration and the extended mean value method to compute the enclosures. The building blocks of ADIODES are interval package BIAS/PROFIL [37], FADBAD/TADIFF. These automatic differentiation (AD) packages TADIFF [14] FADBAD and FADBAD++ [13] are used in VNODE, VNODE-LP and VSPODE to compute the Taylor coefficients of the solution of ODE and the solution of its variational equation while in VALENCIA-IVP [92], they evaluate Jacobian of ODE.

COSY INFINITY is an object oriented beam dynamics package which is based on differential algebraic (DA) methods and Taylor model methods. Existence and uniqueness of a solution is verified by using the Picard iteration together with fixed point theorem. Tight enclosures are computed by reducing influence of overestimation. For this purpose, Taylor expansion in time and initial conditions is applied. In order to reduce the wrapping effect resulting from the overestimation, Taylor polynomials with real floating-point coefficients and guaranteed error bounds for interval remainder terms are computed.

1.2 Contributions

This thesis contributes to computes the error bounds of IVPS of ODEs in the following ways:

Theoretical Achievements

- A new scheme has been developed to compute error bounds of initial value problems by using defect estimates and optimization techniques. This scheme is based on an algorithm for enclosing the solutions of IVPs for ODEs which depends only on Theorem 4.1.3 and the argu-

ments described in chapter 5. KÜHN [43] presented a technique for how to compute the error bounds by estimating the defects of IVPs and their corresponding variational and adjoint equations. He estimated these defects by computing higher order derivatives. We have developed a technique to compute the error bounds of IVPs for ODEs by approximating the defect estimates but in a quite different way. We use optimization techniques rather than computing higher order derivatives. For that purpose, we solve optimization problems in AMPL using solver IPOPT.

- In earlier version of our scheme, we used variational equations of the system of ODEs as a preconditioner. We computed the error bounds of ODEs by using differential inequality presented in Theorem 4.1.1 and 4.1.2. But this scheme did not produce sharp bounds for higher dimensions.

- To resolve this problem, we have developed a new theory in Theorem 4.1.3 assuming a conditional differential inequality that is more powerful than the traditional approach. The quality of bounds is improved by constructing a new precontioner. CHERNOUSKO in [20, 16, 18, 19, 17] presented techniques to approximate the reachable sets of controlled linear dynamic systems by using two sided ellipsoidal approximation: outer ellipsoid of minimal volume and the inner ellipsoid of maximal volume. But we used only outer ellipsoidal approximation to construct a new preconditioner for nonlinear dynamic systems. Our initial box is an ellipsoid of the form

$$\|U_0(y_0 - u_0))\|^2 \leq \Delta_0,$$

where $y_0 \in \mathbb{R}^n$, $u_0 = \text{mid}(\boldsymbol{y}_0)$, $\boldsymbol{y}_0 \in \mathbb{IR}^n$, $U_0 \in \mathbb{R}^{n\times n}$ is chosen as

$$U_0 = \text{Diag}\,(\overline{y}_0 - \underline{y}_0)^{-1}$$

and $\mathbb{R} \ni \Delta_0 = n/4$, n is the dimension of system of ODEs (see Lemma 5.1.1). This Δ_0 is used as starting defect estimate.

- We have also reproduced Chernousko's scheme in chapter 4, by using differential inequality to compute the error bounds of dynamic systems both with control and without control.

Development of a validated ODE solver DIVIS

- Based on our theory to compute the error bounds of IVPs for ODEs, a new solver DIVIS (Differential Inequality based Validated IVP Solver) has been developed using MATLAB and AMPL. For convenience, most of the structure is automatized. User's input includes three MATLAB files consisting of a system of ODEs, initial conditions and time span.

User also has to choose suitable regularization factors for the precontioner and the step length that still needs to be automatized. Flow diagram of the solver structure is presented in section 5.4.

- The technique of computing the defect estimates is explained by a list of algorithms presented in section 5.2.

- Step by step implementation is described in section 5.3. To compare our resulting enclosures with those computed by existing solvers VNODE-LP, VSPODE and VALENCIA-IVP, a number of MATLAB routines is created. For each problem, these routines automatically generate new C++ files required for these solvers to compute the error bounds.

1.3 Introduction to Thesis

In chapter 2, we will discuss existing validated ODE solvers VALENCIA-IVP, VNODE, VNODE-LP and VSPODE in some detail to explore techniques that have been used to compute validated state enclosures of ODEs. The presentation is given in a uniform format unless some additional parameters are introduced.

In chapter 3, we will discuss Chernousko's technique to compute error bounds of controlled dynamic systems by means of two sided ellipsoidal approximation. For that purpose, a review of some part of Chernousko [16], especially sections $5, 6$ and 8 is presented.

Chapter 4 and 5 contain the results about new approach. These results are based on conditional differential inequalities and are core of the thesis.

Earlier version of our scheme was based on theory of Theorems 4.1.1 and 4.1.2 that are described in section 4.1 of chapter 4. An improved version of this theory is presented in Theorem 4.1.3 that is more efficient for adaptive usage. Then we discuss generalized singular value decomposition. Theory of ellipsoidal approximation of dynamic systems both with and without control. sections 4.3- 4.4.

The arguments presented in section 5.1 are used to develop our new scheme. The techniques used for computing the defect estimates are described by a list of algorithms given in section 5.2. Step by step implementation of our scheme is explained in section 5.3 whereas a flow diagram of the presented scheme is shown in section 5.4.

In chapter 6, numerical results of the new method are shown. The resulting enclosures of ODEs are compared with existing solvers VALENCIA-IVP, VNODE-LP and VSPODE.

Chapter 7 will give an overview of future plan to improve the efficiency of

DIVIS.

A sample of optimization problem consisting of a model, data and run files required to compute defect estimates in AMPL, is shown in Appendix A.

The installation and use of DIVIS package along with a list of headers of MATLAB routines is given in Appendix B.

Chapter 2

Existing methods

Traditionally, validated methods use interval techniques to compute the guaranteed bounds enclosing the true solution of IVPs for ODEs. For that purpose, a number of validated ODE solvers have been developed including AWA, VNODE, VNODE-LP, VALENCIA, VSPODE. In this chapter we will discuss these solvers and will explore the schemes that are implemented to compute the validated enclosures. We will also discuss the schemes presented by Chernousko and Kuhn to computed state enclosures of solution of IVPs in ODEs.

2.1 Preliminaries

In this section, we will discuss some important terminologies required to explain the techniques used in existing solvers.

QR decomposition

A QR decomposition of a real nonsingular $m \times n$ matrix A is given by

$$A = QR,$$

where $Q \in \mathbf{R}^{m \times m}$ is an orthogonal matrix (that is $Q^T Q = QQ^T = I$) and $R \in \mathbf{R}^{m \times n}$ is an upper triangular matrix [28, p. 223] . More generally, a complex $m \times n$ matrix A, with $m \geq n$ can be factorized as the product of an $m \times m$ unitary matrix Q (that is $Q^* Q = QQ^* = I$,, Q^* is the conjugate transpose of Q) and an $m \times n$ upper triangular matrix R. As the bottom $(m - n)$ rows of an $m \times n$ upper triangular matrix consist entirely of zeros, it is often useful to partition R, or both R and Q.:

$$A = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1, Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

where R_1 is an $n \times n$ upper triangular matrix, Q_1 is $m \times n$, Q_2 is $m \times (m-n)$, and Q_1 and $Q_s 2$ both have orthogonal columns. factorization of A.

Mean-value form

If $F : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable on $D \subseteq \mathbb{R}^n$ and $\boldsymbol{a} \subseteq D$, then for any y and $b \in \boldsymbol{a}$,

$$F(y) \in F_m(\boldsymbol{a}, b) := F(b) + F'(\boldsymbol{a})(\boldsymbol{a} - b) \tag{2.1}$$

The expression $F(b) + F'(\boldsymbol{a})(\boldsymbol{a} - b)$ is called the mean value form of F [63, p. 47].

Automatic Differentiation

Automatic differentiation (AD), sometimes alternatively called algorithmic differentiation, is a technique to evaluate the derivatives of a function defined by a computer program.

Dependence Problem

If an interval occurs several times in a calculation using parameters, and each occurrence is taken independently then this can lead to an unwanted expansion of the resulting intervals known as *dependence problem*. For example, computing $F(\boldsymbol{y}) = \boldsymbol{y}^2 + \boldsymbol{y}$ without the power rules where $\boldsymbol{y} = [-1, 1]$, results into

$$\boldsymbol{y}^2 + \boldsymbol{y} = [-1, 1]^2 + [-1, 1] = [0, 1] + [-1, 1] = [-1, 2].$$

The correct results is $[-1/4, 2]$. This excess width is dependence problem [33]. If an interval variable appears only once in an expression no widening of the interval occurs. This problem can be resolved by reformulating function F as $F(\boldsymbol{y}) = \left(\boldsymbol{y} + \frac{1}{2}\right)^2 - \frac{1}{4}$. So the suitable interval calculation is

$$\left([-1, 1] + \frac{1}{2}\right)^2 - \frac{1}{4} = \left[-\frac{1}{2}, \frac{3}{2}\right]^2 - \frac{1}{4} = \left[0, \frac{9}{4}\right] - \frac{1}{4} = \left[-\frac{1}{4}, 2\right]$$

Taylor Models

A Taylor model of a function F on some interval $\boldsymbol{y}$ consists of the Taylor polynomial T_n of order n of F and an interval remainder term I_n, which encloses the approximation error $|F - T_n|$ on $\boldsymbol{y}$. In computations that involve F, the function is replaced by $T_n + I_n$. The polynomial part is propagated by symbolic calculations where possible. The interval remainder term is processed

according to the rules of interval arithmetic. All truncation and roundoff errors in intermediate operations are also enclosed into the remainder interval of the final result [80].

Consider the initial value problem (IVP)

$$y'(t) \;=\; F(t,y), \quad y(t_0) \in \boldsymbol{y}_0, \qquad t \in [t_0, t_m] \tag{2.2}$$

with $t_0, t_m \in \mathbb{R}$, $F \in C^{p-1}(D)$, $p \geq 2$, $D \subseteq \mathbb{R}^n$ is open, $F : D \to \mathbb{R}^n$, $y \in \mathbb{R}^n, \boldsymbol{y}_0 \subseteq D$.

Most validated methods are based on the combination of following two algorithms.

Algorithm I: (Existence and enclosure) Find a step size h and a coarse enclosure interval $\boldsymbol{y_i} \subseteq D$ such that for $t \in \boldsymbol{t_i} := [t_i, t_{i+1}]$, the solution $y(t)$ exists and satisfies $y(t) \in \boldsymbol{y_i}$.

Algorithm II: (Tightening) Compute a tight enclosure $\boldsymbol{y}_{i+1}$ for $y(t)$ at $t = t_{i+1}$ such that $\boldsymbol{y}_{i+1} \subseteq \boldsymbol{y}_i$.

2.2 AWA(Anfangswertaufgabe)

Historically, AWA was one of the most important packages to compute guaranteed enclosures for the solution of ordinary IVP. This was written in Pascal-XSC, and was presented by Lohner [53, 51]. This is a one step method and proves existence and uniqueness of the solution by using Picard iteration and gives a rough enclosure of the solution. In order to compute correct enclosures of the solution, a mean value method and the Taylor expansion on a variational equation is applied on global errors. Wrapping effect is reduced by using coordinate transformations and intersecting different enclosures. The efficiency of AWA is affected by Algorithm I which is limited to the Euler step size.

Let

$$y_{i+1} = y_i + h\phi(y_i) + \varepsilon_{i+1} \tag{2.3}$$

be the exact solution, that is $y_{i+1} = y(t_{i+1})$, ε_{i+1} be the exact local discretization error with $y(0) = y_0 + \varepsilon_0$, where ε_0 is an error in the initial value and ϕ be the Picard Lindelöf operator given by

$$\phi(y)(t) = y_0 + \int_0^t F(\tau, y(\tau))d\tau \tag{2.4}$$

applied to validate the existence and uniqueness of the solution with

$$\boldsymbol{y}_{i+1} = \phi(\boldsymbol{y}_i) \subseteq \boldsymbol{y}_i. \tag{2.5}$$

Let $\varepsilon_0, \varepsilon_1, \cdots \varepsilon_{i+1}$ be independent variables. Then (2.3) can be defined as the function

$$y_{i+1} := y_{i+1}(\varepsilon_0, \varepsilon_1, \cdots, \varepsilon_{i+1}) \tag{2.6}$$

of $i+2$ independent variables. Since the exact values of such local discretization variables are not known, therefore, these variables are enclosed by intervals $\boldsymbol{\varepsilon}_k$. The enclosure $\boldsymbol{y}_{i+1}$ of the solution $y(t)$ at $t = t_{i+1}$ is computed by evaluating (2.6) over $\boldsymbol{\varepsilon}_k$. These enclosures of the local errors are computed by using Taylor expansion at $t = t_i$. This results into the solution y at t_i as a Taylor polynomial of degree $p-1$, and the local error ε_{i+1} as the remainder term of order n.

$$\varepsilon_{i+1} = \frac{h^p}{p!} y^{(p)}(\tau_{i+1}), \quad \tau_{i+1} \in (t_i, t_{i+1}). \tag{2.7}$$

The pth derivative of y can be computed by applying automatic differentiation. If the enclosure $\boldsymbol{y}_{i+1}$ of y exists, then the enclosure $\boldsymbol{\varepsilon}_{i+1}$ can be computed. The first enclosure can be roughly computed by applying (2.4) and (2.5). The enclosure of the solution can be obtained by evaluating (2.3), or (2.6) over these intervals. But this enclosure may be wide. To resolve this problem, a mean value form is applied to (2.3).

For that purpose, let $\boldsymbol{\xi}_{i+1} = (\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \cdots, \boldsymbol{\varepsilon}_{i+1})$ and $\delta_k \in \boldsymbol{\varepsilon}_k$ be arbitrary point for $k = 0, 1, \cdots, i+1$, say mid point. Then a mean value form is applied on the interval extension of (2.3), and is given as,

$$\boldsymbol{y}_{i+1} = \tilde{y}_{i+1} + \sum_{k=0}^{i+1} \frac{\partial \boldsymbol{y_{i+1}}}{\partial \boldsymbol{\varepsilon}_k}(\boldsymbol{\xi}_{i+1})(\boldsymbol{\varepsilon}_k - \delta_k), \tag{2.8}$$

where $\tilde{y}_{i+1} = y_{i+1}(\delta_0, \delta_1, \cdots, \delta_{i+1})$ is an approximation of $y(t_{i+1})$ with $\tilde{y}_0 = \delta_0 + y_0$, $\delta_k \in \boldsymbol{\varepsilon}_k$ and

$$\tilde{y}_{i+1} = \tilde{y}_i + h\phi(\tilde{y}_i) + \delta_{i+1} \in \tilde{y}_i + h\phi(\tilde{y}_i) + \boldsymbol{\varepsilon}_{i+1}. \tag{2.9}$$

(2.8) can be written in this form

$$\boldsymbol{y}_{i+1} = \tilde{y}_{i+1} + \boldsymbol{\gamma}_{i+1}, \tag{2.10}$$

where $\tilde{y}_{i+1}$ is an approximate solution and $\boldsymbol{\gamma}_{i+1}$ is the global error. The quality of the bounds can be improved by suitable choice of evaluation of the sum $\boldsymbol{\gamma}_{i+1}$. Let $\boldsymbol{A}_i = I + h\phi'(\boldsymbol{y}_i)$. Following are some possibilities to evaluate the sum $\boldsymbol{\gamma}_{i+1}$.

- **Interval vector:** Combining this sum $\boldsymbol{\gamma}_{i+1}$ into one interval vector

  $$\boldsymbol{\gamma}_0 := \boldsymbol{\varepsilon}_0 - \delta_0, \;\; \gamma_{i+1} \in \boldsymbol{\gamma}_{i+1} := \boldsymbol{A}_i\boldsymbol{\gamma}_i + \boldsymbol{\varepsilon}_{i+1} - \delta_{i+1}, \; i \geq 0 \tag{2.11}$$

  results into sharp bounds. But the bounds grow rapidly in case of oscillating solutions. This problem can be resolved by using coordinate transformation. In that case the sum $\boldsymbol{\gamma}_{i+1}$ is combined with an interval vector $\hat{\boldsymbol{\gamma}}_{i+1}$ and a regular matrix B_{i+1} as

  $$\begin{aligned} \boldsymbol{\gamma}_0 &= B_0\hat{\boldsymbol{\gamma}}_0, \;\; B_0 := I, \;\; \hat{\boldsymbol{\gamma}}_0 := \boldsymbol{\varepsilon}_0 - \delta_0, \\ \boldsymbol{\gamma}_{i+1} &= B_{i+1}\hat{\boldsymbol{\gamma}}_{i+1}, \\ \hat{\boldsymbol{\gamma}}_{i+1} &:= (B_{i+1}^{-1}\boldsymbol{A}_iB_i)\hat{\boldsymbol{\gamma}}_i + B_{i+1}^{-1}(\boldsymbol{\varepsilon}_{i+1} - \delta_{i+1}), \; i \geq 0. \end{aligned} \tag{2.12}$$

- **Parallelepiped:** When $B_{i+1} \in \boldsymbol{A}_i B_i$ for $i \geq 0$ in (2.12), the enclosure of such type forms a parallelepiped in general. But this method fails if the matrix B_{i+1}^{-1} becomes ill conditioned due to large interval of integration. This happens in the case of the linear systems with constant coefficients whose eigenvalues have different real parts.

- $QR-$ **decomposition:** If the matrix B_{i+1} is orthogonal, then the solution is certainly bounded to the moving orthogonal coordinate system. Let $B_{i+1} := Q_{i+1}$, where $\tilde{B}_{i+1} \approx Q_{i+1}R_{i+1}$ is a $QR-$factorization and $\tilde{B}_{i+1} \in \boldsymbol{A}_i B_i$. In this linear transformation, $\boldsymbol{\gamma}_{i+1}$ is enclosed in a rotated rectangle. Prior to $QR-$factorization, columns of $\tilde{B}_{i+1}$ are reordered according to their Euclidean length by applying pivoting method.

Another way of computing $\boldsymbol{\gamma}_{i+1}$ is the combination of all methods mentioned above. Let

$$\begin{aligned}
\hat{\boldsymbol{\gamma}}_0^{(1)} &= \hat{\boldsymbol{\gamma}}_0^{(2)} = \boldsymbol{\varepsilon}_0 - \delta_0, \;\; B_0 := I, \\
\hat{\boldsymbol{\gamma}}_{i+1}^{(1)} &= \boldsymbol{A}_i \hat{\boldsymbol{\gamma}}_i^{(1)} + \boldsymbol{\varepsilon}_{i+1} - \delta_{i+1}, \; i \geq 0, \\
\hat{\boldsymbol{\gamma}}_{i+1}^{(2)} &= (B_{i+1}^{-1}\boldsymbol{A}_i B_i)\hat{\boldsymbol{\gamma}}_i^{(2)} + B_{i+1}^{-1}(\boldsymbol{\varepsilon}_{i+1} - \delta_{i+1}), \; i \geq 0. \qquad (2.13)
\end{aligned}$$

Then

$$\boldsymbol{\gamma}_{i+1} = B_{i+1}(\hat{\boldsymbol{\gamma}}_{i+1}^{(2)} \cap B_{i+1}^{-1}\hat{\boldsymbol{\gamma}}_{i+1}^{(1)}) \cap (\hat{\boldsymbol{\gamma}}_{i+1}^{(1)} \cap B_{i+1}\hat{\boldsymbol{\gamma}}_{i+1}^{(2)}). \qquad (2.14)$$

Since the solution (2.10) is the combination of approximation and global error, therefore, the accuracy of the enclosure can be improved by computing the approximate solution $\tilde{y}_{i+1}$ with high accuracy.

2.3 VNODE (Validated Numerical ODE)

VNODE is an object-oriented, C++ package designed to compute validated solution of IVP for ODEs. It is built on the automatic differentiation (AD) packages TADIFF and FADBAD. TADIFF is used to produce Taylor coefficients with respect to time t for the solutions of an ODE and its associated variational equation that is generated by FADBAD. VNODE and FADBAD/TADIFF are built on top of the interval arithmetic package PROFIL/BIAS [37]. We will discuss the methods employed in VNODE. These methods are based on interval Taylor series (ITS) and interval Hermite-Obreschkoff (IHO) schemes and involve Taylor expansion with respect to time t.

VNODE computes the validated enclosures $\boldsymbol{y}_i, i = 1, 2, \cdots, m$ to the solution of (2.2) at $t_0 < t_1 < \cdots, t_m$ such that

$$y(t_i; t_0, \boldsymbol{y}_0) \subseteq \boldsymbol{y}_i \;\; \text{for} \;\; i = 1, 2, \cdots, m.$$

Such enclosures are computed in two phases namely, Algorithm 1 and Algorithm 2.

Now we discuss automatic generation of Taylor coefficients (TCs) and the methods to implement Algorithm 1 and Algorithm 2. The method for computing TCs deals with autonomous form of 2.2, that is

$$y'(t) \;=\; F(y), \quad y_0 \in \boldsymbol{y}_0. \tag{2.15}$$

Denote the Taylor coefficients by

$$\begin{aligned} F^{[0]}(y) &= y, \\ F^{[j]}(y) &= \frac{1}{j}\Big(\frac{\partial F^{[j-1]}}{\partial y}F\Big)y \;\text{ for }\; j \geq 1. \end{aligned}$$

Then for the IVP (2.15), the jth Taylor coefficient of its solution $y(t_i) = y_i$ becomes

$$\frac{y^{(j)}(t_i)}{j!} = F^{[j]}(y_i).$$

These coefficients are compute by using automatic differentiation (AD) packages TADIFF and FADBAD++. If the input is an interval vector, then interval Taylor coefficients are generated. By using Algorithm 1, a priori bound can be computed by applying Picard Lindelöf operator (2.4) and Banach fixed point theorem but with disadvantage of step size restriction. To enable the larger step size, a higher-order enclosure (HOE) method is implemented by Algorithm I as follows.

Validating existence and uniqueness:

If y_i is in the interior of the $\tilde{\boldsymbol{y}}_i$ and

$$y_i + \sum_{j=1}^{k-1}(t-t_i)^j F^{[j]}(y_i) + (t-t_i)^k F^{[k]}(\tilde{\boldsymbol{y}}_i) \subseteq \tilde{\boldsymbol{y}}_i \tag{2.16}$$

$(k \geq 1)$ for all $t \in [t_i \;\; t_{i+1}]$ and all $y_i \in \boldsymbol{y}_i$ then there exists a unique solution to (2.2) and $y(t_i) = y_i$ for all $y_i \in \boldsymbol{y_i}$ and

$$y(t; t_i, y_i) \in y_i + \sum_{j=1}^{k-1}(t-t_i)^j F^{[j]}(y_i) + (t-t_i)^k F^{[k]}(\tilde{\boldsymbol{y}}_i)$$

for all $t \in [t_i \;\; t_{i+1}]$ and all $y_i \in \boldsymbol{y_i}$.

Computing tighter enclosure:

Tight enclosures can be computed by using ITS and HOE methods.

- **ITS method.** Using $\tilde{\boldsymbol{y}}_i$, tighter enclosure $\tilde{\boldsymbol{y}}_{i+1}$ is computed satisfying (2.2). Consider a Taylor series expansion,

$$\tilde{\boldsymbol{y}}_{i+1} := \tilde{\boldsymbol{y}}_i + \sum_{j=1}^{k-1} h_i^j F^{[j]}(\tilde{\boldsymbol{y}}_i) + h_i^k F^{[k]}(\tilde{\boldsymbol{y}}_i)$$

$(h_i = t_{i+1} - t_i)$, that contain the true solution but with increase in width of $\boldsymbol{y}_{i+1}$, that is

$$w(\boldsymbol{y}_{i+1}) \geq w(\boldsymbol{y}_i), \text{ and usually } w(\boldsymbol{y}_{i+1}) > w(\boldsymbol{y}_i),$$

even if the solutions are contracting. Therefore, mean-value evaluation is applied. Then

$$\begin{aligned} y_i + \sum_{j=1}^{k-1} h_i^j F^{[j]}(y_i) + h_i^k F^{[k]}(\tilde{\boldsymbol{y}}_i) \quad \subseteq \quad & \hat{y}_i + \sum_{j=1}^{k-1} h_i^j F^{[j]}(\hat{y}_i) + h_i^k F^{[k]}(\tilde{\boldsymbol{y}}_i) \\ & + \Big\{ I_n + \sum_{j=1}^{k-1} h_i^j \frac{\partial F^{[j]}}{\partial y}(\boldsymbol{y}_i) \Big\} (\boldsymbol{y}_i - \hat{y}_i), \end{aligned} \tag{2.17}$$

where I is an $n \times n$ identity matrix. The above Jacobians can be evaluated by using FADBAD++ package. (2.17) can be written as

$$\boldsymbol{y}_{i+1} := u_{i+1} + \boldsymbol{\xi}_{i+1} + \boldsymbol{S}_i(\boldsymbol{y}_i - \hat{y}_i), \tag{2.18}$$

where $u_{i+1} = \hat{y}_i + \sum_{j=1}^{k-1} h_i^j F^{[j]}(\hat{y}_i)$ is a point approximation, $\boldsymbol{\xi}_{i+1} = h_i^k F^{[k]}(\tilde{\boldsymbol{y}}_i)$ is an enclosure of the truncation error and $S_i(\boldsymbol{y}_i - \hat{y}_i)$, is an enclosure of the global error propagated to t_{i+1} with

$$\boldsymbol{S}_i = I + \sum_{j=1}^{k-1} h_i^j \frac{\partial F^{[j]}}{\partial y}(\boldsymbol{y}_i).$$

Since the roundoff error in the above computation is enclosed, the enclosure on the true solution is rigorous.

The higher order terms in Taylor series expansion yield overestimation. Therefore, the tight bounds for nonlinear ODEs with initial set $\boldsymbol{y}_i$ cannot be computed unless this set is sufficiently small. But there is no overestimation in the Jacobian evaluation of linear ODEs. The overestimation also arises because of wrapping effect originating from the the product $\boldsymbol{S}_i(\boldsymbol{y}_i - \hat{y}_i)$. That wrapping effect can be reduced using parallelepiped method. But it breaks down due to the inverse of a matrix. Lohner $QR-$ decomposition with $\tilde{B}_{i+1} = \boldsymbol{m}(\boldsymbol{A}_i B_i)$ reduces the wrapping effect.

- **Interval Hermite-Obreschkoff (IHO) method** [73]. This method consists of a predictor phase and a corrector phase. An enclosure $\boldsymbol{y}_i^0$ of the solution at $t = t_i$ is computed in a predictor phase that is later used in a corrector phase to compute a tighter enclosure $\boldsymbol{y}_i \subseteq \boldsymbol{y}_i^0$ at t_i. This method is based on the formula

$$\begin{aligned}\sum_{j=0}^{q}(-1)^j c_j^{q,p} h_i^j F^{[j]}(y_{i+1}) &= \sum_{j=0}^{p} c_j^{p,q} h_i^j F^{[j]}(y_i) \\ &\quad +(-1)^q \frac{q!p!}{(p+q)!} h_i^k F^{[k]}(y; t_i, t_{i+1}),\end{aligned} \tag{2.19}$$

  where $k = p + q + 1$,

$$c_j^{q,p} = \frac{q!(q+p-j)!}{(p+q)!(q-j)!} \quad (q, p \text{ and } j \geq 0),$$

$$y_i = y(t_i; t_0, y_0), \text{ and } y_{i+1} = y(t_{i+1}; t_0, y_0).$$

  For $q > 0$, (2.19), becomes implicit and $\boldsymbol{y}_i^0$ is tightened by applying Newton-like step. The wrapping effect is reduced by $QR-$factorization method.

For the same step size and order, this method is more stable than that of ITS method and produces small local error with fewer Jacobian evaluation.

2.4 VNODE-LP

Validated ODE solvers compute the guaranteed bounds of the solutions by first verifying the existence and uniqueness of the solution and then producing tight enclosures of the true solution. But it needs to verify that the implementation of the method is correct and it produces rigorous bounds. For that purpose, Nedialkov developed a rigorous solver VNODE-LP by reimplementing VNODE (with some algorithmic improvements) with literate programming [38, 39]. The advantage of using LP is that one can split an algorithm into small pieces to make an easy to understand documentation and verifiable implementation at each step. This solver tries to compute the enclosures of the solution of (2.2) at t_m but if the bounds become too wide at, then it returns the bounds at some $t_1 \in [t_0, t_m]$. This is also based on Taylor series and Hermite-Obreschkoff methods. It is a fixed ordered method. The default order is set to 20 but can be varied between 20 and 30 depending upon the systems to be solved. The stepsize varies to in order to keep the estimate of the local excess per unit step below a user specified tolerance. It works well for the system of ODES with point initial conditions or narrow initial boxes. It produces Taylor coefficients and Jacobian of Taylor coefficients by applying AD. LAPACK and BLAS are used for non rigorous linear algebra.

2.5 VSPODE (Validating Solver for Parametric ODEs)

Traditional methods compute tight enclosures of ODEs having uncertain initial conditions. But due to interval dependence problem, parametric uncertainty in ODEs yields wider enclosures. This problem is resolved when these parameters are treated as additional state variables resulting into more computational cost due to wrapping. VSPODE presents a scheme to compute solution enclosures by applying the Taylor model integration. Consider the IVP,

$$y'(t) = F(y, \theta), \quad y(t_0) = y_0 \in \boldsymbol{y}_0, \quad \theta \in \boldsymbol{\theta}, \tag{2.20}$$

where $t \in [t_0, t_m]$, for some $t_m > t_0$, $y_0 \in \mathbb{R}^n$, $\theta \in \mathbb{R}^p$, θ is a parameter. $F : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is $(k-1)-$ times continuously differentiable with respect to the state variables y on $\mathbb{R}^n$, and $(q+1)-$ times continuously differentiable with respect to the parameters on $\mathbb{R}^p$. k is the order of truncation error in the ITS method to be used and q is the order of the Taylor model to be used that represents dependence on the uncertain quantities (parameters and/or initial conditions).

To **validate the existence and uniqueness** of the solution, ITS method with respect to time t is applied to (2.20) with same approach as in case of VNODE. But **computation of tight enclosure** involve Taylor models used to deal with uncertain parameters and initial conditions.

Consider the Taylor models representation of uncertain parameters $y_0 \in \boldsymbol{y}_0$ and $\theta \in \boldsymbol{\theta}_0$ as follows.

$$y_0 \in T_{y_0} = \mathrm{m}(\boldsymbol{y}_0) + (y_0 - \mathrm{m}(\boldsymbol{y}_0)) + [0,\ 0]^n$$

and

$$\theta \in T_\theta = \mathrm{m}(\boldsymbol{\theta}_0) + (\theta - \mathrm{m}(\boldsymbol{\theta}_0)) + [0,\ 0]^p,$$

where $[0,\ 0]^n$ is the n vector with all components $[0,\ 0]$.

Let $y(t; t_0, y_0, \theta)$ be the solution to (2.20) and at $t = t_i$

$$y(t_i; t_0, y_0, \theta) \in p_i(y_0, \theta) + \boldsymbol{\nu}_i, \tag{2.21}$$

where $p_i : \mathbb{R}^{n+p} \to \mathbb{R}^n$ is a polynomial in y_0 and θ of some degree, say l and $\boldsymbol{\nu}_i \in \mathbb{IR}^n$. This means that there is Taylor model at t_i. Now at t_{i+1} for any $y_i \in p_i(y_0, \theta) + \boldsymbol{\nu}_i$ and any $\theta \in T_\theta$,

$$\begin{aligned} y(t_{i+1}; t_0, y_0, \theta) &\in \sum_{j=0}^{k-1} h_i^j F^{[j]}(y_i, \theta) + h_i^k F^{[k]}(\tilde{\boldsymbol{y}}_i, \boldsymbol{\theta}) \\ &\subseteq \sum_{j=0}^{k-1} h_i^j F^{[j]}(p_i + \boldsymbol{\nu}_i, T_\theta) + \boldsymbol{\omega}_{i+1}, \end{aligned} \tag{2.22}$$

where the Taylor coefficients $F^{[j]}$ are functions of both y and θ and can be expressed as

$$\begin{aligned}(y_i)_0 = F^{[0]}(y_i,\theta) &= y_i,\\ (y_i)_1 = F^{[1]}(y_i,\theta) &= F(y_i,\theta),\\ (y_i)_j = F^{[j]}(y_i,\theta) &= \frac{1}{j}\Big(\frac{\partial F^{[j-1]}}{\partial y}F\Big)(y_i,\theta) \ \text{ for } j \geq 2.\end{aligned}$$

$\boldsymbol{\omega}_{i+1} = h_i^k F^{[k]}(\tilde{\boldsymbol{y}}_i, \boldsymbol{\theta})$, in which $\tilde{\boldsymbol{y}}_i$ is a priori enclosure over $[t_i\ t_{i+1}]$. The term $\boldsymbol{\omega}_{i+1}$ is computed with HOE method. The Taylor coefficients $F^{[j]}$ for $j = 1, \cdots, k-1$, can be enclosed by applying Taylor model arithmetic using TADIFF and FADBAD++.

To evaluate $F^{[j]}$ with p_i and T_θ, mean-value theorem is applied to (2.22) as follows:

$$\begin{aligned}y(t_{i+1}; t_0, y_0, \theta) \ \in \ &\sum_{j=0}^{k-1} h_i^j F^{[j]}(p_i, T_\theta) + \boldsymbol{\omega}_{i+1} + \\ &\Big(\sum_{j=0}^{k-1} h_i^j \frac{\partial F^{[j]}}{\partial y}(\boldsymbol{y}_i, \boldsymbol{\theta})\Big)\boldsymbol{\nu}_i. \end{aligned} \tag{2.23}$$

While evaluating $\sum_{j=0}^{k-1} h_i^j F^{[j]}(p_i, T_\theta)$, a polynomial $p_{i+1}(y_0, \theta)$ of degree l can be constructed to enclose the resulting higher order terms. Including this enclosure and $\boldsymbol{\omega}_{i+1}$ in an interval vector $\boldsymbol{u}_{i+1}$, gives

$$\sum_{j=0}^{k-1} h_i^j F^{[j]}(p_i, T_\theta) + \boldsymbol{\omega}_{i+1} \subseteq p_{i+1}(y_0, \theta) + \boldsymbol{u}_{i+1}. \tag{2.24}$$

Let

$$\boldsymbol{V}_i = \sum_{j=0}^{k-1} h_i^j \frac{\partial F^{[j]}}{\partial y}(\boldsymbol{y}_i, \boldsymbol{\theta})$$

then (2.23) and (2.24) imply that

$$y(t_{i+1}; t_0, y_0, \theta) \ \in \ p_{i+1}(y_0, \theta) + \boldsymbol{u}_{i+1} + \boldsymbol{V}_i\boldsymbol{\nu}_i. \tag{2.25}$$

The product $\boldsymbol{V}_i\boldsymbol{\nu}_i$ results into a wrapping effect that can be reduced by replacing (2.21) with the representation

$$\{p_i(y_0, \theta) + B_i s | y_0 \in \boldsymbol{y}_0, \theta \in \boldsymbol{\theta}, s \in \boldsymbol{s}_i\},$$

where $B_i \in \mathbb{R}^{n\times n}$ is a nonsingular and $s_i \in \mathbb{R}^n$, as an enclosure on the solution set at t_i. Then (2.25) becomes

$$y(t_{i+1}; t_0, y_0, \theta) \ \in \ p_{i+1}(y_0, \theta) + \boldsymbol{u}_{i+1} + (\boldsymbol{V}_i B_i)\boldsymbol{s}_i.$$

For the next step, B_{i+1} and $\boldsymbol{s}_{i+1}$ are computed as in Lohner's method.

VSPODE evaluates the Jacobian of $F^{[j]}$ over $\boldsymbol{y}_i$ (in this case $\boldsymbol{\theta}$) like in traditional methods and wrapping is treated in the same way as that in Lohner's method resulting into same global error propagated like in traditional methods.

2.6 VALENCIA-IVP

ValEncIA-IVP, VALidation of state EnClosures using Interval Arithmetic for IVPs computes the validated state enclosures for the system of ODEs with uncertain initial conditions and parameters. The basic idea is to compute non validated approximate solution of IVPs and the corresponding guaranteed error bounds. These bounds are determined by interval arithmetic fixed point iteration. Consider the IVP in the form of time dependent ODEs

$$y_s'(t) = F_s(y_s(t), p(t), t), \tag{2.26}$$

$$\boldsymbol{y}_{s(0)} = \boldsymbol{y}_{s_0} = [\underline{y}_{s_0}, \overline{y}_{s_0}], \quad \boldsymbol{p}(t) = [\underline{p}(t), \overline{p}(t)], \tag{2.27}$$

where $y_s(t) \in \mathbb{R}^{n_s}$ is a state vector, $p(t) \in \mathbb{R}^{n_p}$ is a vector of uncertain parameters, and $F_s : D \to \mathbb{R}^{n_s},\ D \subset \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \times \mathbb{R}^1$ is nonlinear state-space representation. The variation rates

$$\Delta p(t) \in \Delta \boldsymbol{p}(t) : \ = \ [\underline{\Delta p}(t), \overline{\Delta p}(t)] \tag{2.28}$$

of time dependent system parameters $p(t)$ are described by additional ODEs

$$p'(t) = \Delta p(t), \tag{2.29}$$

where $p(t)$ and $\Delta p(t)$ are bounded. Combining (2.26) and (2.29) into a single set of ODEs, the IVP becomes

$$y'(t) = F(y(t), t) \ = \ \begin{pmatrix} F_s(y_s(t), p(t), t) \\ \Delta p(t) \end{pmatrix}, \quad \boldsymbol{y}(0) = \boldsymbol{y}_{s_0} = [\underline{y}_{s_0}, \overline{y}_{s_0}] \tag{2.30}$$

with the extended state vector

$$y(t) = \begin{pmatrix} y_s(t) \\ p(t) \end{pmatrix} \in \mathbb{R}^n, \ \ n = n_s + n_p$$

containing the original system states $y_s(t)$ and time varying system parameters $p(t)$ with $F : D \to \mathbb{R}^n,\ D \subset \mathbb{R}^n \times \mathbb{R}^1$.

Now suppose that

$$\boldsymbol{y}_{encl}(t) = u(t) + \boldsymbol{\varepsilon}(t), \tag{2.31}$$

where $\boldsymbol{y}_{encl}(t)$ is the validated enclosure of all reachable states consisting of an arbitrary non-validated approximate solution $u(t)$ of (2.30) and guaranteed error bounds $\boldsymbol{\varepsilon}(t)$ which are determined by the following two stage procedure.

Stage 1. Computing the time derivatives $\boldsymbol{\varepsilon}'(t)$ of error term by applying iteration formula

$$\begin{aligned} \boldsymbol{\varepsilon}'^{(k+1)}(t) &= \boldsymbol{y}_{encl}'^{(k)}(t) - u'(t) \\ &= F(\boldsymbol{y}_{encl}^{(k)}(t), t) - u'(t) \\ &= F(u(t) + \boldsymbol{\varepsilon}^{(k)}(t), t) - u'(t) \\ &= r(\boldsymbol{\varepsilon}^{(k)}(t), t). \end{aligned} \tag{2.32}$$

According to fixed point theorem, this iteration converges to a verified enclosure of $\boldsymbol{\varepsilon'}(t)$, if $\boldsymbol{\varepsilon'}^{(k+1)}(t) \subseteq \boldsymbol{\varepsilon'}^{(k)}(t)$ holds for $0 \le t \le t_m = T$. The iteration (2.32) is continued until $\boldsymbol{\varepsilon'}^{(k+1)}(t) \approx \boldsymbol{\varepsilon'}^{(k)}(t)$.
Stage 2. Replacing the verified integration of $\boldsymbol{\varepsilon'}^{(k+1)}(t),\ 0 \le t \le T$, with respect to time according to

$$\begin{aligned} \boldsymbol{\varepsilon}^{(k+1)}(T) &\subseteq \boldsymbol{\varepsilon}^{(k+1)}(0) + \int_0^T \boldsymbol{\varepsilon'}^{(k+1)}(\tau)d\tau \\ &= \boldsymbol{\varepsilon}^{(k+1)}(0) + \int_0^T r(\boldsymbol{\varepsilon}^{(k)}(\tau), \tau)d\tau \end{aligned}$$

by the guaranteed bound

$$\boldsymbol{\varepsilon}^{(k+1)}(T) \subseteq \boldsymbol{\varepsilon}^{(k+1)}(0) + T \cdot r([0;T], \boldsymbol{\varepsilon}^{(k)}([0;T])). \tag{2.33}$$

The choice of $\boldsymbol{\varepsilon}(0)$ must satisfy $\boldsymbol{y}_0 \subseteq u(0) + \boldsymbol{\varepsilon}(0)$.

Now we will discuss the key components of VALENCIA-IVP including computation of suitable approximate solution and the techniques for the reduction of the overestimation in (2.32).

A non-validated approximation $y_i,\ i = 1, \cdots, m$ of the original IVP with $y_0 = \mathrm{mid}(\boldsymbol{y}_0),\ t_m = T$ can be computed either analytically or numerically. For higher dimensions, numerical approach is suitable. The iterative formula (2.32) requires the analytic expression for $u(t)$ and $u'(t)$ which is determined by minimization of the distance measure between numerically determined points y_i and the approximate solution $u(t)$ of the IVP.

$$D = \sum_{i=1}^{m} \|(y_i - u(t_i))\|_2^2 .$$

Linear interpolations

$$u(t) = y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i}.(t - t_i) \tag{2.34}$$

with the time derivative

$$u'(t) = \frac{y_{i+1} - y_i}{t_{i+1} - t_i} \tag{2.35}$$

for $t_i \le t \le t_{i+1},\ i = 1, \cdots, m-1$ lead to good results. y_i is computed by an explicit Euler method with constant step size.

The iteration (2.32) is initialized by choosing the interval enclosures $\boldsymbol{\varepsilon}(t)$ and $\boldsymbol{\varepsilon'}(t)$ satisfying $\boldsymbol{y}_0 \subseteq u(0) + \boldsymbol{\varepsilon}(0)$. The evaluation of (2.32) is continued until $\boldsymbol{\varepsilon'}^{(1)}(t) \subseteq \boldsymbol{\varepsilon'}^{(0)}(t)$. Otherwise, initial guesses for $\boldsymbol{\varepsilon}(t)$ and $\boldsymbol{\varepsilon'}(t)$ are modified.

Convergence of the iteration is improved by splitting the time interval $[0;T]$ into small subintervals and then (2.33) becomes

$$\boldsymbol{\varepsilon}^{(k+1)}(t_{i+1}) = \boldsymbol{\varepsilon}^{(k+1)}(0) + \sum_{j=0}^{i} (t_{i+1} - t_j).r([t_j\ \ t_{i+1}], \boldsymbol{\varepsilon}^{(k)}([t_j\ \ t_{i+1}])) \tag{2.36}$$

for all $t_i,\ \ i = 1, \cdots, m-1$.

Overestimation produced by *natural interval evaluation* of (2.32) can be reduced by applying a mean-value theorem to (2.32)

$$r(a) \ \in \ r(a_x) + \frac{\partial r}{\partial a}\Big|_{a=\boldsymbol{a}}.(\boldsymbol{a} - a_x) \ \text{ for all } \ a \in \boldsymbol{a}, \tag{2.37}$$

where

$$\boldsymbol{a} \ = \ \begin{pmatrix} \boldsymbol{\varepsilon}(t_i) \\ [t_i \ \ t_{i+1}] \end{pmatrix} \ \text{ and } \ a_x = \text{mid}(\boldsymbol{a}). \tag{2.38}$$

The intersection of *natural interval evaluation* with *mean-value rule evaluation* results into the tightest possible bounds.

Further reduction in overestimation can be made by applying a **monotonicity test** . For example, $\inf\left(\frac{\partial r_i}{\partial a_j}\right) > 0,\ j = 1, \cdots, n+1$, $\boldsymbol{a}$ can be replaced by $\underline{a}_j$ to compute the infimum of the range of r_i over $\boldsymbol{a}$ and by $\overline{a}_j$ to compute its supremum.

VALENCIA-IVP implements **consistency test** to reduce the overestimation. The purpose of this test is to detect and eliminate the inconsistent subintervals of the state enclosure $\boldsymbol{y}_{\text{encl}}(t)$ originated from overestimation.

2.7 Kühn's rigorous error bounds

W. Kühn [43] developed a method to compute the rigorous error bounds of ODEs. The method was based on the defect estimates. We discuss this method in some detail.

Consider a C^2 function $F : D \to \mathbb{R}^n$, $D \subseteq \mathbb{R}^n$ is open. The IVP on a given interval $[0, T]$ is given by

$$y'(t) \ = \ F(y(t)), \ \ y(0) \in \eta + \Sigma \tag{2.39}$$

with $\eta + \Sigma$ as an initial compact set. The variational and the adjoint equations are

$$Y'(t) \ = \ DF_{y(t)}Y(t), \ \ Y(0) = I, \tag{2.40}$$

$$Z'(t) \ = \ -Z(t)DF_{y(t)}, \ \ Z(0) = I, \tag{2.41}$$

respectively.

The rigorous error estimates can be found by computing the defect estimates of the approximate solution of (2.39), (2.40) and (2.41) and estimating the second derivative of F.

For this, let u, U and W be the approximate solutions of (2.39), (2.40) and (2.41) respectively with $u(0) = \eta$, $U(0) = I = W(0)$. The defects of u, U and W are defined by

$$\begin{aligned} \delta u(t) &= u'(t) - F(u(t)), \\ \delta U(t) &= U'(t) - DF_{u(t)}U(t), \\ \delta W(t) &= W'(t) + W(t)DF_{u(t)}. \end{aligned}$$

The suitable norms required to estimate the second derivative of F, and for functions $\phi \in C([0,T], \mathbb{R}^n)$ are chosen as:
For fixed vector $x \in \mathbb{R}^n$, the second derivative D^2F_x of F at x is a bilinear map, and the induced operator norm is

$$\|D^2F_x\| = \max_i \sum_{j,k} \left|\frac{\partial^2 F_i(x)}{\partial x_j \partial x_k}\right|.$$

For functions $\phi \in C([0,T], \mathbb{R}^n)$, the L_p norm

$$\|\phi\|_1 = \int_0^h \|\phi(t)\| dt, \tag{2.42}$$

and the maximum norm

$$\|\phi\| = \max_{t\in[0,T]} \|\phi(t)\|. \tag{2.43}$$

are used. (2.42) and (2.43) imply that

$$\|\phi\|_1 \leq h\|\phi\|.$$

Now since the solution W of the adjoint equation is the inverse of the solution U of the variational equation, therefore W can be used to approximate U^{-1}. Following theorem by Kühn justifies this argument.

2.7.1 Theorem. *Let s be a positive number such that*

$$\|\delta W U\|_1 + \|W\delta U\|_1 \leq \frac{s}{(s+1)^2}.$$

Then $U(t)$ is invertible on $[0,T]$ and

$$\|U^{-1}(t) - W(t)\| \leq s\|W(t)\|. \tag{2.44}$$

Using (2.44), we see that

$$\|U^{-1}(t) - W(t)\| \leq \|U^{-1}(t)\| - \|W(t)\| \leq s\|W(t)\|$$

or

$$\|U^{-1}\| \leq (s+1)\|W(t)\|.$$

Now consider the linearization of $y(t, y_0)$ about $y_0 = \eta$

$$y(t, y_0) \;=\; y(t, \eta) + Y(t, \eta)(y_0 - \eta) + O(y_0 - \eta)^2. \tag{2.45}$$

Replacing $y(t, \eta)$ and $Y(t, \eta)$ by the corresponding approximate solution, (2.45)

$$y(t, y_0) \;\approx\; u(t) + U(t)(y_0 - \eta).$$

Following is the Theorem 1 by Kühn which gives the rigorous error bounds of ODEs. Here $\|\delta u\|_1$, $\|\delta U\|_1$ are the defects of u and U.

2.7.2 Theorem. *Let $\alpha > 0$, γ be the radius of Σ, and let β be such that $\|D^2 F_x\| \leq \beta$ for all x in the neighborhood χ of the graph of u defined by*

$$\chi \;=\; \{u(t) + v : t \in [0, T] \text{ and } \|v\| \leq \|U(t)\|(\alpha + \gamma)\}. \tag{2.46}$$

If $U(t)$ is invertible on $[0, T]$, and

$$\|U^{-1}\|(\|\delta u\|_1 + \|\delta U\|_1(\alpha + \gamma) + \frac{1}{2}\beta\|U\|_2^2(\alpha + \gamma)^2) \;\leq\; \alpha, \tag{2.47}$$

then (2.39) has a solution on $[0, T]$ for all $y_0 \in \eta + \Sigma$, and

$$\|y(t, y_0) - u(t) - U(t)(y_0 - \eta)\| \;\leq\; \alpha\|U(t)\|. \tag{2.48}$$

Kühn estimated the defects involved in (2.47) by computing higher order derivatives which is a difficult task.

Chapter 3

Chernousko's error bounds for linear ODEs

Different schemes have been developed to compute the guaranteed state estimation for uncertain dynamic systems but ellipsoidal approximation is the most efficient technique. The advantages of this method include explicit representation of resultant estimates, smooth boundaries of domain, possibility of optimization, developing algorithms, etc. In this method, reachable sets are approximated by means of ellipsoids. Different properties of reachable sets were explored and applied by Krasovsky [40, 41], Kurzhanski [44], Lee & Markus [47]. Ellipsoidal approximations of these sets in state space were discussed by Bertsekas [7], Kurzhanski [44], Schweppe [97, 98], Schlaepfer & Schweppe [95]. Chernousko also presented a method to approximate the reachable sets of uncertain dynamic systems by means of ellipsoids. He applied two sided (inner and outer) ellipsoidal approximations that are optimal with respect to their volume. In this chapter, we will present a review some part of Chernousko [16] including sections 5,6 and 8.

3.1 Reachable sets

Reachable sets play an important role in control theory. A number of basic problems of this theory can be solved in terms of reachable sets such as guaranteed estimation (filtering) in dynamic systems, time-optimal control etc. Chernousko defines reachable sets, their evolutionary property, subreachable and superreachable sets as follows: Consider a dynamic system

$$y'(\tau) \quad = \quad F(\tau, y(\tau)). \tag{3.1}$$

Let M be a given closed set in $\mathbb{R}^n$ containing all possible initial states of the system. In particular, the set M can be a point: $M = \{y_0\}$ where y_0 is the given initial state. Then

$$y(0) \quad = \quad y_0. \tag{3.2}$$

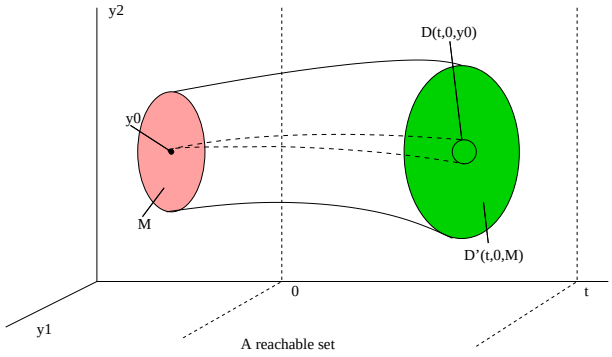

Figure 3.1:

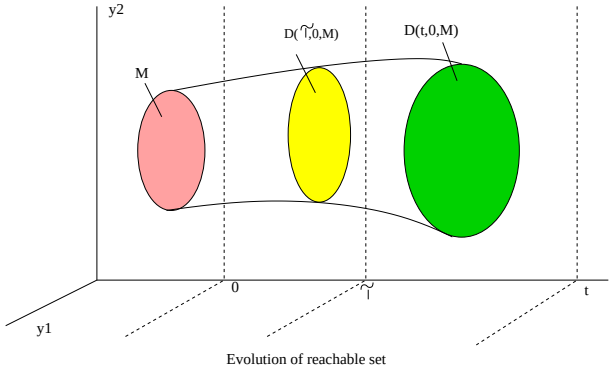

Figure 3.2:

3.1.1 Definition. The set of the end points $y(t)$ at $t \geq 0$ of all trajectories $y(.)$ of (3.1) under the initial conditions $y(0) \in M$ is called the reachable set of the system and is denoted by

$$D(t, 0, M) = D_t := \{y(t) \mid y(0) \in M, y'(\tau) = F(\tau, y(\tau)) \text{ for } \tau \in [0, t]\} \quad (3.3)$$

The initial set M and the reachable set D_t are shown in Figure 3.1 for n=2. The set D_t is the union of all sets $D(t, 0, y_0)$, where $y_0 \in M$.

Evolutionary Property

Reachable sets defined by the definition 3.1.1 have the following property:

$$D_t = D[t, \tau, D(\tau, 0, M)], \quad (3.4)$$

where τ is any time instant, $\tau \in [0, t]$. This property implies that the reachable set for instant t can be obtained from the same set for the instant τ by prolongation of all trajectories, starting from the instant τ up to the instant t.

3.1.2 Definition. The sets $D_-(t)$ are called subreachable sets for (3.1) with (3.2) if the following inclusion holds for all $\tau \in [0,t]$:

$$D_-(t) \subset D[t,\tau,D_-(\tau)] \tag{3.5}$$

and besides if

$$D_-(0) \subset M. \tag{3.6}$$

3.1.3 Definition. The sets $D_+(t)$ are called superreachable sets for (3.1) with (3.2) if the following inclusion holds for all $\tau \in [0,t]$:

$$D_+(t) \supset D[t,\tau,D_+(\tau)] \tag{3.7}$$

and besides if

$$D_+(0) \supset M. \tag{3.8}$$

It follows from (3.4)-(3.8) that for all time instants

$$D_-(t) \subset D[t,0,D_-(0)] \subset D_t \subset D[t,0,D_+(0)] \subset D_+(t).$$

Therefore, the subreachable and superreachable sets provide two-sided approximations of the reachable set D_t for all $t \geq 0$

$$D_-(t) \subset D_t \subset D_+(t). \tag{3.9}$$

3.2 Ellipsoidal approximation of reachable sets

Several approaches can be used to obtain reachable sets that produce accurate approximation such as polyhedral approximation by means of support function yields accurate approximation but this is computationally expensive. The approximation made by canonical shapes like rectangular parallelepiped and ellipsoid needs limited number of parameters (only $n + n(n + 1/2)$) and produce finite approximate error that cannot be reduced. But this approach is applicable. Ellipsoidal estimation of reachable sets is an efficient technique where as rectangular parallelepiped are not invariant with respect to linear transformation.

Chernousko defines an ellipsoid by the following inequality

$$E(a,Q) := \{y : (y-a)^T Q^{-1}(y-a) \leq 1\}, \tag{3.10}$$

where $a \in \mathbb{R}^n$ is the center of ellipsoid, $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive definite and invertible matrix.

The matrix Q can be diagonalized by rotating the coordinate axes so that these axes become parallel to the principal axes of the ellipsoid. That is

$$Q = \mathrm{diag}(Q_{11}, \cdots, Q_{nn}).$$

Then (3.10) is reduced to

$$\sum_{i=1}^{n} Q_{ii}^{-1}(x_i - a_i)^2 = \sum_{i=1}^{n}(x_i - a_i)^2/c_i^2 \leq 1,$$

where c_i are the semiaxes of the ellipsoid. That is

$$Q_{ii} = c_i^2, \quad i = 1, \cdots, n. \tag{3.11}$$

If one of the two matrices Q or Q^{-1} is not positive definite, say Q, then some of the principal axes of the ellipsoid are zero. Then the ellipsoid degenerates because the volume becomes zero in that case. For example, a disk in three dimensional space. In case when Q^{-1} is not positive definite, then the some of the principal axes become infinite and the volume of the ellipsoid is also infinite. A cylinder is an example of such type of degenerated ellipsoid.

If $Q \to 0$, then the corresponding ellipsoid degenerates into a point, that is, $E(a, Q) = x = a$ [16, p. 57-58].

3.2.1 Affine transformation

Let

$$z \;=\; Ay + b \tag{3.12}$$

be an affine transformation in $\mathbb{R}^n$, where $A \in \mathbb{R}^{n\times n}$ is a nonsingular and $b \in \mathbb{R}^n$. It can be observed that (3.12) applied to any ellipsoid results into an ellipsoid. To show that, choose $y \in E(a, Q)$ to be an arbitrary vector and solve (3.12) with respect to y. Put $y = A^{-1}(z - b)$ into (3.10) then

$$\begin{aligned}(A^{-1}[z-b]-a)^T Q^{-1}(A^{-1}[z-b]-a)& \\ = (z - Aa - b)^T((A^T)^{-1}Q^{-1}A^{-1})(z - Aa - b)& \\ = (z - [Aa + b])^T(AQA^T)^{-1}(z - [Aa + b]) \leq 1,&\end{aligned}$$

which implies that z belongs to an ellipsoid with center at $Aa + b$ and the matrix AQA^T. That is

$$z \;=\; Ay + b \in E(Aa + b, AQA^T), \quad y \in E(a, Q). \tag{3.13}$$

Rewrite (3.13),

$$AE(a, Q) + b \;=\; E(Aa + b, AQA^T). \tag{3.14}$$

3.2.2 Approximating the convex sets

For ellipsoidal approximation of convex sets, inner ellipsoids E_- of the maximal volume and the outer ellipsoids E_+ of the minimal volume will be used. Following results are taken from [16, p. 62]

3.2.1 Theorem. *For any bounded set D in $\mathbb{R}^n$, there exists a unique ellipsoid E_+ of the minimal volume containing D, $E_+ \supset D$*

3.2.2 Theorem. *For any close, convex set D in $\mathbb{R}^n$, there exists a unique ellipsoid E_- of the maximal volume contained in D, $E_- \subset D$ [16, p. 62]*

Chernousko forgot to mention that D must also be bounded. The outer ellipsoid of the minimal volume E_+ and the inner ellipsoid of the maximal volume E_- can be found by computing the vector a and the symmetric positive definite matrices Q satisfying the conditions

$$D \subset E(a, Q) = E_+, \qquad \det Q \to \min, \tag{3.15}$$

$$D \supset E(a, Q) = E_-, \qquad \det Q \to \max. \tag{3.16}$$

According to Theorem 3.2.1, (3.15) has unique solution for any bounded set D, and Theorem 3.2.2 implies that (3.16) has the unique solution for any closed, convex set, bounded D.

3.3 Sum of ellipsoids

Let

$$\begin{aligned} E(a_1, Q_1): &\quad (y - a_1)^T Q_1^{-1} (y - a_1) \le 1, \\ E(a_2, Q_2): &\quad (y - a_2)^T Q_2^{-1} (y - a_2) \le 1 \end{aligned} \tag{3.17}$$

be two ellipsoids in n-dimensional Euclidean space $\mathbb{R}^n$, where $a \in \mathbb{R}^n$ be the centers of ellipsoids, $Q \in \mathbb{R}^{n \times n}$ be symmetric positive definite matrices, $i = 1, 2$.

Then the **Minkowski sum** of above two ellipsoids (3.17) is the set of all points y such that $y = y_1 + y_2$, where y_1 and y_2 belong to the ellipsoids of the equation (3.17), that is,

$$\begin{aligned} y = y_1 + y_2 \in S &= E(a_1, Q_1) + E(a_2, Q_2), \\ y_1 \in E(a_1, Q_1), &\quad y_2 \in E(a_2, Q_2). \end{aligned} \tag{3.18}$$

The set S is bounded, closed, and convex, but, in general it is not an ellipsoid. For example, for $n = 2$, if both ellipsoids degenerate into segments such that one segment of each ellipsoid has zero length and the center of the segments

coincide with the origin of coordinates $y = 0$, then the sum (3.18) of the segments results into a parallelogram with sides parallel to the segments.

Following statement comes from [16, p. 70]

3.3.1 Problem. Find the ellipsoid $E(a_-, Q_-)$ of the maximal volume contained in sum of two ellipsoids (3.18), that is, an ellipsoid such that

$$E(a_-, Q_-) \quad \subset \quad S, \quad \det Q_- \to \max . \tag{3.19}$$

Theorem 3.2.2 implies that (3.19) has a unique solution. This is true in only when one of the ellipsoids say $E(a_1, Q_1)$ degenerates. That is the matrix Q_1, is non-negative definite, whereas the other matrix Q_2 of the ellipsoid $E(a_2, Q_2)$ is positive definite.

3.3.1 Simultaneous diagonalization of quadratic forms

Let the vectors z_i be given by the equalities

$$z_i \quad = \quad A(y_i - a_i), \quad i = 1, 2. \tag{3.20}$$

[16, p. 70] choses the matrix A to be nonsingular so that both matrices of the ellipsoids in (3.18) can be diagonalized. This is possible because one of the matrices, namely Q_2 is positive definite. (3.20) implies that

$$y_i - a_i \quad = \quad A^{-1} z_i, \quad i = 1, 2. \tag{3.21}$$

Using (3.21) in (3.17), it can be seen that

$$z_i^T (A Q_i A^T)^{-1} z_i \quad \leq \quad 1, \quad i = 1, 2. \tag{3.22}$$

Both equations (3.22) are reduced to the canonical form if

$$A Q_i A^T \quad = \quad D_i, \quad i = 1, 2, \tag{3.23}$$

where D_1 and D_2 are diagonal matrices with elements

$$\begin{array}{rcl} D_i & = & \mathrm{diag}(c_1^i, \cdots, c_n^i), \quad i = 1, 2, \\ c_j^1 & \geq & 0, \quad c_j^2 \geq 0, \quad j = 1, \cdots, n. \end{array} \tag{3.24}$$

To find the matrix A, we consider the eigen value problem

$$Q_1 y = \lambda Q_2 y, \tag{3.25}$$

where λ is an eigen value. Using the result from [103, p. 277],

$$\det(Q_1 - \lambda Q_2) = 0. \tag{3.26}$$

The characteristic equation (3.26) involves only λ not eigen vectors say x. Suppose λ_j are the roots (3.26). Then for each λ_j, solve

$$(Q_1 - \lambda Q_2)x^j = 0 \text{ or } Q_1 x^j = \lambda_j Q_2 x^j \tag{3.27}$$

Some of the roots λ may be multiple but for each multiplicity l there exists exactly l linearly independent eigenvectors z^j [29]. If the eigenvalues λ_j are taken according to their multiplicity then there will be always n eigenvalues λ_j and n linearly independent eigenvectors x^j. The matrix A can be represented by

$$A^T \;=\; \{x^1, \cdots, x^n\}. \tag{3.28}$$

Here x^j are the eigen vectors. The diagonal matrices are equal to

$$D_1 = \text{diag}(\lambda_1, \cdots, \lambda_n), \quad D_2 = I. \tag{3.29}$$

Note: Coordinate transformation represented by A multiplies all volumes by the same constant, hence solving the minimization problem in transformed coordinates solves the original problem.

Assume that the matrix A of (3.20) satisfies (3.23). Then (3.18) and (3.21) imply that

$$y \;=\; y_1 + y_2 = a_1 + a_2 + A^{-1}(y_1 + y_2). \tag{3.30}$$

Since the vectors y_1 and y_2 belong to the ellipsoids in (3.18), the vectors z_1 and z_2 belong to the respective ellipsoids (3.22). Therefore the vectors z_1 and z_2 belong to the set S_z, which is the Minkowski sum of ellipsoids (3.22). Using (3.30), (3.22) and (3.23)

$$z = z_1 + z_2 \;\in\; S_z = E(0, D_1) + E(0, D_2), \tag{3.31}$$

$$y \;=\; a_1 + a_2 + A^{-1}z. \tag{3.32}$$

To solve the Problem 3.3.1, the sum S from (3.18) will be replaced with the sum S_z from (3.31). Since the ellipsoids $E(0, D_i),\ i = 1, 2$ in (3.31) have common center at origin of coordinates, and diagonal matrices, so the principal axes of these ellipsoids coincide with the coordinate axes. Therefore, the sum S_z is a closed convex set such that all coordinate hyperplanes are hyperplanes of symmetry.

3.3.2 Problem. Find the ellipsoid $E(0, D_-)$ of the maximal volume such that

$$E(0, D_-) \subset S_z \tag{3.33}$$

holds [16, p. 72].

D_- is a diagonal matrix with positive entries given as

$$D_- \quad = \quad \text{diag}(c_{1-}, \cdots, c_{n-}), \quad c_{j-} > 0, \quad j = 1, \cdots, n. \tag{3.34}$$

Now the positive elements of the diagonal matrix D_- will be computed, such that

$$S_z \quad \supset \quad E(0, D_-), \quad \prod_{j=1}^{n} c_{j-} \to \max \tag{3.35}$$

holds. Theorem 3.2.2 implies that (3.35) has unique solution.

Since the semiaxes of the ellipsoids $E(0, D_i)$ are equal to $(c_j^i)^{1/2}$, $i = 1, 2$; $j = 1, \cdots, n$, so the convex set S_z defined by (3.31) as the sum of these ellipsoid is the contained in parallelepiped P having semiaxes equal to the sum of the respective semiaxes of these ellipsoids:

$$S_z \quad \subset \quad P, \quad P : |z_j| \leq (c_j^1)^{1/2} + (c_j^2)^{1/2}, \quad j = 1, \cdots, n. \tag{3.36}$$

Let $E(0, D)$ be an ellipsoid with same axes as that of parallelepiped and touching it from within at points lying on the coordinate axes. It will be proved that this ellipsoid is contained in S_z, that is,

$$\begin{aligned} S_z \quad &\supset \quad E(0, D), \quad D = \text{diag}(c_1, \cdots, c_n), \\ c_j \quad &= \quad \left((c_j^1)^{1/2} + (c_j^2)^{1/2}\right)^2, \quad j = 1, \cdots, n. \end{aligned} \tag{3.37}$$

Let $z \in E(0, D)$ and $z = z^1 + z^2$, where z^i be the vectors with following components:

$$z_j^i \quad = \quad (\frac{c_j^i}{c_j})^{1/2} z_j, \quad i = 1, 2; \quad j = 1, \cdots, n, \tag{3.38}$$

with z_i as the components of the vector z.

Then $z \in E(0, D)$ implies that

$$\sum_{j=1}^{n} (c_j)^{-1} (z_j)^2 \leq 1.$$

This inequality when used in (3.38) results into

$$\sum_{j=1}^{n} (c_j^i)^{-1} (z_j^i)^2 \leq 1, \quad i = 1, 2.$$

This shows that any vector $z \in E(0, D)$ can be represented as the sum $z = z^1 + z^2$, where $z^i \in E(0, D_i)$, $i = 1, 2$ which proves the inclusion (3.37).

Next step is to show that the ellipsoid $E(0, D_i)$ has the maximal volume among all ellipsoids inscribed into S_z. Since (3.36) holds therefore, one only

needs to prove that $E(0, D_i)$ has the maximal volume amongst all ellipsoids inscribed into parallelepiped P. When this parallelepiped is stretched along the coordinate axes, it is transformed into the n-dimensional cube $P_n^* : |y_j| \leq 1,\ \ j = 1, \cdots, n$. This transformation reduces ellipsoid $E(0, D_i)$ from (3.38) into unit ball $|y| \leq 1$. Therefore, it only needs to show that the cube P_n^* has maximal volume among all parallelepiped circumscribed about the unit ball $|y| \leq 1$.

This statement will be proved by applying mathematical induction with respect to the space dimension n. For $n = 1$, it is trivial. n=2, implies that the area of the square circumscribed about a circle is smaller than the area of any parallelogram circumscribed about the same circle. Suppose that the statement is true for some n. Take an arbitrary $(n + 1)$-dimensional parallelepiped P_{n+1} with $(n + 1)$-dimensional unit ball. The volume of P_{n+1} is equal to its height (not smaller than the diameter of the ball) multiplied by the volume of n-dimensional parallelepiped P_n that forms the base of P_{n+1}. P_n contains the n-dimensional unit ball. Since the statement is assumed to be true for n, the volume of P_n is not smaller than the volume of the cube P_n^*. As a result, the volume of P_{n+1} is not smaller than the volume of the cube P_n^* multiplied by the diameter of the unit ball. This product is equal to the volume of the cube P_{n+1}^*. This proves the statement.

Therefore, the ellipsoid $E(0, D)$ in (3.37) has maximal volume, that is, $D_- = D$. Then the solution (3.37) of (3.35) can be written as

$$D_- \ = \ (D_1^{1/2} + D_2^{1/2})^2. \tag{3.39}$$

The square roots of diagonal matrices with nonnegative elements are also diagonal matrices whose elements are equal to the square roots of the corresponding elements of the original matrices.

Now take the general case by considering the original problem (3.19). The vectors y and z are bound by the affine transformation (3.32) that converts the sum S_z of the ellipsoids (3.31) into the sum of the ellipsoids (3.18). Since the affine transformations do not change the ratio of the volume of any two domains, the ellipsoid $E(0, D)$ is transformed into the ellipsoid $E(a_-, Q_-)$ of the maximal volume contained in S. By using (3.14) and (3.32) the parameters of $E(a_-, Q_-)$ are obtained as

$$a_- \ = \ a_1 + a_2, \quad Q_- = A^{-1} D_- (A^T)^{-1}. \tag{3.40}$$

Rewriting (3.40) for Q_- by using (3.39) and (3.23), one can have

$$Q_- \ = \ A^{-1} \Big((AQ_1A^T)^{1/2} + (AQ_2A^T)^{1/2} \Big)^2 (A^T)^{-1}. \tag{3.41}$$

Here A is a nonsingular matrix of the affine transformation satisfying (3.23). This matrix A may not be unique. At the same time, the solution of our problem is unique and is independent of the choice of A.

Now (3.41) is simplified for Q_- by assuming A to be an arbitrary nonsingular matrix. Since Q_1 and Q_2 are symmetric and Q_1 is nonnegative definite, whereas Q_2 is positive definite matrix, so the matrices AQ_1A^T and AQ_2A^T as well as their square roots, possess the respective properties.

The following identity holds for any function of matrices $f(Z)$

$$f(CZC^{-1}) \;=\; Cf(Z)C^{-1}, \tag{3.42}$$

C be an arbitrary nonsingular matrix.

Take $A = UB$ in (3.41) where U is an orthogonal matrix, $U^T = U^{-1}$, and transform (3.41) using (3.42)

$$\begin{aligned} Q_- &= B^{-1}U^{-1}\Big((U(BQ_1B^T)U^{-1})^{1/2} + (U(BQ_2B^T)U^{-1})^{1/2}\Big)^2 U(B^T)^{-1} \\ &= B^{-1}U^{-1}\Big(U((BQ_1B^T)^{1/2} + (BQ_2B^T)^{1/2})U^{-1}\Big)^2 U(B^T)^{-1} \\ &= B^{-1}\Big((BQ_1B^T)^{1/2} + (BQ_2B^T)^{1/2})\Big)^2 (B^T)^{-1}. \end{aligned} \tag{3.43}$$

Comparing the latter expression for Q_- with (3.41), it is observed that (3.41) is invariant if A is replaced by UA, where U is an orthogonal matrix.

Now one can find the matrix A satisfying (3.23) as $A = UQ_2^{-1/2}$ exists. Taking into account the invariance proved above, $A = UQ_2^{-1/2}$ is replaced in (3.41) by $A = Q_2^{-1/2}$, that is

$$\begin{aligned} Q_- &= Q_2^{1/2}\Big(I + (Q_2^{-1/2}Q_1Q_2^{-1/2})^{1/2}\Big)^2 Q_2^{1/2} \\ &= Q_2^{1/2}\Big(I + Q_2^{-1/2}Q_1Q_2^{-1/2} + 2(Q_2^{-1/2}Q_1Q_2^{-1/2})^{1/2}\Big)Q_2^{1/2} \\ &= Q_1 + Q_2 + 2Q_2^{1/2}(Q_2^{-1/2}Q_1Q_2^{-1/2})^{1/2}Q_2^{1/2}. \end{aligned} \tag{3.44}$$

When both the matrices Q_1 and Q_2 are positive definite, then they can change places in the last expression. Rewriting (3.40)

$$\begin{aligned} a_- &= a_1 + a_2, \\ Q_- &= Q_1 + Q_2 + 2Q_2^{1/2}(Q_2^{-1/2}Q_1Q_2^{-1/2})^{1/2}Q_2^{1/2} \\ &= Q_1 + Q_2 + 2Q_1^{1/2}(Q_1^{-1/2}Q_2Q_1^{-1/2})^{1/2}Q_1^{1/2}. \end{aligned} \tag{3.45}$$

This results into the following theorem.

3.3.3 Theorem. *The parameters of the ellipsoid $E(a_-, Q_-)$ of the maximal volume contained in the sum S of (3.18), one of which may be degenerate (the matrix Q_1 is nonnegative definite, and the matrix Q_2 is positive definite), are given by (3.45) and (3.41). The matrix A in (3.41) satisfies (3.23). Both equations (3.41) for Q_- are equivalent if both matrices Q_1 and Q_2 are positive definite. If the matrix Q_1 is not positive definite, the second equation (3.45) for Q_- should be used.*

Now we discuss a particular case for the addition of the ellipsoids, when *one of the ellipsoids in (3.18) is much smaller than the other.*

In this case suppose that all semiaxes of the ellipsoid $E(a_1, Q_1)$ are proportional to the small number ϵ, whereas the other ellipsoid $E(a_2, Q_2)$ has finite semiaxes. Let

$$Q_1 \;=\; \epsilon^2 Q_{01}, \quad 0 < \epsilon \ll 1, \tag{3.46}$$

Q_{01} be a symmetric nonnegative definite matrix, ϵ be a small parameter. As $\epsilon \to 0$, the ellipsoid $E(a_1, Q_1)$ degenerates into a point. The matrix Q_2 does not depend on ϵ.

By inserting (3.46) into (3.41) and (3.45) for the inner ellipsoid, accuracy upto $O(\epsilon)$ is obtained

$$\begin{aligned} a_- &= a_1 + a_2, \\ Q_- &= Q_2 + 2\epsilon A^{-1}(AQ_{01}A^T)^{1/2}(AQ_2A^T)^{1/2}(A^T)^{-1} \\ &= Q_2 + 2\epsilon Q_{01}^{1/2}\Big(Q_{01}^{-1/2}Q_2Q_{01}^{-1/2}\Big)^{1/2}Q_{01}^{1/2} \\ &= Q_2 + 2\epsilon Q_2^{1/2}(Q_2^{-1/2}Q_{01}Q_2^{-1/2})^{1/2}Q_2^{1/2}. \end{aligned} \tag{3.47}$$

Here the transformation matrix A reduces both matrices Q_{01} and Q_2 to the diagonal form simultaneously.

3.4 Ellipsoidal approximations for dynamic systems

In this section, the ellipsoidal approximation of the reachable sets for differential equations is discussed. Let

$$\dot{y} \;=\; C(t)x + K(t)u + F(t), \tag{3.48}$$

$$u \;\in\; E[0, G(t)] \tag{3.49}$$

be a vector differential equation describing a linear system with state vector $y \in \mathbb{R}^n$, control vector $u \in \mathbb{R}^m$. The n-vector $F(t)$, $n \times n$ matrix C, $n \times m$ matrix K and the $m \times m$ matrix G are the piecewise continuous functions of time. The initial conditions are

$$y(0) \;\in\; E(a_0, Q_0), \tag{3.50}$$

where $a_0 \in \mathbb{R}^n$ and $Q_0 \in \mathbb{R}^{n\times n}$ is a given symmetric positive define matrix. The reachable set of (3.48) with the initial condition (3.50) is denoted by

$$D_t \;=\; D[t, 0, E(a_0, Q_0)]. \tag{3.51}$$

Now the two sided estimates on the reachable sets (3.51) by means of sub-reachable and superreachable ellipsoids are obtained as:

$$E[a_-(t), Q_-(t)] \subset D[t, 0, E(a_0, Q_0)] \subset E[a_+(t), Q_+(t)], \tag{3.52}$$

where $a_-(t)$ and $a_+(t)$ are n-vectors, $Q_-(t)$ and $Q_+(t)$ are symmetric positive definite $n \times n$ matrices. These vectors and matrices are to be found as functions of time for $t \geq 0$.

Rewrite (3.48) and (3.49) as

$$\dot{y} \;=\; C(t)y + \nu,\;\; \nu \in E[F(t), B(t)],\;\; B(t) = KGK^T. \tag{3.53}$$

In order to derive the equations of the approximating ellipsoids, the differential equation (3.53) is replaced by its finite-difference approximation,

$$y(t+h) \;=\; y(t) + hC(t)y(t) + h\nu(t),\;\; h > 0. \tag{3.54}$$

Rewriting (3.54) as

$$y(t+h) \;=\; y_1 + y_2,\;\; y_1 = h\nu(t),\;\; y_2 = [I + hC(t)]y(t). \tag{3.55}$$

Now using (3.48), one can see that

$$y_1 \;\in\; E(hF, h^2B). \tag{3.56}$$

But in Chernousko[1], (3.56) was misprinted as

$$y_1 \;\in\; E(f, h^2B). \tag{3.57}$$

Let (3.52) hold for some $t \geq 0$. The vector $y(t)$ belonging to the reachable set (3.51) implies that the vector y_2 from (3.55) belongs to the set D'. Using the affine transformation law, it is observed that the following inclusion holds

$$\begin{aligned} E[(I + hC)a_-, (I + hC)Q_-(I + hC^T)] \;&\subset D' \\ &\subset E[(I + hC)a_+, (I + hC)Q_+(I + hC^T)]. \end{aligned} \tag{3.58}$$

The vector $y(t+h)$ belongs to the reachable set $D[t+h, 0, E(a_0, Q_0)]$. This vector is represented in (3.55) as a sum of the two vectors y_1 and y_2. The first vector y_1 belongs to the ellipsoid (3.56) that degenerates into a point as $h \to 0$ and the second one y_2 belongs to the set D' which satisfying the two sided ellipsoidal estimates (3.58). Now the two sided ellipsoidal approximation of the reachable set $D[t+h, 0, E(a_0, Q_0)]$ is computed in the following way. The inner ellipsoidal approximation is computed by taking the sum of two ellipsoids: (3.56) and ellipsoid in the left-hand side of (3.58), whereas the outer estimate is computed by considering the outer ellipsoidal

approximation for the sum of an ellipsoid (3.56) and ellipsoid in the right-hand side of (3.58). Now using (3.55)-(3.58), one can have

$$\begin{aligned} E[a_-(t+h), Q_-(t+h)] &\subset D[t+h, 0, E(a_0, Q_0)] \\ &\subset E[a_+(t+h), Q_+(t+h)], \\ a_\pm(t+h) &= hf + (I + hC)a_\pm, \\ Q_-(t+h) &= (I + hC)Q_-(I + hC^T) + \\ &\quad +2hA^{-1}(ABA^T)^{1/2}(AQ_-A^T)^{1/2}A^{-T}, \\ Q_+(t+h) &= (I + hC)Q_+(I + hC^T) + h(q^{-1}B + qQ_+), \\ q &= (n^{-1}\mathrm{Tr}[(Q_+)^{-1}B])^{1/2}. \end{aligned} \tag{3.59}$$

The nonsingular matrix $A \in \mathbb{R}^{n\times n}$ is chosen in such a way that both the matrices ABA^T AQ_-A^T are diagonal. The arguments t of the functions f, C, a_-, a_+, Q_- and Q_+ are omitted for the brevity.

Rewriting equations (3.59) for a_- and a_+:

$$\begin{aligned} h^{-1}[a_-(t+h) - a_-(t)] &= Ca_-(t) + f, \\ h^{-1}[a_+(t+h) - a_+(t)] &= Ca_+(t) + f. \end{aligned}$$

As $h \to 0$,

$$\dot{a}_- = Ca_-(t) + f, \qquad \dot{a}_+ = Ca_+(t) + f. \tag{3.60}$$

Simplifying (3.59) for Q_- and Q_+,

$$\begin{aligned} Q_-(t+h) - Q_- &= h(CQ_- + Q_-C^T + 2A^{-1}(ABA^T)^{1/2}(AQ_-A^T)^{1/2}A^{-T}), \\ Q_+(t+h) - Q_+ &= h(CQ_+ + Q_+C^T + qQ_+q^{-1}B). \end{aligned}$$

Dividing both sides of these equations by h and the letting $h \to 0$,

$$\begin{aligned} \dot{Q}_- &= CQ_- + Q_-C^T + 2A^{-1}(ABA^T)^{1/2}(AQ_-A^T)^{1/2}A^{-T}, \\ \dot{Q}_+ &= CQ_+ + Q_+C^T + qQ_+ + q^{-1}B, \\ q &= (n^{-1}\mathrm{Tr}[(Q_+)^{-1}B])^{1/2}. \end{aligned} \tag{3.61}$$

It was assumed that (3.52) holds at some time instant $t \geq 0$ and then was proved that these inclusion also hold for $t+h, h > 0$. But for the initial point $t = 0$, (3.52) are satisfied if

$$\begin{aligned} a_-(0) &= a_+(0) &&= a_0, \\ Q_-(0) &= Q_+(0) &&= Q_0 \end{aligned} \tag{3.62}$$

are chosen.

Therefore, (3.52) hold for all $t \geq 0$. The parameter of the approximating ellipsoid satisfy the differential equation (3.60) and (3.61) and the initial

equations (3.62). Since the vectors a_- and a_+ satisfy the same equation and initial conditions, one can have $a_-(t) = a_+(t) = a(t)$, and $a(t)$ satisfies the following IVP

$$\dot{a} \;=\; C(t)a + F(t), \quad a(0) = a_0. \tag{3.63}$$

Now rewriting (3.52),

$$E[a(t), Q_-(t)] \subset \; D[t, 0, E(a_0, Q_0)] \; \subset E[a(t), Q_+(t)]. \tag{3.64}$$

According to (3.60) and (3.61), the matrix $Q_-(t)$ satisfies the differential equation and the initial condition:

$$\begin{aligned}
\dot{Q}_- &= CQ_- + Q_-C^T + 2A^{-1}(ABA^T)^{1/2}(AQ_-A^T)^{1/2}A^{-T} \\
&= CQ_- + Q_-C^T + 2B^{1/2}(B^{-1/2}Q_-B^{-1/2})^{1/2}B^{1/2}, \\
&= CQ_- + Q_-C^T + 2Q_-^{1/2}((Q_-)^{-1/2}B(Q_-)^{-1/2})^{1/2}Q_-^{1/2}, \\
Q_-(0) &= Q_0, \quad B = KGK^T.
\end{aligned} \tag{3.65}$$

And for matrix $Q_+(t)$ there is the following differential equation and the initial condition:

$$\begin{aligned}
\dot{Q}_+ &= CQ_+ + Q_+C^T + qQ_+ + q^{-1}B, \quad Q_+(0) = Q_0, \\
q &= (n^{-1}\mathrm{Tr}[Q_+^{-1}B])^{1/2}, \quad B = KGK^T.
\end{aligned} \tag{3.66}$$

So, the two-sided ellipsoidal estimates (3.64) for the reachable set of the equations (3.48)-(3.50) are computed. The centers of the approximating ellipsoids coincide and their evolution is described by linear differential equation (3.63) whereas the differential equations (3.65) and (3.66) are used for evolution of the matrices Q_- and Q_+ of the approximating ellipsoids.

Now if there is no control involved, then the following differential equation for both matrices Q_- and Q_+ of the approximating ellipsoids is obtained

$$\dot{Q}_\pm \;=\; CQ_\pm + Q_\pm C^T, \quad Q_\pm(0) = Q_0. \tag{3.67}$$

In the next chapter, the new notation is used in the development of this chapter.

$$a_\pm(t) \;=\; y_{c\pm}(t), \;\; y_{c\pm}(t) \in \mathbb{R}^n \tag{3.68}$$

and

$$\begin{aligned}
(Q_\pm)^{-1} &= \Delta_y^{-1}U_\pm^T U_\pm, \\
B^{-1} &= \Delta_c^{-1}S^T S
\end{aligned} \tag{3.69}$$

for some symmetric positive definite matrices $(S^TS)^{-1} \in \mathbb{R}^{n\times n}$ and $(U_\pm^TU_\pm)^{-1} \in \mathbb{R}^{n\times n}$ and some numbers $\Delta_y = \Delta_{y\pm} \geq 0$ and $\Delta_c =\geq 0$. Now differentiate (3.68) with respect to t

$$\dot{a}(t) \;=\; \dot{y_c}(t). \tag{3.70}$$

Comparing (3.63) and (3.70) and taking $a = y_c$, the IVP becomes,

$$y_{c\pm}(0) = y_c(0), \qquad \dot{y}_{c\pm}(t) = C(t)y_{c\pm}(t) + F(t). \tag{3.71}$$

Rewriting (3.69),

$$\begin{aligned} B &= \Delta_c(S^TS)^{-1}, \\ Q_\pm &= \Delta_y(U_\pm^TU_\pm)^{-1}. \end{aligned} \tag{3.72}$$

Now differentiate second equation with respect to t to find the differential equation for inner ellipsoid and for brevity take $Q_- = Q$,

$$\begin{aligned} \dot{Q} &= -\Delta_y U^{-1}\dot{U}U^{-1}U^{-T} - \Delta_y U^{-1}U^{-T}\dot{U}^TU^{-T} \\ &= -\Delta_y U^{-1}(\dot{U}U^{-1} + U^{-T}\dot{U}^T)U^{-T}. \end{aligned}$$

Now inserting the value of $\dot{Q}$ from (3.65),

$$-\Delta_y U^{-1}(\dot{U}U^{-1} + U^{-T}\dot{U}^T)U^{-T} = \Delta_y U^{-1}U^{-T}C^T + \Delta_y CU^{-1}U^{-T}$$
$$+2(\Delta_y^{-1}U^TU)^{-1/2}\sqrt{(\Delta_y^{-1}U^TU)^{1/2}(\Delta_c^{-1}S^TS)^{-1}(\Delta_y^{-1}U^TU)^{1/2}}(\Delta_y^{-1}U^TU)^{-1/2}.$$

Simplifying the above expression, one can see that

$$\begin{aligned} &(UC + \dot{U})U^{-1} + U^{-T}(C^TU^T + \dot{U}^T) \\ &+2pU(U^TU)^{-1/2}\sqrt{(U^TU)^{1/2}(S^TS)^{-1}(U^TU)^{1/2}} \\ &(U^TU)^{-1/2}U^T = 0, \quad p = \sqrt{\frac{\Delta_c}{\Delta_y}} \\ &U^T(UC + \dot{U}) + (C^TU^T + \dot{U}^T)U \\ &+2pU^TU(U^TU)^{-1/2}\sqrt{(U^TU)^{1/2}(S^TS)^{-1}(U^TU)^{1/2}} \\ &(U^TU)^{-1/2}U^TU = 0, \quad p = \sqrt{\frac{\Delta_c}{\Delta_y}}. \end{aligned}$$

or

$$0 = U^T(UC + \dot{U} + pU(U^TU)^{1/2}\sqrt{(U^TU)^{1/2}(S^TS)^{-1}(U^TU)^{1/2}}(U^TU)^{-1/2}U^TU +$$
$$+\Big(U^T(UC + \dot{U} + pU(U^TU)^{1/2}\sqrt{(U^TU)^{1/2}(S^TS)^{-1}(U^TU)^{1/2}}(U^TU)^{-1/2}U^TU\Big)^T,$$
$$p = \sqrt{\frac{\Delta_c}{\Delta_y}}.$$

The IVP for inner ellipsoid now becomes

$$\begin{aligned} &\dot{U_-} = -U_-C- \\ &\qquad -pU_-(U_-^TU_-)^{1/2}\sqrt{(U_-^TU_-)^{1/2}(S^TS)^{-1}(U_-^TU_-)^{1/2}}(U_-^TU_-)^{-1/2}U_-^TU_-, \\ &U_-(0) = U_0, \quad p = \sqrt{\frac{\Delta_c}{\Delta_y}}. \end{aligned} \tag{3.73}$$

To find the differential equation for the outer ellipsoid, differentiate second equation of (3.72) with respect to t and for brevity take $Q_+ = Q$,

$$\begin{aligned}\dot{Q} &= -\Delta_y U^{-1}\dot{U}U^{-1}U^{-T} - \Delta_y U^{-1}U^{-T}\dot{U}^T U^{-T} ss \\ &= -\Delta_y U^{-1}(\dot{U}U^{-1} + U^{-T}\dot{U}^T)U^{-T}.\end{aligned}$$

Inserting the value of $\dot{Q}$ from (3.66),

$$\begin{aligned}-\Delta_y U^{-1}(\dot{U}U^{-1} + U^{-T}\dot{U}^T)U^{-T} &= \Delta_y C U^{-1}U^{-T} + \Delta_y U^{-1}U^{-T}C^T \\ &\quad + q\Delta_y U^{-1}U^{-T} + q^{-1}\Delta_c S^{-1}S^{-T}, \\ q = \sqrt{n^{-1}\mathrm{Tr}[\Delta_y^{-1}U^T U(\Delta_c^{-1}S^T S)^{-1}]}.&\end{aligned} \tag{3.74}$$

By taking $q = p\mu$ with $p = \sqrt{\Delta_c/\Delta_y}$, $\mu = \sqrt{n^{-1}\mathrm{Tr}[U^T U(S^T S)^{-1}]}$, the above equation becomes

$$\begin{aligned}U^{-1}(\dot{U}U^{-1} + U^{-T}\dot{U}^T)U^{-T} + CU^{-1}U^{-T} \\ + U^{-1}U^{-T}C^T + p\mu U^{-1}U^{-T} + \tfrac{p}{\mu}S^{-1}S^{-T} = 0.\end{aligned} \tag{3.75}$$

The simplification results into the following expression,

$$\begin{aligned}U^T(\dot{U} + UC + \tfrac{p\mu}{2}U + \tfrac{p}{2\mu}U(S^T S)^{-1}U^T U) + \\ + (U^T(\dot{U} + UC + \tfrac{p\mu}{2}U\tfrac{p}{2\mu}U(S^T S)^{-1}U^T U))^T = 0.\end{aligned} \tag{3.76}$$

Therefore the IVP for the outer ellipsoid becomes

$$\begin{aligned}\dot{U_+} &= -U_+ C + \frac{p\mu}{2}U_+ + \frac{p}{2\mu}U_+(S^T S)^{-1}U_+^T U_+, \;\; U_+(0) = U_0, \\ p &= \sqrt{\frac{\Delta_c}{\Delta_y}}, \;\; \mu = \sqrt{n^{-1}\mathrm{Tr}[U_+^T U_+(S^T S)^{-1}]}.\end{aligned} \tag{3.77}$$

Chapter 4

Ellipsoidal bounds using differential inequalities

Dynamic system with bounded disturbances such as electrical, mechanical and other systems whose parameters can vary in an uncertain way, arise in many applications. These uncertainties may arise in the form of stiffness, electric resistance, capacitance, etc. In order to overcome these problems, controlled dynamical systems are introduced.

In this chapter, we will develop a theory to approximate the ellipsoidal bounds by means of conditional differential inequalities (CDI). Then using this concept, we will develop the results to compute ellipsoidal approximation of linear dynamic system both with control and without control.

4.1 Conditional differential inequalities

An inequality connecting the arguments, the unknown function and its derivative is known as differential inequality. For example, $y'(t) \geq F(t, y(t))$, where y is an unknown function of the argument t.

The theory of differential inequalities plays an essential role to compute our enclosures of ODEs. It dates back to Gronwall [31]. Kamke [36] proved and made use of an inequality which was, in fact, that of Gronwall [85, p 12]. In 1964, Walter [107] gave a systematic approach to this theory. Lakshmikantham & Leela [46] in 1969 used differential inequalities in order to construct the sup and sub functions. In 1980, Adams [1] characterized the system of ODEs for which tight inclusions can be computed by using differential inequalities. Nickel [45] in 1983 computed bounds for the set of solutions of functional-differential equations using the concept of differential inequalities. In 1994, Neumaier [83] used logarithmic norms and differential inequalities to compute rigorous enclosures for the solution of dissipative differential equations. He gave an idea to automatically compute

enclosures of the solutions of stiff ODEs without step size restrictions. In 2002, PAPAMICHAIL & ADJIMAN constructed bounds on the set of solutions of parameter dependent ODEs and their second order sensitivities using idea of differential inequalities.

The following theorem gives an idea of how to compute the error bounds of ODEs by using differential inequalities. This theorem is a generalized form of a theorem from Walter [107, pp. 65-65].

4.1.1 Theorem. *Let* $d,\ e \in C^1(\underline{t}, \overline{t})$ *and* $(t, d(t)),\ (t, e(t)) \in D(f)$ *for* $t \in [\underline{t}, \overline{t}[$. *If*

$$d(\underline{t}) \ \leq \ e(\underline{t}), \tag{4.1}$$

$$d'(t) < f(d(t)), \quad e'(t) \ \geq \ f(e(t)) \quad \text{for all} \ \ t \in \left[\underline{t}, \overline{t}\right[, \tag{4.2}$$

then

$$d(t) \leq e(t) \quad \text{for all} \ \ t \in \left[\underline{t}, \overline{t}\right[. \tag{4.3}$$

Proof. If we define $\Delta(t) := e(t) - d(t)$, we get $\Delta(\underline{t}) = e(\underline{t}) - d(\underline{t}) \geq 0$. If $\Delta(\underline{t}) > 0$, then

$$\Delta(t_1) = \Delta(\underline{t}) + \Delta'(\underline{t})t_1 + o(t_1) > 0 \text{ for small } t_1 > \underline{t}. \tag{4.4}$$

Otherwise $d(\underline{t}) = e(\underline{t})$, and (4.1) and (4.2) imply that $d'(\underline{t}) < f(d(\underline{t})) = f(e(\underline{t})) \leq e'(\underline{t})$, which shows that (4.4) also holds in this case. It follows from the continuity that

$$d(t_1) < e(t_1) \quad \text{for small} \ \ t_1 > 0. \tag{4.5}$$

Now we assume that

$$d(t_2) > e(t_2) \quad \text{for some} \ \ t_2 \in \left[\underline{t}, \overline{t}\right[. \tag{4.6}$$

Since d and e are continuous, (4.5) and (4.6) imply that there is a point $t_0 \in]t_1, \overline{t}[$ at which $d(t_0) = e(t_0),\ \ d(t) < e(t)$ for all $t_1 \leq t < t_0$. For $t \uparrow t_0$, this gives

$$\frac{d(t) - d(t_0)}{t - t_0} > \frac{e(t) - e(t_0)}{t - t_0} \quad \text{for all} \ \ t_1 \leq t < t_0.$$

In the limit $t \uparrow t_0$, we get

$$d'(t_0) \geq e'(t_0) \tag{4.7}$$

but as $d(t_0) = e(t_0)$, (4.2) implies $d'(t_0) < f(d(t_0)) = f(e(t_0)) \leq e'(t_0)$, contrary to (4.7). This contradiction shows that the assumption (4.6) cannot hold. Therefore (4.3) holds. □

As a consequence of Theorem 4.1.1, we prove the next result which can be translated into an algorithm to compute the defect estimates.

4.1.2 Theorem. *Given an interval $[\underline{t}, \overline{t}]$, an increasing sequence $\Delta_0 < \cdots, < \Delta_L$ of numbers, and a second sequence $\Delta'_0 \cdots, \Delta'_{L-1}$.*

Let $\Delta : [\underline{t}, \overline{t}] \to \mathbb{R}$ be a function with

$$\Delta(\underline{t}) \;\; \leq \;\; \Delta_J \tag{4.8}$$

for some $J \in \{1, \cdots, L-1\}$ and for all $t \in [\underline{t}, \overline{t}]$,

$$\Delta'(t) \;\; < \;\; \Delta'_l \;\; \text{if} \;\; \Delta_{l-1} < \Delta(t) \leq \Delta_l, \;\; l = 1, 2, \ldots L-1. \tag{4.9}$$

Let $\overline{\Delta} : [\underline{t}, \overline{t}] \to \mathbb{R}$ be the continuous piecewise linear function defined as follows.
Case I. If $\Delta'_J \Delta'_{J-1} > 0$ and $\Delta'_J > 0$, define

$$s_J := \underline{t}, \qquad s_{l+1} = s_l + (\Delta_{l+1} - \Delta_l)/\Delta'_l, \quad \text{for } l = J, \cdots, K,$$

where K is the largest number greater or equal to J with $\Delta'_J, \cdots, \Delta'_K > 0$, and

$$\overline{\Delta}(t) := \begin{cases} \Delta_l + \Delta'_l(t - s_l) & \text{if } s_K \geq \overline{t} \text{ or } K \neq L \;, \\ \Delta_l & \text{otherwise} \end{cases} \tag{4.10}$$

for $t \in [s_l, s_{l+1}]$, $l = J, \cdots .K$
Case II. If $\Delta'_J \Delta'_{J-1} > 0$ and $\Delta'_J < 0$, define

$$s_{J-1} := \underline{t}, \qquad s_{l+1} = s_l - (\Delta_{l+1} - \Delta_l)/\Delta'_l, \quad \text{for } l = J-1, \cdots, K,$$

where K is the smallest number smaller or equal to $J-1$ with $\Delta'_{J-1}, \cdots, \Delta'_K < 0$, and

$$\overline{\Delta}(t) := \begin{cases} \Delta_{l+1} + \Delta'_l(t - s_l), & \text{if } s_K \geq \overline{t} \text{ or } K \neq 0 \;, \\ \Delta_{l+1} & \text{otherwise} \end{cases} \tag{4.11}$$

for $t \in [s_l, s_{l+1}]$, $l = J-1, \cdots, K$.
Case III. Otherwise, define

$$\overline{\Delta}(t) := \Delta_J, \;\; \text{for all } t \in [\underline{t}, \overline{t}]. \tag{4.12}$$

Then

$$\Delta(t) \;\; \leq \;\; \overline{\Delta}(t) \;\; \text{for all} \;\; t \in [\underline{t}, \overline{t}]. \tag{4.13}$$

Proof. We check the continuity of $\overline{\Delta}$ for all cases separately.

Suppose $\Delta'_J, \cdots, \Delta'_K > 0$, then for $l = J, \cdots, K$, we apply the limit $t \uparrow s_{l+1}$, and get

$$\begin{aligned} \overline{\Delta}(s_{l+1}) &= \Delta_l + \Delta'_l(s_{l+1} - s_l) \\ &= \Delta_l + ((\Delta_{l+1} - \Delta_l)/(s_{l+1} - s_l))(s_{l+1} - s_l)) = \Delta_{l+1}. \end{aligned}$$

Taking the limit $t \downarrow s_{l+1}$, we get

$$\overline{\Delta}(s_{l+1}) = \Delta_{l+1} + \Delta'_{l+1}(s_{l+1} - s_{l+1}) = \Delta_{l+1},$$

which implies that $\overline{\Delta}(t)$ is continuous for all $t \in [s_l, s_{l+1}], l = J, \cdots, K, J < K$.

If $\Delta'_{J-1}, \cdots, \Delta'_K < 0$, then, for all $l = J-1, \cdots, K$, with $J > K$ we take the limit $t \uparrow s_{l+1}$, and get

$$\begin{aligned} \overline{\Delta}(s_{l+1}) &= \Delta_{l+1} + \Delta'_l(s_{l+1} - s_l) \\ &= \Delta_{l+1} + (\Delta_l - \Delta_{l+1})/(s_{l+1} - s_l)(s_{l+1} - s_l) \\ &= \Delta_l. \end{aligned}$$

Taking the limit $t \downarrow s_{l+1}$, we get

$$\overline{\Delta}(s_{l+1}) = \Delta_l + \Delta'_{l-1}(s_{l+1} - s_{l+1}) = \Delta_l.$$

This shows that $\overline{\Delta}(t)$ is continuous for all $t \in [s_l, s_{l+1}], l = J, \cdots, K,\ J > K$.

When $\Delta'_J = 0$, $\overline{\Delta}(t)$ is constant for all $t \in [\underline{t}, \overline{t}]$, therefore, continuity holds.

Now let f be the piecewise constant function defined on $[0, \Delta_L[$ with

$$f(\Delta) = \Delta'_l, \tag{4.14}$$

where Δ'_l is constant for $l = 1, 2, \cdots L-1$ and

$$\Delta_l \leq \Delta < \Delta_{l+1}, \quad \text{if} \quad \Delta'_l > 0, \tag{4.15}$$

$$\Delta_{l-1} < \Delta \leq \Delta_l, \quad \text{if} \quad \Delta'_l < 0. \tag{4.16}$$

When $\Delta'_J > 0$, we differentiate $\overline{\Delta}(t)$ and using (4.14) for $t \in [s_l, s_{l+1}]$, we get $\overline{\Delta'}(t) = \Delta'_l = f(\overline{\Delta}(t))$ if $\Delta_l \leq \overline{\Delta}(t) < \Delta_{l+1},\ \ l = J, \cdots, K$.

When $\Delta'_{J-1} < 0$, we differentiate $\overline{\Delta}(t)$ and using (4.14) for $t \in [s_l, s_{l+1}]$, we get $\overline{\Delta'}(t) = \Delta'_l = f(\overline{\Delta}(t))$ if $\Delta_l < \overline{\Delta}(t) \leq \Delta_{l+1},\ \ l = J-1, \cdots, K,\ \ J > K$. So, we can apply Theorem 4.1.1 with Δ in place of d and $\overline{\Delta}$ in place of e and see that (4.13) holds. □

We illustrate in Figure 4.1 ,the resulting family of solutions $\overline{\Delta}$ for the case where $L = 13$,

$$\begin{aligned} (\Delta_0, \cdots, \Delta_L) &= (-\inf; -200; -160; -120; -80; -30; -10; 0; ... \\ &\quad 30; 60; 100; 160; 190; 200; \inf), \\ (\Delta'_0, \cdots, \Delta'_L) &= (1; 5; 1; -4; -1; 4; 1; -5; -1; -4; 1; 4; -5). \end{aligned}$$

In the first version of our implementation, we used Theorem 4.1.2 to compute the bounds for the solutions of the system of ODEs. But later we developed the following result assuming a conditional differential inequality that is more

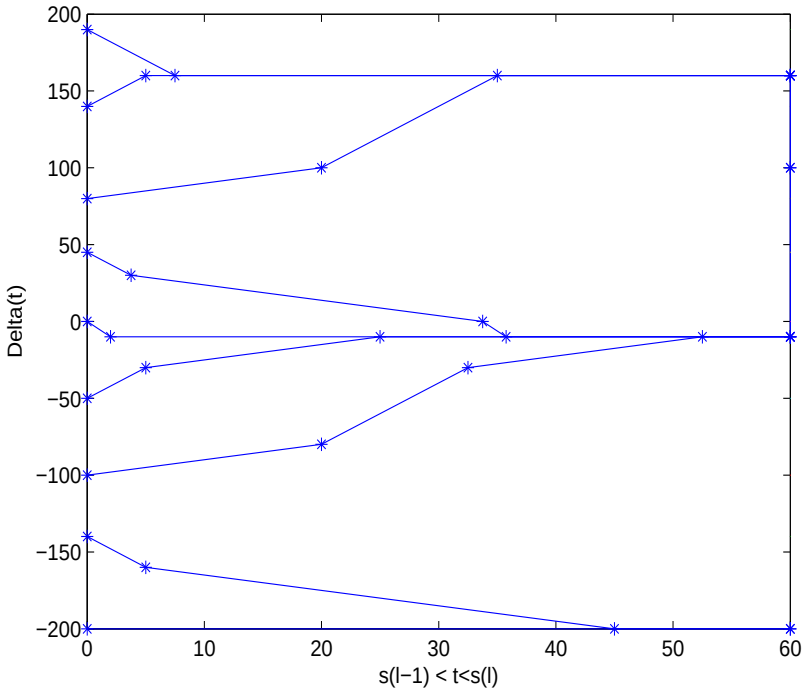

Figure 4.1: Worst situation violating $s_L \geq \overline{t}$

powerful than the traditional approach. This technique is more efficient for adaptive usage. It is the basis of our new method to compute the bounds for solution of the system of ODEs.

Following Theorem is applied when bounds growth is expected (see Case I in section 5.1)

4.1.3 Theorem. *Suppose that* $\Delta(\underline{t}) \leq \Delta_0 < \Delta_+$ *and for all* $t \in [\underline{t}, \overline{t}]$

$$\Delta(t) \in [\Delta_0, \Delta_+] \;\Rightarrow\; \Delta'(t) \leq \Delta'_+. \tag{4.17}$$

Then

$$\Delta(t) \leq \Delta_0 + \Delta'(t - \underline{t}) \quad \text{for } t \in [\underline{t}, t_1], \tag{4.18}$$

where

$$t_1 = \overline{t}, \quad \Delta' = 0, \quad \text{if } \Delta'_+ \leq 0 \tag{4.19}$$

and

$$t_1 = \min(\overline{t}, \underline{t} + (\Delta_+ - \Delta_0)/\Delta'_+), \quad \Delta' = \Delta'_+, \quad \text{if } \Delta'_+ > 0. \tag{4.20}$$

Proof. We first make the stronger assumption that

$$\Delta(t) \in [\Delta_0, \Delta_+] \;\Rightarrow\; \Delta'(t) < \Delta'_+, \quad \text{for all } t \in [\underline{t}, \overline{t}] \tag{4.21}$$

and show that (4.18) holds with (4.19) and (4.20).

Let t_0 be the sequence of all $t_1 \le \overline{t}$ such that (4.18) holds. We define $\delta(t) := \Delta_0 + \Delta'(t - \underline{t}) - \Delta(t)$, and hence $\delta(\underline{t}) = \Delta_0 - \Delta(\underline{t}) \ge 0$. If $\delta(\underline{t}) > 0$, then

$$\delta(t') \;=\; \delta(\underline{t}) + \delta'(\underline{t})(t' - \underline{t}) + o(t' - \underline{t}) > 0 \quad \text{for small } t' > \underline{t}. \tag{4.22}$$

Otherwise $\Delta(\underline{t}) = \Delta_0$, hence $\delta'(\underline{t}) = \Delta' - \Delta'(\underline{t}) \ge \Delta'_+ - \Delta'(\underline{t}) > 0$ by (4.21) and we see that (4.22) holds too. Thus (4.22) holds generally.

By continuity, we see that (4.18) also holds for t_0 in place of t_1. Therefore, if (4.18) is violated, then $t_0 < t_1$, and for some sequence $t_l \downarrow t_0$, we have

$$\Delta(t_l) \;>\; \Delta_0 + \Delta'(t_l - \underline{t}).$$

Taking the limit, we find that

$$\Delta(t_0) \;=\; \Delta_0 + \Delta'(t_0 - \underline{t}) \ge \Delta_0, \tag{4.23}$$

$$\Delta'(t_0) \;=\; \lim_{t_l \to t_0} \frac{\Delta(t_l) - \Delta(t_0)}{t_l - t_0} \ge \Delta' \ge \Delta'_+. \tag{4.24}$$

Because of (4.21), (4.23) and (4.24), we see that $\Delta(t_0) > \Delta_+$. But then (4.23) implies $\Delta'(t_0 - \underline{t}) > \Delta_+ - \Delta_0 > 0$, hence

$$\Delta' \;> 0, \quad t_0 > \underline{t} + \frac{\Delta_+ - \Delta_0}{\Delta'} \ge t_1, \tag{4.25}$$

a contradiction. Now we return to the general case and assume that (4.17) holds. Then for all $\epsilon > 0$, we see that

$$\Delta(t) \in [\Delta_0, \Delta_+] \;\Rightarrow\; \Delta'(t) < \Delta'_+ + \epsilon. \tag{4.26}$$

Therefore, (4.18) holds with

$$t_1 = \overline{t}, \quad \Delta' = 0, \quad \text{if } \Delta'_+ + \epsilon \le 0 \tag{4.27}$$

and

$$t_1 = \min(\overline{t}, \underline{t} + (\Delta_+ - \Delta_0)/(\Delta'_+ + \epsilon)), \quad \Delta' = \Delta'_+ + \epsilon, \quad \text{if } \Delta'_+ + \epsilon > 0. \tag{4.28}$$

We now distinguish three cases from (4.28) as follows:

(i). $\Delta'_+ > 0$, then for $\epsilon \to 0$, we have
$t_1 = \underline{t} + (\Delta_+ - \Delta_0)/\Delta'_+, \quad \Delta' = \Delta'_+ > 0,$
so (4.18) holds with (4.20).

(ii). $\Delta'_+ = 0$, then $t_1 = \overline{t}$, $\Delta' = 0$, so (4.18) holds with (4.19).

(iii). $\Delta'_+ < 0$, then for small $\epsilon > 0$, we have (4.27) so for $\epsilon \to 0$, (4.18) holds with (4.19). Thus the Theorem 4.1.3 holds in each case.

□

We use following result when bounds are decaying (see Case II in section 5.1).

4.1.4 Theorem. *Let* $\Delta_- < \Delta_0$ *and for all* $t \in [\underline{t}, \overline{t}]$

$$\Delta(t) \in [\Delta_-, \Delta_0] \;\Rightarrow\; \Delta'(t) \leq \Delta'_-. \tag{4.29}$$

If $\Delta'_- < 0$, *then*

$$\Delta(t) \leq \Delta_0 + \Delta'(t - \underline{t}) \quad \text{for } t \in [\underline{t}, t_1], \tag{4.30}$$

where

$$t_1 = \min(\overline{t}, \underline{t} + (\Delta_- - \Delta_0)/\Delta'_-, \quad \Delta' = \Delta'_-. \tag{4.31}$$

Proof. This is proved by essentially the same argument as in proof of Theorem 4.1.3. □

4.2 The generalized singular value decomposition

The generalized singular value decomposition(GSVD) of two nonsingular matrices $U_1 \in \mathbb{R}^{m\times r}$ and $U_2 \in \mathbb{R}^{n\times r}$ is factorized as

$$U_i = V_i \Sigma_i R Q^T, \;\; i = 1, 2, \tag{4.32}$$

where $V_1 \in \mathbb{R}^{m\times r}$, $V_2 \in \mathbb{R}^{n\times r}$ and $Q \in \mathbb{R}^{r\times r}$ are orthogonal matrices, $R \in \mathbb{R}^{r\times r}$ is a nonsingular upper triangular matrix. $\Sigma_1 \in \mathbb{R}^{r\times r}$ and $\Sigma_2 \in \mathbb{R}^{r\times r}$ are diagonal matrices with elements

$$\begin{aligned} \Sigma_i \quad &= \quad \mathrm{Diag}(\sigma_{i1}, \cdots, \sigma_{ir}), \;\; i = 1, 2, \\ \sigma_{1j} \geq 0, \quad \sigma_{2j} \geq 0, \quad &j = 1, \cdots, r, \;\; \sigma_{1j}^2 + \sigma_{2j}^2 = 1. \end{aligned} \tag{4.33}$$

4.2.1 Proposition . *Let* $\Delta_1, \Delta_2 \in \mathbb{R}_+$, *two nonsingular matrices* U_1, U_2 *be given in terms of GSVD (4.32) and*

$$U = V_2 \Sigma R Q^T, \tag{4.34}$$

where $\Sigma = \mathrm{Diag}(\sigma_1, \cdots, \sigma_r) \in \mathbb{R}^{r\times r}$ *is a diagonal matrix with elements*

$$\sigma_j \;=\; \frac{\sigma_{1j}\sigma_{2j}}{\sigma_{1j} + q\sigma_{2j}} = \sigma_{2j} - \frac{q\sigma_{2j}^2}{\sigma_{1j} + q\sigma_{2j}}, \;\; q = \sqrt{\Delta_1/\Delta_2}, \;\; \Delta_2 > 0, \;\; j = 1, \cdots, r. \tag{4.35}$$

Then for every y with

$$\|U(y - y_c)\|^2 \;\leq\; \Delta := \Delta_2, \tag{4.36}$$

where

$$y_c = y_{c1} + y_{c2}, \tag{4.37}$$

there exist vectors y_1 and y_2 such that

$$y \;=\; y_1 + y_2, \tag{4.38}$$

$$\|U_1(y_1 - y_{c1})\|^2 \leq \Delta_1, \qquad \|U_2(y_2 - y_{c2})\|^2 \leq \Delta_2. \tag{4.39}$$

Proof. We define

$$\overline{y} := A^{-1}(y - y_c), \tag{4.40}$$

where the matrix A is chosen to be

$$A \;=\; QR^{-1}, \tag{4.41}$$

so that

$$UA \;=\; V_2\Sigma. \tag{4.42}$$

Therefore,

$$y - y_c \;=\; A\overline{y}. \tag{4.43}$$

We insert (4.43) into (4.36), and get

$$\|UA\overline{y}\|^2 \;\leq\; \Delta, \tag{4.44}$$

or

$$\overline{y}^T(UA)^T UA\overline{y} \;\leq\; \Delta. \tag{4.45}$$

Then (4.34) implies that

$$\overline{y}^T\Sigma^2\overline{y} \;\leq\; \Delta. \tag{4.46}$$

Now we shall prove that there exists vectors $\overline{y}_1$ and $\overline{y}_2$ such that

$$\overline{y} \;=\; \overline{y}_1 + \overline{y}_2, \tag{4.47}$$

and

$$\overline{y}_i^T\Sigma_i^2\overline{y}_i \;\leq\; \Delta_i, \;\; i = 1, 2. \tag{4.48}$$

For this we define

$$\overline{y}_{ij} := \;\sqrt{\Delta_i/\Delta_2}\sigma_j/\sigma_{ij}\overline{y}_j, \quad i = 1, 2, \;\; j = 1, \cdots, r. \tag{4.49}$$

Here $\overline{y}_j$ are the components of the vector $\overline{y}$. By (4.46),

$$\sum_{j=1}^{r} \sigma_j^2 \overline{y}_j^2 \leq \Delta, \;\; j = 1, \cdots, r. \tag{4.50}$$

Now (4.49) and (4.50) together imply that

$$\sum_{j=1}^{r} \sigma_{ij}^2 \overline{y}_{ij}^2 \leq \Delta_i, \;\; i = 1, 2, \tag{4.51}$$

which shows that any $\overline{y}$ satisfying (4.46) is represented as the sum $\overline{y} = \overline{y}_1 + \overline{y}_2$, where ${\overline{y}_i}^T \Sigma_i^2 \overline{y}_i \leq \Delta_i$.

By (4.32) and (4.41), the matrix $U_i A$ is equal to $V_i \Sigma_i$, so that

$$(U_i A)^T U_i A \;\; = \;\; \Sigma_i^2, \;\; i = 1, 2. \tag{4.52}$$

Then (4.48) becomes

$${\overline{y}_i}^T A^T U_i^T U_i A \overline{y}_i \leq \Delta_i, \tag{4.53}$$

or

$$\|U_i A \overline{y}_i\|^2 \;\; \leq \;\; \Delta_i, \;\; i = 1, 2. \tag{4.54}$$

Now we define

$$y_i := y_{ci} + A\overline{y}_i, \;\; i = 1, 2. \tag{4.55}$$

Then

$$y_i - y_{ci} \;\; = \;\; A\overline{y}_i, \;\; i = 1, 2. \tag{4.56}$$

by inserting (4.56) in (4.54), we get the required result. That is,

$$\|U_i(y_i - y_{ci})\|^2 \;\; \leq \;\; \Delta_i, \;\; i = 1, 2. \tag{4.57}$$

□

4.3 Ellipsoidal bounds without control

We consider the differential equation

$$y' = C(t)y + F(t), \tag{4.58}$$

where $y \in \mathbb{R}^n$ is a state vector, and the n-vector $F(t)$ and the $n \times n$ matrix $C(t)$, are piecewise continuous functions of time. The set of initial conditions is given in the form

$$\|U_0(y(0) - y_c(0))\|^2 \;\; \leq \;\; \Delta_y, \tag{4.59}$$

where $U_0 \in \mathbb{R}^{m\times n}$, $y_c \in \mathbb{R}^n$ and $\Delta_y \in \mathbb{R}_+$. The vector $y(t)$ satisfies

$$\|U_0(y(0) - y_c(0))\|^2 \;\leq\; \Delta_y, \quad y'(\tau) = C(\tau)y(\tau) + F(\tau) \text{ for } \tau \in [0,t]. \tag{4.60}$$

The reachable set of (4.58) and (4.60) with initial conditions (4.59) is denoted by

$$D_t \;=\; D[t, 0, \|U_0(y(0) - y_c(0))\|^2 \leq \Delta_y]. \tag{4.61}$$

4.3.1 Proposition. *If U is defined by solving*

$$U'(t) \;=\; -U(t)C(t), \quad U(0) = U_0, \tag{4.62}$$

and if we define

$$Q(t) := \Delta_y(U^T(t)U(t))^{-1}, \tag{4.63}$$

then

$$Q'(t) \;=\; C(t)Q(t) + Q(t)C(t)^T \tag{4.64}$$

holds

Proof. First we will show that U is nonsingular. For that purpose, we differentiate $\ln\det(U(t))$ with respect to variable t and get

$$\begin{aligned}\frac{d}{dt}(\ln(\det(U(t)))) &= Tr(U(t)^{-1}U'(t)) \\ &= -TrC(t) \quad \text{from (4.62)} \\ &= \gamma(t) \quad \text{(say)}.\end{aligned} \tag{4.65}$$

Now integrating (4.65), we see that

$$\ln(\det(U(t))) = \int \gamma(t)dt \Rightarrow \det(U(t)) = e^{\int \gamma(t)dt} \neq 0.$$

This implies that U is non-singular. Therefore, U^TU is a symmetric positive definite matrix. This means that Q defined in (4.63) is also a symmetric positive definite matrix.

To show that (4.64) holds, we differentiate (4.63) with respect to t, for brevity omit argument t. Then we get

$$\begin{aligned}Q' &= -\Delta_y(U^TU)^{-1}(U^TU' + (U')^TU)(U^TU)^{-1} \\ &= -U^{-1}U^{-T}U^TU'\Delta_y(U^TU)^{-1} - \Delta_y(U^TU)^{-1}(U')^TUU^{-1}U^{-T} \\ &= -U^{-1}U'\Delta_y(U^TU)^{-1} - \Delta_y(U^TU)^{-1}(U')^TU^{-T}\end{aligned}$$

Replacing the values of C and Q from (4.62) and (4.63) respectively, we get the required result (4.64). □

4.3.2 Theorem. *Let $y_c(t)$ be the solution of IVP*

$$\dot{y}_c(t) = C(t)y_c(t) + F(t), \tag{4.66}$$

let $U(t)$ be the solution of IVP

$$U'(t) = -U(t)C(t),\ \ U(0) = U_0. \tag{4.67}$$

Now if

$$\|U_0(y(0) - y_c(0))\|^2 \ \leq \ \Delta_y,\ \ y'(\tau) = C(\tau)y + F(\tau) \tag{4.68}$$

holds for all $\tau \in [0, t]$, then the following statements hold.

(i) If $y(0) := y_0$ satisfies $\|U_0(y(0) - y_c(0))\|_2^2 \leq \Delta_y$, then the every solution of IVP (4.58) satisfies $\|U(t)(y(t) - y_c(t))\|_2^2 \leq \Delta_y$.
(ii) If y_t satisfies $\|U(t)(y_t - y_c(t))\|_2^2 \leq \Delta_{y-}$, then there exists y_0 with $\|U_0(y(0) - y_c(0))\|_2^2 \leq \Delta_y$, such that the IVP (4.58) has the solution with $y(t) = y_t$.

Proof. (i) If we take

$$f(d) := \ \ \varepsilon \ \ > 0, \tag{4.69}$$

$$d(t) := \|U(t)(y(t) - y_c(t))\|^2 \ \ \text{and} \ \ \ e(t) := \epsilon(t - \underline{t}) = \Delta_y \tag{4.70}$$

in Theorem 4.1.1, then we see that (4.1) and (4.3) hold. We show that the assumption (4.2) is also true. For this we differentiate (4.70) w.r.t. t and for brevity we omit parameter t,

$$d' \ \ = \ \ 2(U(y - y_c))^T(U'(y - y_c) + U(y' - y_c')). \tag{4.71}$$

Using (4.58), (4.66) and (4.67), we get

$$\begin{aligned} d' \ &= \ 2(U(y - y_c))^T(-UC(y - y_c)) + U(Cy + F - (Cy_c + F))) \\ &= \ 0 < \varepsilon = e'. \end{aligned} \tag{4.72}$$

Therefore, Theorem 4.1.1 implies that $d(t) \leq e(t)$ for all $t \in [\underline{t},\ \overline{t}]$. By replacing the values of $d(t)$ and $e(t)$ from (4.70), we get the required result. That is, $\|U(t)(y(t) - y_c(t))\|_2^2 \leq \Delta_y$.

(ii) We take $\Delta_{y-} := \Delta_y + O(h^2)$ and assume that (ii) holds for some $t \geq 0$. We replace (4.58), (4.62) and (4.67) for y, y_c and U with $h > 0$ by Taylor series expansion

$$\begin{aligned} y(t + h) \ &= \ y(t) + hy'(t) + O(h^2) \\ &= \ y(t) + hC(t)y(t) + hF(t) + O(h^2), \end{aligned} \tag{4.73}$$

$$\begin{aligned} y_c(t + h) \ &= \ y_c(t) + hy_c'(t) + O(h^2) \\ &= \ hF(t) + (I + hC(t))y_c(t) + O(h^2), \end{aligned} \tag{4.74}$$

$$\begin{aligned} U(t+h) &= U(t) + hU'(t) + O(h^2) \\ &= U(t)(I + hC(t))^{-1} + O(h^2), \end{aligned} \tag{4.75}$$

respectively

To show that (ii) holds for time instant $t+h,\ h>0$, we may assume that

$$\|U(t+h)(y(t+h) - y_c(t+h))\|^2 \ \leq \ \Delta_y.$$

Inserting the value of $y(t+h)$ from (4.73), we have

$$\begin{aligned} \|U(t+h)(y(t) + hC(t)y(t) + hF(t) + O(h^2) - y_c(t+h))\|^2 &\leq \Delta_y, \\ \|U(t+h)((I + hC(t))y(t) - y_c(t+h) + hF(t) + O(h^2))\|^2 &\leq \Delta_y. \end{aligned} \tag{4.76}$$

Using (4.74) and (4.75) in (4.76), the left hand side becomes

$$U(t)(I + hC(t))^{-1}((I + hC(t))y(t) - (I + hC(t))y_c(t) + O(h^2)).$$

After simplification, we get

$$\|U(t)(y(t) - y_c(t))\|^2 \ \leq \ \Delta_y + O(h^2). \tag{4.77}$$

This implies that there exists $y(t)$ which satisfies (4.60).

Now rewriting (4.74) for $y_c(t)$, we get

$$h^{-1}(y_c(t+h) - y_c(t)) \ = \ C(t)y_c(t) + F(t).$$

As $h \to 0$, we have

$$y_c^{'}(t) \ = \ C(t)y_c(t) + F(t). \tag{4.78}$$

Simplifying (4.75) for $U(t)$,

$$\begin{aligned} U(t+h)(I + hC(t)) &= U(t), \\ h^{-1}(U(t+h) - U(t)) &= -U(t+h)C(t). \end{aligned}$$

Letting $h \to 0$, we get

$$U(t)' \ = \ -U(t)C(t). \tag{4.79}$$

We assumed that (ii) holds at some time $t \geq 0$ and then proved that this implication also holds for $t+h, h>0$. But for initial point $t=0$, (ii) is satisfied if we take

$$\begin{aligned} y_c(0) &= y_{c0}, \\ U(0) &= U_0. \end{aligned} \tag{4.80}$$

Therefore, (ii) holds for all $t \geq 0$. □

4.4 Ellipsoidal approximation with control

Suppose we have a controlled dynamic system described by the following differential equation

$$\dot{y}(t) \;=\; F(t, u(t), y(t)),\;\; t \geq 0, \tag{4.81}$$

$$\|R_c(t)(y(t) - y_c(t)\| \;\leq\; \Delta(t), \tag{4.82}$$

$$\|R(y(0) - y_c(0))\| \;\leq\; \Delta_y. \tag{4.83}$$

Here t is a time instant, $y \in \mathbb{R}^n$ is a state vector, $y_c(t) \in \mathbb{R}^n$, $R_c(t) \in \mathbb{R}^{m\times m}$.

The reachable set $D(t, y_0, \Delta)$ of (4.81) consists of the end points $y(t)$ at $t \geq 0$ of all the trajectories y of (4.81) satisfying (4.81)–(4.83) with $\tau \in [0, t]$ in place of t.

The initial set M and the reachable set $D(t, y_0, \Delta)$ are shown in figure 1.1 for $n = 2$. The set $D(t, y_0, \Delta)$ is the union of all sets $D(t, y_0, 0)$, where $y_0 \in M$.

The reachable sets have a property which implies that the reachable set for instant t can be obtained from the same set for instant τ by prolongation of all trajectories starting from instant τ up to instant t.

These sets play an important role in control theory as they can be represented as the solutions of many controlled dynamical systems. These solutions can be approximated by means of ellipsoids. The ellipsoids are closely connected with quadratic forms and Gaussian probability distribution. Now we will compute the ellipsoidal approximation of the solution of the dynamical system.

An ellipsoid can be defined by the following inequality

$$\|U(y - y_c)\|^2 \;\leq\; \Delta_y, \tag{4.84}$$

where $y_c(t) \in \mathbb{R}^n$ is the center of the ellipsoid, $U \in \mathbb{R}^{m\times n}$ is a nonsingular matrix.

In order to get the parameters of the ellipsoid (4.84), we consider the following vector differential equation

$$y' = C(t)y + K(t)u + F(t), \tag{4.85}$$

$$\|H(t)(u(t) - 0)\|^2 \leq \Delta(t), \tag{4.86}$$

where $y \in \mathbb{R}^n$ is a state vector, $u \in \mathbb{R}^m$ is the control vector, the n-vector $F(t)$ and the $n \times n$ matrix $C(t)$, $n \times m$ matrix $K(t)$,and $m \times m$ matrix $H(t)$ are the piecewise continuous functions of time. The initial conditions are

$$\|U_0(y(0) - y_c(0))\|^2 \;\leq\; \Delta_y. \tag{4.87}$$

Let $p = \sqrt{\Delta_c/\Delta_y}$, and $y_c(t)$ be the solution of IVP

$$y_c^{'}(t) \;=\; C(t)y_c(t) + K(t)u(t) + F(t), \tag{4.88}$$

let $U_-(t)$ be the solution of IVP

$$U_-'(t) \;=\; -U_-(t)C(t) - \sqrt{\Delta_c/\Delta_y}V_2\Sigma_2^2\Sigma_1^{-1}RQ^T \tag{4.89}$$

and let $U_+(t)$ be the solution of IVP

$$\begin{aligned} \dot{U_+} &= -U_+C + \frac{p\mu}{2}U_+ + \frac{p}{2\mu}U_+(S_+^TS_+)^{-1}U_+^TU_+,\ U_+(0) = U_0, \\ \mu &= \sqrt{n^{-1}\mathrm{Tr}[U_+^TU_+(S_+^TS_+)^{-1}]}. \end{aligned} \tag{4.90}$$

4.4.1 Theorem. *If*

$$\|U_0(y(0) - y_c(0))\|^2 \;\leq\; \Delta_y, \tag{4.91}$$

and

$$y'(\tau) = C(\tau)y + K(\tau)u + F(\tau), \tag{4.92}$$

hold for all $\tau \in [0,t]$, then the following statements hold.

(i) If y_0 satisfies $\|U_0(y(0) - y_c(0))\|_2^2 \leq \Delta_y$, then the every solution of IVP (4.85) satisfies $\|U_+(t)(y(t) - y_c(t))\|_2^2 \leq \Delta_y$.
(ii) If y_t satisfies $\|U_-(t)(y_t - y_c(t))\|_2^2 \leq \Delta_y$, then there exists y_0 with $\|U_0(y(0) - y_c(0))\|_2^2 \leq \Delta_y$, such that the IVP (4.85) has the solution with $y(t) = y_t$.

Proof. (i) If we take

$$d(t) := \|U_+(t)(y(t) - y_c(t))\|^2 \quad \text{and} \quad e(t) := \frac{\varepsilon}{\alpha p} + (\Delta_y - \frac{\varepsilon}{\alpha p})e^{\alpha p(t-\underline{t})} \tag{4.93}$$

in Theorem 4.1.1, then we see that (4.1) and (4.3) hold. We show that the assumption (4.2) is also true. For this we differentiate (4.93) with respect to t and for brevity omit the parameter t,

$$d' \;=\; 2(U_+(y - y_c))^T(U_+'(y - y_c) + U_+(y' - \dot{y_c})). \tag{4.94}$$

Now using (4.85), (4.88) and (4.90), with $p = \sqrt{\Delta_\nu/\Delta_y}$ and

$\mu = \sqrt{n^{-1}\mathrm{Tr}[U_+^T U_+ (S_+^T S_+)^{-1}]}$, we have

$$\begin{aligned} d' &= 2(U_+(y-y_c))^T \Big((-U_+C - \frac{p}{2}U_+ - \frac{p}{2\mu}U_+(S^TS)^{-1}U_+^TU_+)(y-y_c) + \\ &\quad + U_+(Cy + Ku + F - (Cy_c + Ku + F)) \Big) \\ &= 2(y-y_c)^T U_+^T \Big(-\frac{p}{2}U_+(y-y_c) - \frac{p}{2\mu}U_+(S^TS)^{-1}U_+^TU_+(y-y_c) - \\ &\quad - U_+(Cy - Cy_c) + U_+(Cy - Cy_c) \Big) \\ &= -p(y-y_c)^T U_+^T U_+(y-y_c) - \frac{p}{\mu}(y-y_c))^T U_+^T U_+ (S^TS)^{-1} U_+^T U_+(y-y_c). \end{aligned} \tag{4.95}$$

Simplifying the above expression, we see

$$d' = -p\|U_+(y-y_c)\|^2 - \frac{p}{\mu}\|S^{-T}U_+^TU_+(y-y_c)\|^2. \tag{4.96}$$

Comparing (4.96) and (4.2), we must show that

$$f(d) + p\|U_+(y-y_c)\|^2 + \frac{p}{\mu}\|S^{-T}U_+^TU_+(y-y_c)\|^2 > 0. \tag{4.97}$$

Now using the relation $\|x\| = \|A^{-1}Ax\| \leq \|A^{-1}\|\,\|Ax\|$, wee see that

$$\|U_+(y-y_c)\|_2^2 \leq \|(S^{-T}U_+^T)^{-1}\|_2^2 \|S^{-T}U_+^TU_+(y-y_c)\|^2.$$

If we put

$$c := \|(U_+S^{-1})^{-T}\|^2 = \|(S^{-T}U_+^T)^{-1}\|^2$$

then

$$\|U_+(y-y_c)\|_2^2 \leq c.\|S^{-T}U_+^TU_+(y-y_c)\|_2^2.$$

Using this relation in (4.97), we get

$$f(d(t)) + pd(t) + \frac{p}{\mu c}d(t) > 0, \text{ for all } t \in [\underline{t}\ \overline{t}]. \tag{4.98}$$

Therefore, we choose

$$f(d) := \epsilon - \alpha p d \text{ and, } \alpha := 1 + \frac{1}{\mu c} \tag{4.99}$$

and a small constant $\epsilon > 0$. Now using Theorem 4.1.1, we see that

$$e(t) = \frac{\varepsilon}{\alpha p} + (\Delta_y - \frac{\varepsilon}{\alpha p})e^{\alpha p(t-\underline{t})}$$

and as $\varepsilon \to 0$, $e(t) = \Delta_y$ which implies that $d(t) \leq e(t)$ for all $t \in [\underline{t}\ \overline{t}]$. By replacing the values of $d(t)$ and $e(t)$ from (4.70), we get the required result. That is, $\|U_+(t)(y(t) - y_c(t))\|_2^2 \leq \Delta_{y+}$.

(ii) We take $\Delta_{y-} := \Delta_y + O(h^2)$ and assume that (ii) holds for some $t \geq 0$. We replace (4.85), (4.88) and (4.89) for y, y_c and U_- with $h > 0$ by Taylor series expansion

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + O(h^2), . \\ &= hF(t) + hK(t)u(t) + (I + hA(t))y(t) + O(h^2). \quad (4.100) \end{aligned}$$

$$\begin{aligned} y_c(t+h) &= y_c(t) + hy_c^{'}(t) + O(h^2) \\ &= hF(t) + hK(t)u(t) + (I + hA(t))y_c(t) + O(h^2). \quad (4.101) \end{aligned}$$

$$\begin{aligned} U_-(t+h) &= U_-(t) + hU_-'(t) + O(h^2) \\ &= U_-(t) - hU_-(t)C(t) - hpV_2\Sigma_2^2\Sigma_1^{-1}RQ^T + O(h^2) \\ &= U_-(t)(I + hC(t))^{-1} - hpV_2\Sigma_2^2\Sigma_1^{-1}RQ^T \\ p &= \sqrt{\Delta_c/\Delta_y} \qquad (4.102) \end{aligned}$$

respectively.

To show that (ii) holds for time instant $t + h,\ h > 0$, we may assume that

$$\|U_-(t+h)(y(t+h) - y_c(t+h))\|^2 \ \leq \ \Delta_y.$$

Inserting the value of $y(t+h)$ from (4.100), we have

$$\begin{aligned} \|U_-(t+h)(y(t) + hC(t)y(t) + hK(t)u(t) + hF(t) + O(h^2) - y_c(t+h))\|^2 \\ \leq \Delta_y, \\ \|U_-(t+h)((I + hC(t))y(t) - y_c(t+h) + hK(t)u(t) + hF(t) + O(h^2))\|^2 \\ \leq \Delta_y. \end{aligned} \qquad (4.103)$$

Using (4.101) and (4.102) in (4.103) and omitting parameter t, the left hand side becomes

$$\begin{aligned} &(U_-(I + hC)^{-1} - pV_2\Sigma_2^2\Sigma_1^{-1}RQ^T)((I + hC)y + hKu + hF, \\ &-(I + hC)y_c - hKu - -hF + O(h^2)), \quad p = \sqrt{\Delta_c/\Delta_y}. \end{aligned}$$

After simplification, we get

$$\begin{aligned} \|U_-(y - y_c) - pV_2\Sigma_2^2\Sigma_1^{-1}RQ^T((I + hC)y - (I + hC)y_c))\|^2 \\ \leq \Delta_y + O(h^2), \quad p = \sqrt{\Delta_c/\Delta_y}. \end{aligned} \qquad (4.104)$$

As $h \to 0$, $\|U_-(t)(y(t) - y_c(t))\|_2^2 \leq \Delta_y$, which implies that there exists $y(t)$ satisfying (4.91) and (4.92).

Now rewriting (4.101) for $y_c(t)$, we get

$$h^{-1}(y_c(t+h) - y_c(t)) = C(t)y_c(t) + K(t)u(t) + F(t).$$

As $h \to 0$, we have

$$y_c^{'}(t) = C(t)y_c(t) + K(t)u(t) + F(t). \tag{4.105}$$

Simplifying (4.102) for $U_-(t)$,

$$\begin{aligned} U_-(t+h)(I + hC(t)) &= U^-(t) - hpV_2\Sigma_2^2\Sigma_1^{-1}RQ^T(I + hC(t)), \\ h^{-1}(U_-(t+h) - U_-(t)) &= -U_-(t+h)C(t) - pV_2\Sigma_2^2\Sigma_1^{-1}RQ^T(I + hC(t)), \\ p &= \sqrt{\Delta_c/\Delta_y}. \end{aligned}$$

As $h \to 0$, we get

$$\begin{aligned} U_-'(t) &= -U_-(t)C(t) - pV_2\Sigma_2^2\Sigma_1^{-1}RQ^T, \\ p &= \sqrt{\Delta_c/\Delta_y}. \end{aligned} \tag{4.106}$$

We assumed that (ii) holds at some time $t \geq 0$ and then proved that this implication also holds for $t + h, h > 0$. But for initial point $t = 0$, (ii) is satisfied if we take

$$\begin{aligned} y_c(0) &= y_{c0}, \\ U_-(0) &= U_0. \end{aligned} \tag{4.107}$$

Therefore (ii) holds for all $t \geq 0$. □

4.4.2 Remark I. Note that [11], if $\sigma_1, \sigma_2, \cdots, \sigma_n$ are the singular values of $(U_+S^{-1})^T$, then by using the property $\mathrm{Tr}AB = \mathrm{Tr}BA$, we see that

$$\mathrm{Tr}(U_+^TU_+S^{-1}S^{-T}) = \mathrm{Tr}(S^{-T}U_+^TU_+S^{-1}) = \mathrm{Tr}(U_+S^{-1})^TU_+S^{-1} = \sum_{i=1}^{n}\sigma_i^2 \geq \sigma_{\max}^2,$$

which gives

$$\mu \geq \sigma_{\max}^2. \tag{4.108}$$

Also since

$$c = \|(U_+S^{-1})^{-T}\|^2 = \sigma_{\max}^{-2}, \tag{4.109}$$

comparing (4.108) and (4.109), $\mu \geq \frac{1}{c}$, or $\mu c \geq 1$. (4.98) reduces into

$$f(d(t)) + pd(t) > 0, \text{ for all } t \in [\underline{t}\ \overline{t}].$$

4.4.3 Remark II. It can be observed that when there is no control involved, then $\Delta_c = 0$ and by comparing (4.89) and (4.90), we get the same differential equation both for inner and outer ellipsoids. That is,

$$U_-'(t) = -U(t)C(t) = U_+'(t).$$

This implies that the inner and the outer bounds are optimal and they differ only by the effect of control uncertainty.

Chapter 5

Error bounds by optimization

In this chapter we shall discuss the techniques used to compute error bounds of ODEs. We shall combine our technique with Chernousko and Kühn's methods to compute error bounds of ODEs. Our defect estimates are based on differential inequalities and global optimization of the constants involved. The bounds are estimated by approximating outer ellipsoids. The initial ellipsoidal approximation is supposed as

$$\|U_0(\boldsymbol{y_0} - u_0)\| \leq \Delta_0,$$

where $\boldsymbol{y_0} \in \mathbb{R}^n$ is a box containing the initial state vector, $u_0 \in \mathbb{R}^n$ is taken to be the midpoint of $\boldsymbol{y_0}$, $U_0 \in \mathbb{R}^{n\times n}$ is a nonsingular matrix (preconditioner) and $\Delta_0 \in \mathbb{R}$ is initial defect estimate. Since we are using outer ellipsoidal approximation that causes an initial wrapping effect, our initial box is wide but the technique presented here makes it feasible to converge the bounds.

A suitable choice of preconditioner improves the efficiency of the method.

5.1 Error bounds by optimization

In this section we will discuss our new scheme to compute the validated enclosures of the solutions of ODEs. The techniques presented by Chernousko [16] and Kühn [43] are combined with new bounds from Theorem 4.1.3 and evaluated with global optimization methods to compute error bounds. We begin with approximate solutions of the differential equations and a t-dependent preconditioning matrix defining the shape of the ellipsoid. We then compute the worst case of the norm of the preconditioned defect by solving a global optimization problem, which produces a value for the size of the enclosing ellipsoid.

To solve the optimization problem, we fix Δ_0 as our starting defect estimate and $\underline{t}$ starting time. We begin with the case when bound growth is expected.

For each polynomial piece $[t_i, t_{i+1}]$ where $t_i := \underline{t}$, we solve optimization problem (5.12) for $[\Delta_0, \Delta_+]$, where

$$\Delta_+ = q\Delta_0, \quad q > 1.$$

. The solution Δ' of this optimization problem tells whether the bounds are decaying or growing. If $\Delta' > 0$, we proceed further to compute the defect estimates $\overline{\Delta}(t), \ t \in [t_i, t_{i+1}]$. While $\Delta' < 0$, implies that the bounds are decaying. In that case we solve the optimization problem (5.17) for $[\Delta_-, \Delta_0]$ to compute the defect estimates $\overline{\Delta}(t), \ t \in [t_i, t_{i+1}]$, where

$$\Delta_- = \Delta_0/q, \quad q > 1$$

. Detailed description of the method is given in section 5.2.

5.1.1 Approximating the solution

We consider the IVP

$$y'(t) \ = \ F(t, y(t)), \quad y(0) \in \boldsymbol{y}_0, \quad t \in [0, T], \tag{5.1}$$

where $F \in C^1(D, \mathbb{R}^n)$ is a continuously differentiable function from $D \subseteq \mathbb{R} \times \mathbb{R}^n$ to $\mathbb{R}^n$, the function $y : [0, T] \to \mathbb{R}^n$ is unknown to be solved for, and $\boldsymbol{y}_0$ is a box. We want to find error bounds for an approximate continuously differentiable solution

$$u(t) \ \approx \ y(t),$$

of (5.1), given as a piecewise polynomial

$$u(t) \ = \ u_i(t) \ \text{ for } \ t_{i-1} \leq t < t_i,$$

where each $u_i(t)$ is a vector of cubic polynomials. The error between the approximate and the exact solution is given by

$$\epsilon(t) = y(t) - u(t).$$

We define

$$\eta(t) \ = \ U(t)\epsilon(t), \tag{5.2}$$

where the preconditioner $U : [0, T] \to \mathbb{R}^{n \times n}$ is a matrix of continuously differentiable piecewise cubic polynomials, given as

$$U(t) \ = \ U_i(t) \ \text{ for } \ t_{i-1} \leq t < t_i,$$

$0 = t_0 < t_1 < \cdots < t_n = T$ and each $U_i(t)$ is a matrix of cubic polynomials.

Given $\underline{t} < \overline{t}$, we define piecewise continuously differentiable solutions u and U by using cubic Hermite interpolation

$$u(t) = a(t-\underline{t})^2(t-\overline{t}) + b(t-\underline{t})^2 + c(t-\underline{t}) + d, \quad t \in [\underline{t}, \overline{t}] \tag{5.3}$$

$$\dot{u}(t) = 2a(t-\underline{t})(t-\overline{t}) + a(t-\underline{t})^2 + 2b(t-\underline{t}) + c, \quad t \in [\underline{t}, \overline{t}] \tag{5.4}$$

and

$$U(t) = A(t-\underline{t})^2(t-\overline{t}) + B(t-\underline{t})^2 + C(t-\underline{t}) + D, \quad t \in [\underline{t}, \overline{t}] \tag{5.5}$$

$$\dot{U}(t) = 2A(t-\underline{t})(t-\overline{t}) + A(t-\underline{t})^2 + 2B(t-\underline{t}) + C, \quad t \in [\underline{t}, \overline{t}], \tag{5.6}$$

where the coefficients a, b, c, d and A, B, C and D are computed by using the continuity of the function and its first derivative and are given as:

$$\begin{aligned} a &:= \Big(\dot{u}(\overline{t}) + \dot{u}(\underline{t}) - 2(u(\overline{t}) - u(\underline{t}))/h\Big)/h^2, \\ b &:= \Big((u(\overline{t}) - u(\underline{t}))/h - \dot{u}(\underline{t})\Big)/h, \\ c &:= \dot{u}(\underline{t}), \\ d &:= u(\underline{t}), \end{aligned} \tag{5.7}$$

and similarly

$$\begin{aligned} A &:= \Big(\dot{U}(\overline{t}) + \dot{U}(\underline{t}) - 2(U(\overline{t}) - U(\underline{t}))/h\Big)/h^2, \\ B &:= \Big((U(\overline{t}) - U(\underline{t}))/h - \dot{U}(\underline{t})\Big)/h, \\ C &:= \dot{U}(\underline{t}), \\ D &:= U(\underline{t}). \end{aligned} \tag{5.8}$$

5.1.2 Error bounds for the preconditioned ODEs

We look for bounds of

$$\Delta(t) = \|\eta(t)\|^2 = \eta(t)^T\eta(t) = \sum \eta_i^2(t) \geq 0, \tag{5.9}$$

where $\|.\|$ denotes the Euclidean norm. Bounds at $t = 0$ come from the following Lemma.

5.1.1 Lemma. *An arbitrary box $\boldsymbol{y}_0$ is contained in the ellipsoid*

$$\mathcal{E} = \{y \in \mathbb{R}^n \mid \|U_0(y - u_0)\|^2 \leq \Delta_0\}, \tag{5.10}$$

where $u_0 = \frac{1}{2}(\overline{y}_0 + \underline{y}_0)$, $\Delta_0 = \|U_0(y-u_0)/2\|^2$. In particular, if $U_0 = \mathrm{Diag}\,(\overline{y}_0 - \underline{y}_0)^{-1}, \Delta_0 = n/4$.

Proof. *If* $y \in \boldsymbol{y}_0$ *then* $|y - u_0| \le \frac{1}{2}(\overline{y}_0 - \underline{y}_0)$, *hence*

$$\begin{aligned} \|U_0(y-u_0))\|^2 &= \sum_{i=1}^{n} (U_0(y-u_0))_i^2 = \sum_{i=1}^{n} \Big(\frac{y_i - u_{0i}}{\overline{y}_{0i} - \underline{y}_{0i}}\Big)^2 \\ &\le \sum_{i=1}^{n} \frac{1}{4}\Big(\frac{\overline{y}_{0i} - \underline{y}_{0i}}{\overline{y}_{0i} - \underline{y}_{0i}}\Big)^2 = \sum_{i=1}^{n} \frac{1}{4} = \frac{n}{4}. \end{aligned}$$

□

Now bounds for (5.9) at $t = 0$ are

$$0 \le \Delta(0) \le \Delta_0. \tag{5.11}$$

To find the bounds for $t > 0$, we differentiate (5.9)

$$\begin{aligned} \Delta' &= 2\eta^T \eta' = 2\eta^T (U\epsilon)' \\ &= 2(U\epsilon)^T (U\epsilon' + U'\epsilon) \\ &= 2(U\epsilon)^T (U(F(t, u+\epsilon) - u') + U'\epsilon). \end{aligned}$$

There are two cases.

Case I. **Bound growth expected.** We let Δ_1' be the maximum value of the optimization problem

$$\begin{aligned} &\max_{\epsilon,t} \ 2(U(t)\epsilon)^T (U(t)(F(t, u(t)+\epsilon) - u(t)') + U(t)'\epsilon) \\ &\text{s.t.} \quad \Delta_0 \le \ \|U(t)\epsilon\|^2 \ \le \Delta_+, \quad t \in [\underline{t}, \overline{t}]. \end{aligned} \tag{5.12}$$

Depending upon the value of Δ_1', we compute the bounds for solution of each component. If $\Delta_1' > 0$, we take $\Delta_+' := \Delta_1'$. Then the definition of Δ_+' implies that

$$\Delta'(t) \le \Delta_+' \text{ if } \Delta(t) \in [\Delta_0, \Delta_+], \ \ t \in [\underline{t}, \overline{t}] \tag{5.13}$$

and we get from Theorem 4.1.3

$$\Delta(t) \le \overline{\Delta}(t) := \Delta_0 + \Delta'(t - \underline{t}) \ \text{ for } t \in [\underline{t}, t_1], \tag{5.14}$$

where

$$t_1 = \overline{t}, \quad \Delta' = 0, \quad \text{if } \Delta_+' \le 0 \tag{5.15}$$

and

$$t_1 = \underline{t} + (\Delta_+ - \Delta_0)/\Delta_+', \quad \Delta' = \Delta_+', \quad \text{if } \Delta_+' > 0. \tag{5.16}$$

Case II. **Decaying bound expected.** If we hope for a decaying bound we solve instead the optimization problem

$$\begin{aligned} &\max_{\epsilon,t} \ 2(U(t)\epsilon)^T (U(t)(F(t, u(t)+\epsilon) - u(t)') + U(t)'\epsilon) \\ &\text{s.t.} \quad \Delta_- \le \ \|U(t)\epsilon\|^2 \ \le \Delta_0, \quad t \in [\underline{t}, \overline{t}]. \end{aligned} \tag{5.17}$$

Now let Δ_2' be the maximum value of (5.17). Then the definition of Δ_2' implies that

$$\Delta'(t) \leq \Delta_2' \text{ if } \Delta(t) \in [\Delta_-, \Delta_0], \ \ t \in [\underline{t}, \overline{t}]. \tag{5.18}$$

If $\Delta_2' < 0$ then we take $\Delta' := \Delta_2'$, and from Theorem 4.1.4 we get the bounds of the solution

$$\Delta(t) \leq \overline{\Delta}(t) := \Delta_0 + \Delta'(t - \underline{t}) \quad \text{for } t \in [\underline{t}, t_1], \tag{5.19}$$

where

$$t_1 = \underline{t} + (\Delta_- - \Delta_0)/\Delta_-', \quad \Delta' = \Delta_-', \quad \text{if } \Delta_-' < 0. \tag{5.20}$$

If $\Delta_2' \geq 0$, we do not get a useful information and must proceed by Case I.

In both cases, if we succeed, then (5.14) and (5.19) imply that

$$\Delta(t) \ \ \leq \ \ \overline{\Delta}(t) \quad \text{for } t \in [\underline{t}, t_1]. \tag{5.21}$$

Now from (5.2) we see that

$$\epsilon(t) \ \ = \ \ U(t)^{-1}\eta(t),$$

and for any vector c with $V(t) := U(t)^{-T}$, we have

$$|c^T\epsilon(t)| \ \ = \ \ |c^T V(t)^T \eta(t)| = |(V(t)c)^T\eta(t)| \leq \|V(t)c\|\|\eta(t)\|,$$

by the Cauchy-Schwarz inequality and from (5.9) and (5.21) it follows that

$$|c^T\epsilon(t)| \ \ \leq \ \ \|V(t)c\|\Delta(t)^{1/2} \leq \|V(t)c\|\overline{\Delta}(t)^{1/2}. \tag{5.22}$$

In particular, for a unit vector $c = e_k$, we have

$$|\epsilon_k(t)| \ \ \leq \ \ \|V(t)e_k\|\overline{\Delta}(t)^{1/2} \ \text{ for } \ t \in [\underline{t}, t_1]. \tag{5.23}$$

This inequality gives computable bounds for the component wise error of the solution.

5.2 Algorithms

In this section, we will present a list of algorithms implemented by (DIVIS) to compute defect estimates for a polynomial piece $[\underline{t}, \overline{t}]$. These algorithms describe step by step implementation of our technique.

We use the following Algorithm to solve optimization problems (5.12) or (5.17) in AMPL and compute Δ'. We start with fix initial time $\underline{t}$, and the defect $\Delta_0 := n/4$, where n is the dimension of the system of ODEs. There are two cases: when the bound growth is expected and the other is a case when bounds are decaying. In first case, we solve the optimization problem

(5.12) for $[\Delta_0, \Delta_+]$, where $\Delta_+ := q\Delta_0, \;\; q > 1$. In second case, we solve the optimization problem (5.17) for $[\Delta_-, \Delta_0]$, where $\Delta_- := \Delta_0/q, \;\; q > 1$. Both possibilities are indicated by a factor pos. We begin with first case and solve our optimization problem in AMPL by using optimization solver IPOPT. To solve this problem, three files with extensions .dat consisting of input data, .mod containing model formulation, and .sa1 that calls data and model files to execute and solve the problem by using IPOPT. Since we are doing local optimization, we solve the optimization problems with different starting points. Then we take the maximum of all solutions.

Notations

$\underline{t}$ is the fix starting time, Δ_0 is initial estimate of defect, q is the factor to compute Δ_+ for optimization problem (5.12) and Δ_- for optimization problem (5.17), a, b, c, d, are coefficients of cubic Hermite polynomial for the system of ODEs and A, B, C, D, are coefficients of cubic Hermite polynomial for the preconditioner, pos is a decision variable, problem_name is the name of the problem to be solved and U is the continuously differentiable solution of the preconditioner computed at $\underline{t}$, Δ' is the maximum of all solutions of optimization problem, Δ_l is computed defect estimate, t_l is the corresponding time, $i_{\max}$ is the maximum number of attempts made to reach 20% of the time span $[\underline{t}, \overline{t}]$, $l_{\max}$ is the maximum number of attempts to reach $\overline{t}$, $i \in J$.

Algorithm 1 Computing Δ' by solving optimization problem (5.12) or (5.17)

Require: $\underline{t}$, $\overline{t}$, Δ_0, q, a, b, c, d, A, B, C, D, U, pos, problem_name, option %option for AMPL solver, deq %system of ODEs, n %dimension of the system of ODEs, N $\%n + n^2$, M %total number of starting points to solve the optimization problems, file_path %directory path where the AMPL files are printed.

Ensure: $q > 1$, $U = U(\underline{t})$ for $\underline{t} = t_0$, $\Delta_0 = n/4$.

if pos **then**

%solving the optimization problem (5.12).

$\Delta_+ = q\Delta_0$, $\Delta_{\text{in}} = [\Delta_0, \Delta_+]$.

$\eta_+ := \sqrt{\Delta_0} < \eta_1 < \cdots < \eta_{\frac{M}{2}} = \sqrt{\Delta_+}$, $\eta_- := -\eta_+$, $\eta = [\eta_-, \eta_+]$.

else

%solving the optimization problem (5.17).

$\Delta_- = \Delta_0/q$, $\Delta_{\text{in}} = [\Delta_-, \Delta_0]$.

$\eta_+ := \sqrt{\Delta_-} < \eta_1 < \cdots < \eta_{\frac{M}{2}} := \sqrt{\Delta_0}$, $\eta_- := -\eta_+$, $\eta = [\eta_-, \eta_+]$.

end if

while $1 \le i \le M$ **do**

$\varepsilon_i = U \backslash \eta_i . e^{(1)}$, tspan=$[\underline{t}, \overline{t}]$.

coeff_ode $= [a, b, c, d]$, coeff_precond $= [A, B, C, D]$.

coeff=[coeff_ode; coeff_precond].

print_ampl_data(problem_name,ε_i, n, coeff, tspan, $\mathbf{N}$, Δ_{in}, file_path).

%printing AMPL data file. See example.dat in Appendix A.

print_ampl_model(problem_name,deq,$\mathbf{N}$,file_path).

%printing AMPL model file. See example.mod in Appendix A.

print_ampl_runfile(problem_name,option,file_path).

%printing AMPL run file. See ode.sa1 in Appendix A.

ampl ode.sa1, %solving optimization problem (5.12) or (5.17) in AMPL by executing ode.sa1.

$\Delta'_i =$ ampl_output.

end while

$\Delta' = \max(\Delta'_i)$.

return Δ'.

Depending upon the sign of Δ', we use conditional differential inequality to compute defect estimates by applying following Algorithm.

Algorithm 2 `Compute` $t_1, \overline{\Delta}(t_1)$

Require: $\underline{t}$, $\overline{t}$, Δ_0, $\Delta_{\pm}$, $\Delta'_{\pm}$.
  if $\Delta'_- < 0$ **then**
    $t_1 = \min(\overline{t}, \underline{t} + (\Delta_- - \Delta_0)/\Delta'_-)$.
    $\Delta' = \Delta'_-$.
  else
    if $\Delta'_+ > 0$ **then**
      $t_1 = \min(\overline{t}, \underline{t} + (\Delta_+ - \Delta_0)/\Delta'_+)$.
      $\Delta' = \Delta'_+$.
    else
      $t_1 = \overline{t}$.
      $\Delta' = 0$.
    end if
  end if
  $\overline{\Delta}(t_1) = \Delta_0 + \Delta'(t_1 - \underline{t})$.
  return $t_1, \overline{\Delta}(t_1)$.

If t_1 computed by Algorithm 2 doesn't reach $\overline{t}$, then there are two possibilities, either $t_1 - t_0 \geq 20\%$ of time span $[\underline{t}, \overline{t}]$ or not. If yes, then apply Algorithm 3 to compute t_l and $\overline{\Delta_l}$ (defect estimates) otherwise use Algorithm 4 for that purpose.

Algorithm 3 `Computing` t_l `and defect estimate (`$\overline{\Delta_l}$`) for` $t_1 \geq (\overline{t} - \underline{t})/5$

Require: $\underline{t}$, $\overline{t}$, Δ_0, $\Delta_{\pm}$, $\Delta'_{\pm}$.
  `Compute` t_1`,` $\overline{\Delta}(t_1)$ `by applying Algorithm 2.`
  if $t_1 \geq (\overline{t} - \underline{t})/5$ **then**
    $t_l = t_1$`,` $\overline{\Delta_l} = \overline{\Delta}(t_1)$.
    return $t_l, \overline{\Delta_l}$.
  end if

Algorithm 4 Computing t_l and defect estimate ($\overline{\Delta}_l$) for $t_1 < (\overline{t} - \underline{t})/5$ and $\Delta' > 0$

Require: $\underline{t}, t_1, \overline{t}, \Delta_0, \Delta', q, i_{\max}, a, b, c, d, A, B, \mathbf{C}, D, U$,pos, problem_name, deq %system of ODEs, option %option for AMPL solver, n %dimension of the system of ODEs, N %$n + n^2$, file_path %directory path where the AMPL files are printed.

Ensure: $\underline{t} < t_1 < (\overline{t} - \underline{t})/5$, $q > 1$.

if $t_1 < (\overline{t} - \underline{t})/5$ and $\Delta' > 0$ **then**
  while $i \geq 2$ **do**
    $\Delta_+ = i\Delta_0 q$.
    With new value of Δ_+, apply Algorithm 1 with pos = true to compute new Δ'.
    if $\Delta' \leq 0$ **then**
      $t_l = \overline{t}$, $\overline{\Delta}_l = \Delta_0$.
      return t_l, $\overline{\Delta}_l$
    else
      apply Algorithm 2 to compute t_1, $\overline{\Delta}(t_1)$.
      Now check both cases: $t_1 \geq (\overline{t} - \underline{t})/5$ or $t_1 < (\overline{t} - \underline{t})/5$.
      if $t_1 \geq (\overline{t} - \underline{t})/5$ **then**
        compute t_l, $\overline{\Delta}_l$ by applying Algorithm 3.
      else
        $i = i + 1$ %increase the value $\Delta_+ = i\Delta_0 q$ and apply %Algorithm 1 with pos = true.
        Continue until $i = i_{\max}$.
        if $i = i_{\max}$ and $t_1 < (\overline{t} - \underline{t})/5$ **then**
          $t_l = \phi$, $\overline{\Delta}_l = \phi$.
          stop.
        end if
      end if
    end if
  end while
  return t_l, $\overline{\Delta}_l$.
end if

The following algorithm is used to compute t_l and $\overline{\Delta_l}$ (defect estimates) for the case $(t_1 - t_0) < (\overline{t} - \underline{t})/5$ and $\Delta' < 0$.

Algorithm 5 Computing t_l and defect estimates ($\overline{\Delta_l}$) for $t_1 < (\overline{t} - \underline{t})/5$ and $\Delta' < 0$

Require: $\underline{t}, t_1, \overline{t}, \Delta_0, \Delta', q, i_{\max}, a, b, c,, d, A, B, \mathbf{C}, D, U$,pos, problem_name,
deq %system of ODEs, option %option for AMPL solver,
n %dimension of the system of ODEs, N %$n + n^2$,
file_path %directory path where the AMPL files are printed.

Ensure: $\underline{t} < t_1 < (\overline{t} - \underline{t})/5$, $q > 1$.

if $t_1 < (\overline{t} - \underline{t})/5$ and $\Delta' < 0$ **then**
  while $i \geq 2$ **do**
    $\Delta_- = \Delta_0/iq$ %decrease the value of Δ_-.
    Now with new value of Δ_-, apply Algorithm 1
    pos = false and compute new Δ'.
    if $\Delta' \geq 0$ **then**
      $t_l = \overline{t}$, $\overline{\Delta_l} = \Delta_0$.
      return t_l, $\overline{\Delta_l}$
    else
      apply Algorithm 2 to compute t_1, $\overline{\Delta}(t_1)$.
      Check both cases: $t_1 \geq (\overline{t} - \underline{t})/5$ or $t_1 < (\overline{t} - \underline{t})/5$.
      if $t_1 \geq (\overline{t} - \underline{t})/5$ **then**
        compute $t_l, \overline{\Delta_l}$ by applying Algorithm 3.
      else
        $i = i + 1$ %decrease the value $\Delta_- = \Delta_0/q$ and again apply
        %Algorithm 1 with pos = false.
        Continue until $i = i_{\max}$.
        if $i = i_{\max}$ and $t_1 < (\overline{t} - \underline{t})/5$ **then**
          $t_l = \phi$, $\overline{\Delta_l} = \phi$.
          stop.
        end if
      end if
    end if
  end while
  return t_l, $\overline{\Delta_l}$.
end if

Following algorithm is applied to compute t_l and $\overline{\Delta_l}$ (defect estimates) when $\Delta' > 0$

Algorithm 6 `Computing` t_l `and defect estimate (`$\overline{\Delta_l}$`) for` $\Delta' > 0$

Require: $\underline{t}$, $\overline{t}$, Δ_0, Δ_+, Δ', q, a, b, c, d, A, B, $\mathbf{C}$, D, U,`pos, problem_name,`
deq %system of ODEs, option %option for AMPL solver,
n %`dimension of the system of ODEs,` N %$n + n^2$,
`file_path` %`directory path where the AMPL files are printed.`
Ensure: $t_{l-1} = \underline{t}$, $\overline{\Delta}_{l-1} = \Delta_0$, $q > 1$.
if $\Delta' > 0$ **then**
`compute` t_1, $\overline{\Delta}(t_1)$ `by applying Algorithm 2.`
`We can have two cases:` $t_1 \geq (\overline{t} - \underline{t})/5$ `or` $t_1 < (\overline{t} - \underline{t})/5$.
if $t_1 \geq (\overline{t} - \underline{t})/5$ **then**
`apply Algorithm 3 to compute` t_l, $\overline{\Delta_l}$.
else
`apply Algorithm 4 to compute` t_l, $\overline{\Delta_l}$.
end if
return t_l, $\overline{\Delta_l}$.
end if

The following algorithm is used to compute t_l and $\overline{\Delta_l}$ (defect estimates) when $\Delta' < 0$

Algorithm 7 Computing t_l and defect estimates ($\overline{\Delta_l}$) for $\Delta' < 0$

Require: $\underline{t}$, $\overline{t}$, Δ_0, Δ', q, a, b, c, d, A, B, $\mathbf{C}$, D, U,pos, problem_name, deq %system of ODEs, option %option for AMPL solver, n %dimension of the system of ODEs, N $\%n + n^2$, file_path %directory path where the AMPL files are printed.
Ensure: $t_{l-1} = \underline{t}$, $\overline{\Delta}_{l-1} = \Delta_0$, $q > 1$.
  if $\Delta' < 0$ **then**
    apply Algorithm 1 with pos = false and compute new Δ'.
    if $\Delta' \geq 0$ **then**
      $t_l = \overline{t}$, $\overline{\Delta_l} = \Delta_0$.
      return t_l, $\overline{\Delta_l}$.
    else
      apply Algorithm 2 to compute t_1, $\overline{\Delta}(t_1)$.
      We can have two cases: $t_1 \geq (\overline{t} - \underline{t})/5$ or $t_1 < (\overline{t} - \underline{t})/5$.
      if $t_1 \geq (\overline{t} - \underline{t})/5$ **then**
        apply Algorithm 3 to compute t_l, $\overline{\Delta_l}$.
      else
        %now $t_1 < (\overline{t} - \underline{t})/5$.
        apply Algorithm 5 to compute t_l, $\overline{\Delta_l}$.
      end if
    end if
    return t_l, $\overline{\Delta_l}.s$
  end if

We apply following algorithm to compute coefficients of cubic Hermite polynomials for the systems of ODEs and the preconditioner by using the continuity of function and its first derivative.

Algorithm 8 Computing coefficients a, b, c, d, A, B, C, D

Require: $\underline{t}$, $\overline{t}$, y %approximate solution of system of ODEs,
$\dot{y}$ %system of ODEs evaluated at $t \in [\underline{t}, \overline{t}]$ and y,
Y %approximate solution of system of the preconditioner,
$\dot{Y}$ %system of the preconditioner evaluated at $t \in [\underline{t}, \overline{t}]$ and Y,
h %step length
Ensure: Function is continuously differentiable.
%Computing coefficients a, b, c, d of cubic Hermite polynomial
%for the system of ODEs
$a = \Big(\dot{y}(\overline{t}) + \dot{y}(\underline{t}) - 2(y(\overline{t}) - y(\underline{t}))/h\Big)/h^2,$
$b = \Big((y(\overline{t}) - y(\underline{t}))/h - \dot{y}(\underline{t})\Big)/h,$
$c = \dot{y}(\underline{t}),$
$d = y(\underline{t}).$
%Computing coefficients A, B, C, D of cubic Hermite
%polynomial for the system of the preconditioner
$A = \Big(\dot{Y}(\overline{t}) + \dot{Y}(\underline{t}) - 2(Y(\overline{t}) - Y(\underline{t}))/h\Big)/h^2,$
$B = \Big((Y(\overline{t}) - Y(\underline{t}))/h - \dot{Y}(\underline{t})\Big)/h,$
$C = \dot{Y}(\underline{t}),$
$D = Y(\underline{t}).$
return $a, b, c, d, A, B, C, D.$

We compute piecewise continuously differentiable solutions u and U by using Hermite interpolation for the interval $[t_1, \overline{t}]$ in order to compute coefficients cubic Hermite polynomials for the systems of ODEs and the preconditioner for knots.

Algorithm 9 Computing solutions $u(t)$ and $U(t)$ and their first derivatives

Require: $\underline{t}$, t_1, $\overline{t}$, a, b, c, d, A, B, C, D.
Ensure: $t = t_1$.
if $t_1 < \overline{t}$ **then**
%Computing cubic Hermite polynomial for the system of ODEs
% and the preconditioner.
$u(t) = a(t - \underline{t})^2(t - \overline{t}) + b(t - \underline{t})^2 + c(t - \underline{t}) + d,$
$U(t) = A(t - \underline{t})^2(t - \overline{t}) + B(t - \underline{t})^2 + C(t - \underline{t}) + D,$
%and their first derivative
$\dot{u}(t) = 2a(t - \underline{t})(t - \overline{t}) + a(t - \underline{t})^2 + 2b(t - \underline{t}) + c,$
$\dot{U}(t) = 2A(t - \underline{t})(t - \overline{t}) + A(t - \underline{t})^2 + 2B(t - \underline{t}) + C.$
return $u(t), U(t), \dot{u}(t), \dot{U}(t).$
end if

Now we present an algorithm to compute lists of t_l and $\overline{\Delta_l}$(defect estimates) for $[\underline{t}, \overline{t}]$

Algorithm 10 Computing lists of t_l and $\overline{\Delta_l}$ for polynomial piece $[\underline{t}, \overline{t}]$

Require: $\underline{t}$, $\overline{t}$, $l_{\max}$, Δ_0, q, a, b, c, d, A, B,, $\mathbf{C}$, D, U, pos, problem_name, deq %system of ODEs, option %option for AMPL solver, n % dimension of the system of ODEs, N %$n+n^2$, file_path %directory path where the AMPL files are printed.
Ensure: $t_{l-1} = \underline{t}, \overline{\Delta}_{l-1} = \Delta_0, U = U(\underline{t})$.
  while $t_l < \overline{t}$ **do**
    apply Algorithm 1 with pos = true and compute Δ'.
    if $\Delta' > 0$ **then**
      compute t_l, $\overline{\Delta}_l$ by applying Algorithm 6.
      if $t_l = \phi = \overline{\Delta}_l$ **then**
        continue
      end if
    else
      if $\Delta' < 0$ **then**
        apply Algorithm 7 to compute t_l, $\overline{\Delta}_l$.
        if $t_l = \phi = \overline{\Delta}_l$ **then**
          continue
        end if
      else
        $t_l = \overline{t}$, $\overline{\Delta}_l = \Delta_0$.
        return t_l, $\overline{\Delta}_l$.
      end if
    end if
    if $t_l < \overline{t}$ **then**
      apply Algorithm 9 to evaluate $u(t), U(t), \dot{u}(t), \dot{U}(t)$ at $t = t_l$ and then using these values for $\underline{t} = t_l$, apply Algorithm 8 to compute coefficients a, b, c, d, and A, B, $\mathbf{C}$, D.
      $\underline{t} = t_l$, $\Delta_0 = \overline{\Delta}_l$.
      $l = l + 1$.
      Continue until $l = l_{\max}$.
      if $t_{l_{\max}} < \overline{t}$ **then**
        $\overline{t} = t_{l_{\max}}, \overline{\Delta}_l = \Delta_{l_{\max}}$.
        stop. % Bounds cannot be improved anymore.
      end if
    end if
  end while
  return t_l, $\overline{\Delta}_l$.

5.3 Implementation

We implemented the theory of error bounds in the following steps.

Step I. We construct an ODE model in MATLAB consisting of the systems of differential equations

$$y'(t) \;=\; F(t, y(t)), \quad t \in [0, T] \tag{5.24}$$

and the following equation for the preconditioner

$$\dot{Y}(t) \;=\; -YF_y - Y(\delta_0 + \delta_k (Y^T Y)^k), \quad t \in [0, T], \tag{5.25}$$

where δ_0 and δ_k are regularization parameters, and k is a nonnegative integer. The initial conditions for this model are chosen as

$$y_0 = \frac{\overline{y}_0 + \underline{y}_0}{2}, \quad Y_0 = \text{Diag}\,(\overline{y}_0 - \underline{y}_0)^{-1}.$$

By using an ODE solver, we compute approximate point wise solutions of the systems (5.24) and (5.25) with $y(0) = y_0 = (\overline{y}_0 + \underline{y}_0)/2$, $Y_0 = \text{Diag}\,(\overline{y}_0 - \underline{y}_0)^{-1}$. In order to check the error bounds, we also solve (5.24) for $y(0) = y_1 = \underline{y}_0$ and $y(0) = y_2 = \overline{y}_0$.

Step II. Using cubic Hermite interpolation, we find an approximate continuously differentiable solution

$$\begin{aligned} u(t) &= u_i(t) \;\text{ for }\; t_{i-1} \le t < t_i, \\ U(t) &= U_i(t) \;\text{ for }\; t_{i-1} \le t < t_i, \end{aligned}$$

where $0 = t_0 < t_1 < \cdots < t_m = T$ for the systems (5.24) and preconditioner (5.25) in the form of piecewise cubic polynomials (5.3) and (5.5) respectively. For each polynomial piece $[t_{i-1}, t_i]$ we do the Steps III-V.

Step III. We encode the optimization problems (5.12) and (5.17) in AMPL and solve them by using AMPL solver IPOPT. For this purpose, we construct the model file, the data file and the command file (Samples of each file are given in Appendix A). Then by applying Algorithm 1, we compute Δ'. Currently we are doing local optimization.

Step IV. We compute t_1, $\overline{\Delta}(t_1)$ by applying Algorithm 2. There are two possibilities: either $t_1 = \overline{t}$ or $t_1 < \overline{t}$. If $t_1 = \overline{t}$, then we take $t_l := t_1 = \overline{t}$ and $\overline{\Delta}_l := \overline{\Delta}(t_1)$ and go to Step V. Otherwise we construct lists of t_l and $\overline{\Delta}_l$ by applying Algorithm 10.

Step V. Using lists of t_l and $\overline{\Delta_l}$, computed by Step IV, we plot the component wise error in the solution by using inequality (5.23). Now we take $\Delta_0 := \overline{\Delta}_l$ and $\underline{t} := t_l$ and return to Step II for the next iteration.

Note: There is a possibility that the program cannot reach $\overline{t}$. In this situation, further attempts are made as described in Algorithm 10. But if after finite many iterations, goal is not achieved, then a stopping criteria is introduced to stop the program at that stage (see Algorithm 10) . This implies that bounds can not be improved anymore.

Since we are doing local optimization, so bounds are not rigorous. In order to get sharper bounds, we would need to do rigorous global optimization with rounding error control.

The description of our solver is given in next subsection 5.4. For further explanation, see Appendix B, in particular, MATLAB routine driver.m tells how to proceed.

5.4 Graphical representation

The graphical representation of our method is as follows:

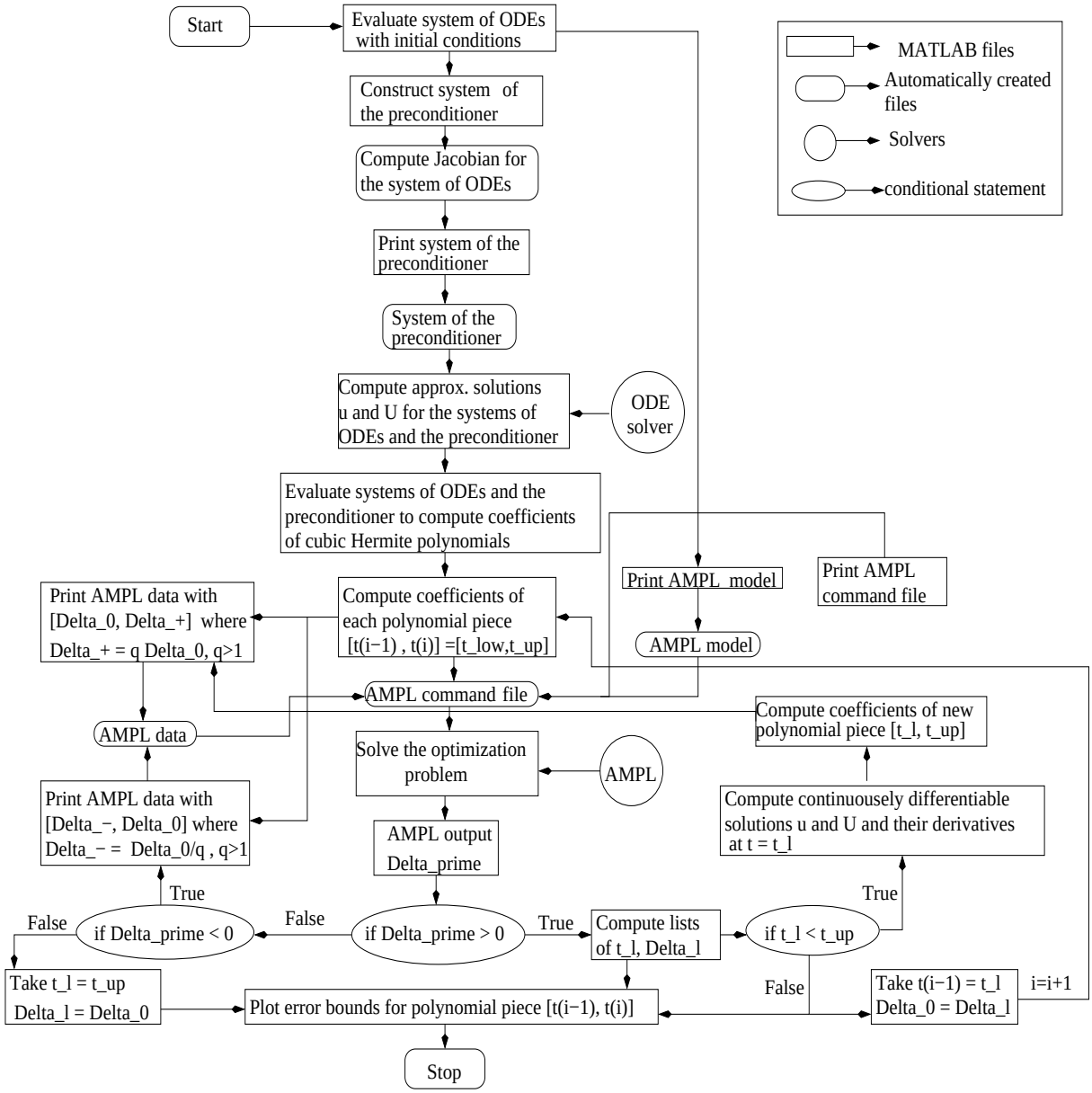

Figure 5.1: Flow diagram for enclosing the solutions

Chapter 6

Numerical results

In this chapter we apply our solver DIVIS, which implements the new methods, to compute error bounds for a number of sample ODEs. The results obtained from DIVIS are compared with the solvers VALENCIA-IVP, VNODE-LP, and VSPODE.

As we shall see, the results obtained from VALENCIA-IVP are not acceptable except for Example 5. In order to produce tight enclosures for VNODE-LP, the order of method is kept between $5 - 8$. For higher orders, VNODE-LP does not behave well and diverges at very early stages, for example, in Example 1, it diverges at $t_1 = 1.06582$ with enclosure $W = 7.844519$ for order $p = 20$. For VSPODE, the orders of Taylor model and interval Taylor series method are chosen as 17 or 18 depending upon the nature of the problem. These are observed to be better choices both for VNODE-LP and VSPODE.

The reported widths are always at t_{end} and are represented by $W_i, i = 1 \cdots n$. In cases where a solver did not reach t_{end}, we usually tried several variants for the solver settings, and reported the best results.

We have approximated the lower bound $\underline{y}$, upper bound $\overline{y}$ and mid point of all variables without rounding error control. In general, it may be significant under restriction of the range but it gives at least some impression. For more reliable inner approximation, one would have to use Monte-Carlo approach.

Blue dotted lines $\cdots$ show three approximated solutions of the system of ODEs with starting points at the lower bound, the midpoint, and the upper bound of the initial box. Magenta —show the validated state enclosures computed by DIVIS, red $--$ show the enclosures computed by VSPODE. The enclosures computed by VALENCIA-IVP are shown in black $--$ and those obtained form VNODE-LP by $-.$.

6.1 Examples

6.1.1 Example 1

$$y' = -y^3,$$

$$t \in [0, 100], \quad y(t_0) = y_0 \in [0, 1].$$

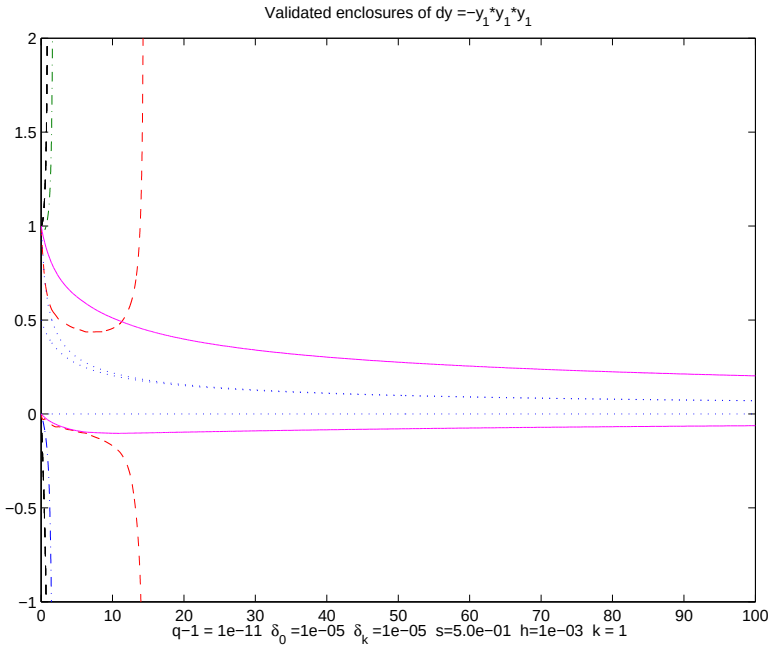

Figure 6.1: Example 1

VALENCIA-IVP, VNODE-LP and VSPODE do not reach t_{end}. VALENCIA-IVP diverges at $t_1 = .998$ and the width of enclosure computed at t_1 is $W = 1.40e+03$. VNODE-LP stops at $t_1 = 1.683$ and the width of enclosure computed at t_1 is $W = 6.84$ with order of method $p = 8$. In case of VSPODE, integration fails at $t_1 = 14.3879$ with $W = 28.83$, with order of Taylor model 18 and order of interval Taylor series method 17.

But DIVIS reaches t_{end} as shown in Table 6.1.

6.1.2 Example 2

$$y' = 1 - y^2,$$

Method	W
VALENCIA-IVP	–
VNODE-LP	–
VSPODE	–
DIVIS	2.65×10^{-1}
y_{approx}	7.05×10^{-2}

Table 6.1: Results for Example 1

$$t \in [0, 4], \quad y(t_0) = y_0 \in [3.9, 4.1].$$

The exact solution is $y(t) = (2y_0 + 2)/(y_0 - (y_0 - 1)/\exp(2t) + 1) - 1$.

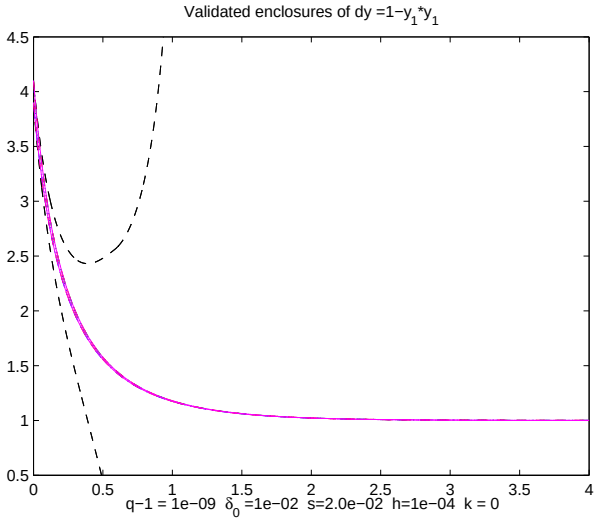

Figure 6.2: Example 2

$\underline{y}(t_{\text{end}}) = 1.00039715$ and $\overline{y}(t_{\text{end}}) = 1.00040790$.

VALENCIA-IVP diverges at $t_1 = 1.17540$ and the width W of validated enclosure computed at t_1 is 1.012704×10^3.

Though VNODE-LP reaches t_{end} with order $p = 5$, but with $y(t_{\text{end}}) \in [1.0004000, 1.0004100]$, which shows that the computed enclosure does not enclose the solution with initial value $y_0 = 3.9$ at t_{end}, Thus the enclosure is erroneous, and there is a bug in the program. Changes in order makes no difference in resulting enclosures.

VSPODE reaches t_{end} with order of Taylor model as 18 and order of interval Taylor series method as 17. DIVIS also reaches t_{end}; the computed enclosures are slightly sharper than those by VSPODE.

Method	W
VALENCIA-IVP	–
VNODE-LP	1.000000×10^{-5}
VSPODE	1.099999×10^{-5}
DIVIS	1.097645×10^{-5}
y_{approx}	1.074276×10^{-5}

Table 6.2: Results for Example 2

6.1.3 Example 3

$$y' = -y,$$

$$t \in [0, 5], \quad y(t_0) = y_0 \in [0, 1].$$

The exact solution is $y(t) = y_0 / \exp(-t)$

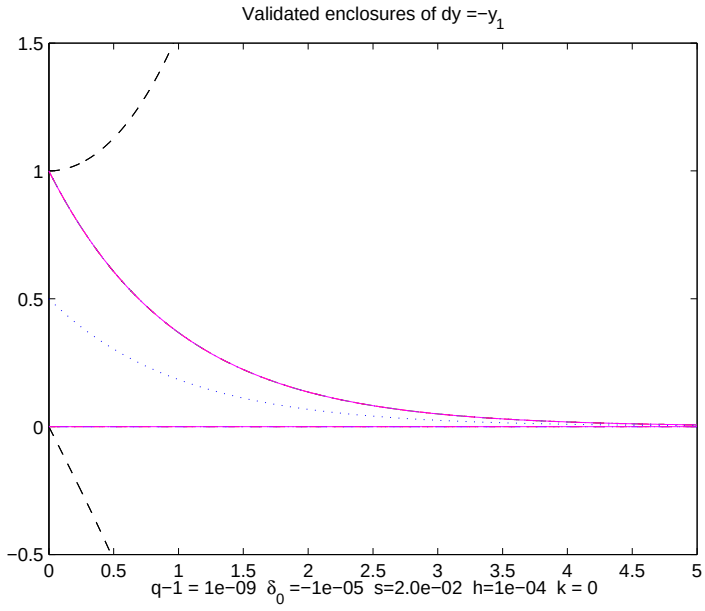

Figure 6.3: Example 3

In this case, a validated state enclosures was computed by VALENCIA-IVP, but the bounds are far too wide where as VNODE-LP with order $p = 5$, VSPODE with orders of Taylor model and interval Taylor series method as 18 and 17 respectively and DIVIS produce essentially the same, almost tight results with 4 decimal places to enclose all possible solutions.

Method	W
VALENCIA-IVP	1.484927×10^{2}
VNODE-LP	6.737950×10^{-3}
VSPODE	6.738000×10^{-3}
DIVIS	6.7379711×10^{-3}
y_{approx}	6.7379474×10^{-3}

Table 6.3: Results for Example 3

6.1.4 Example 4

$$\begin{aligned} y_1' &= -y_2, \\ y_2' &= y_1, \end{aligned}$$

$$t \in [0, 100], \quad y(t_0) \in [0, 1] \times [-1, 0].$$

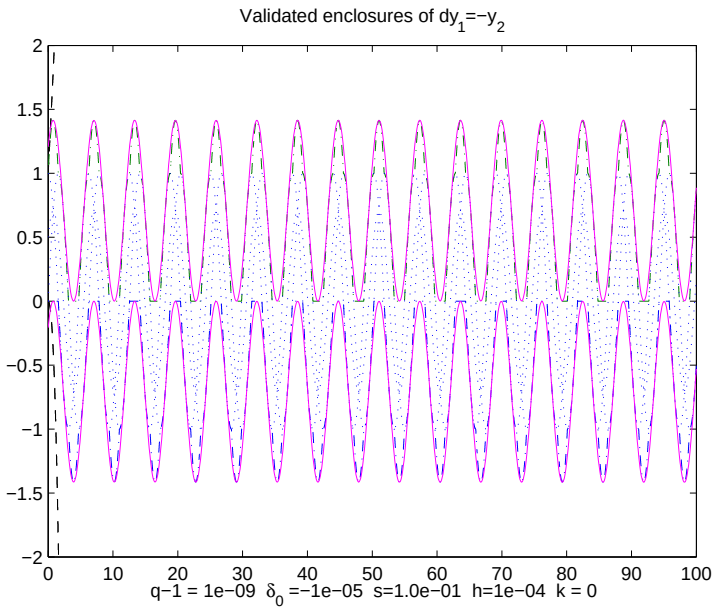

Figure 6.4: First component of Example 4

VALENCIA-IVP stops at $t_1 = 6.907$ and the width of validated enclosures computed, $W_1 = 1.0000 \times 10^3$, $W_2 = 1.0000 \times 10^3$ is far too wide.

The results obtained from VNODE-LP for order $p = 5$, VSPODE for orders of Taylor model and interval Taylor series method as 18 and 17, respectively and from DIVIS are given in Table 6.4.

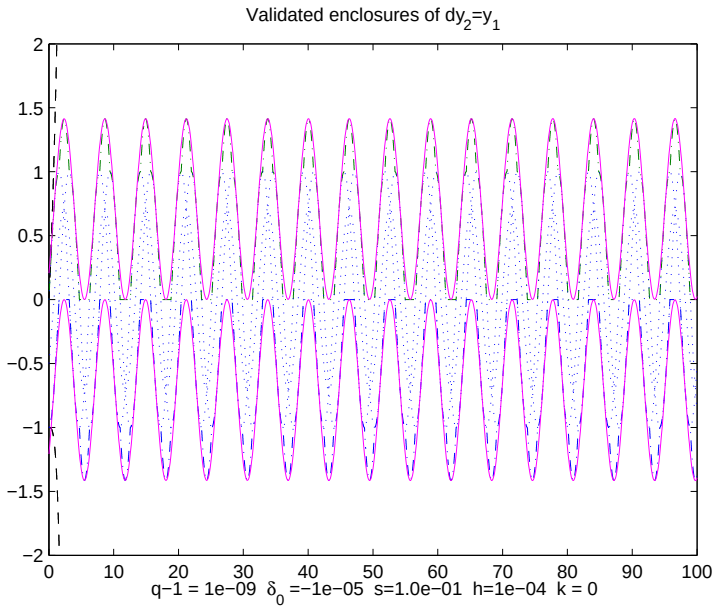

Figure 6.5: Second component of Example 4

Since the ellipsoid enclosing the initial box has bigger volume due to outer ellipsoidal approximation, the enclosures computed by DIVIS are wider than those of VNODE-LP and VSPODE. But no additional wrapping is observed.

Method	W_1	W_2
VALENCIA-IVP	–	–
VNODE-LP	1.3686850	1.3686800
VSPODE	1.3686850	1.3686850
DIVIS	1.4179891	1.4179746
y_{approx}	1.3686829	3.5595318×10^{-1}

Table 6.4: Results for Example 4

6.1.5 Example 5

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= y_1, \end{aligned}$$

$$t \in [0, 5], \quad y(t_0) \in [-.1, 0.1] \times [0.9, 1.1].$$

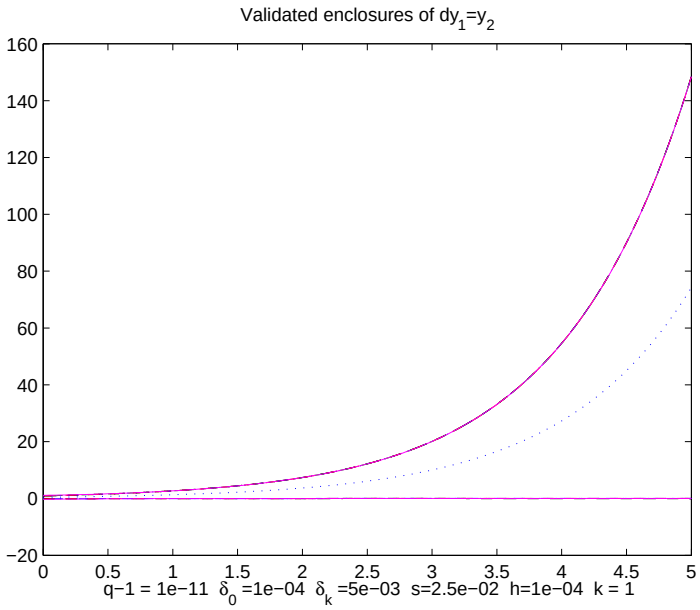

Figure 6.6: First component of Example 5

It can be observed from Table 6.5 that the validated enclosures computed by VALENCIA-IVP, VNODE-LP, VSPODE and DIVIS are essentially the same. The order of method $p = 5$ for VNODE-LP and order of Taylor model and interval Taylor series method for VSPODE are 18 and 17 respectively.

Method	W_1	W_2
VALENCIA-IVP	1.485×10^2	1.485×10^2
VNODE-LP	1.484×10^2	1.484×10^2
VSPODE	1.484×10^2	1.484×10^2
DIVIS	1.484×10^2	1.484×10^2
y_{approx}	1.484×10^2	1.484×10^2

Table 6.5: Results of Example 5

6.1.6 Example 6

$$\begin{aligned} y_1' &= y_1 - 2y_2, \\ y_2' &= 3y_1 - 4y_2, \end{aligned}$$

$$t \in [0, 5], \quad y(t_0) \in [0, 1] \times [-1, 0].$$

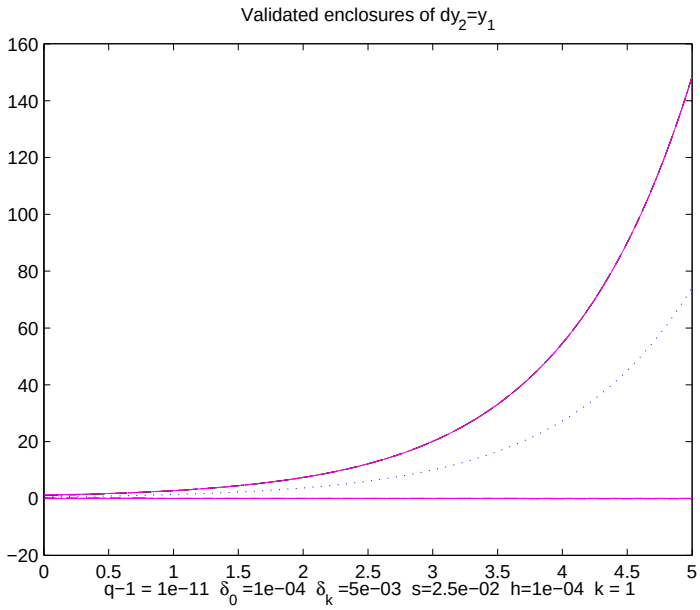

Figure 6.7: Second component of Example 5

VALENCIA-IVP stops at $t_1 = 1.2388$ with wide enclosures $W_1 = 4.5945 \times 10^2$, $W_2 = 9.9963 \times 10^2$.

The enclosures computed by VNODE-LP and VSPODE at t_{end} are almost same upto 4 decimal places and better than those of DIVIS, again due to the unavoidable wrapping by the initial ellipsoid. The order of method $p = 5$ for VNODE-LP and order of Taylor model and interval Taylor series method for VSPODE are 18 and 17 respectively.

Method	W_1	W_2
VALENCIA-IVP	–	–
VNODE-LP	3.3508×10^{-2}	3.3417×10^{-2}
VSPODE	3.3508×10^{-2}	3.3417×10^{-2}
DIVIS	3.4181×10^{-2}	3.4092×10^{-2}
y_{approx}	6.7379×10^{-3}	6.7379×10^{-3}

Table 6.6: Results of Example 6

6.1.7 Example 7

$$\begin{aligned} y_1' &= y_1^2 y_2^2 - y_1^4 - y_1 y_2^2 + y_1^3 - y_2^4, \\ y_2' &= y_1 y_2^3 - y_1^3 y_2 - y_2^3 + y_1^2 y_2, \end{aligned}$$

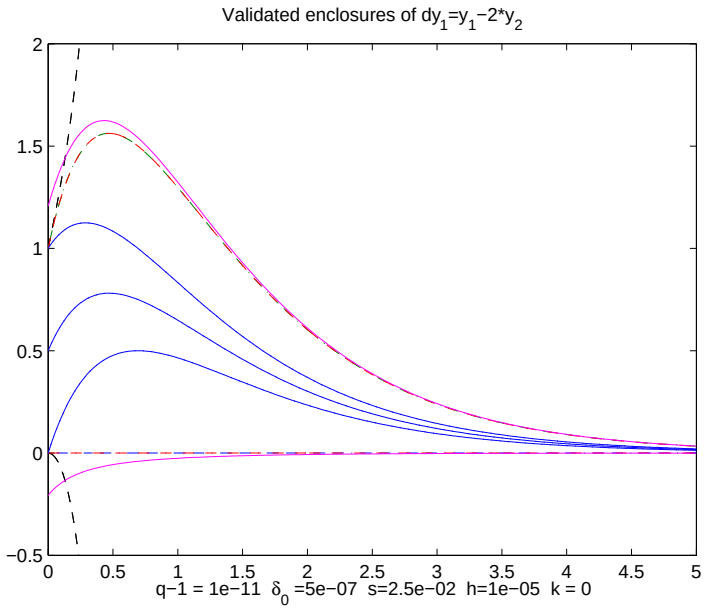

Figure 6.8: First component of Example 6

$$t \in [0, .56], \quad y(t_0) \in [0.2, 1] \times [0.6, 1].$$

VALENCIA-IVP computes enclosures until $t_1 = .1608$ with wide bounds $W_1 = 1.6349 \times 10^1$, $W_2 = 1.1209 \times 10^1$. a VNODE-LP (with order $p = 5$) even reaches only up to $t_1 = .1558$ with wide bounds $W_1 = 5.1006$, $W_2 = 3.6407$. VSPODE reaches t_{end} but the enclosures are again wide. The order of Taylor model and interval Taylor series method are 18 and 17 respectively.

DIVIS computes much tighter enclosures at t_{end}.

Method	W_1	W_2
VALENCIA-IVP	–	–
VNODE-LP	–	–
VSPODE	4.1367	2.4300
DIVIS	1.4949	5.2537×10^{-1}
y_{approx}	3.1500×10^{-1}	3.9532×10^{-1}

Table 6.7: Results of Example 7

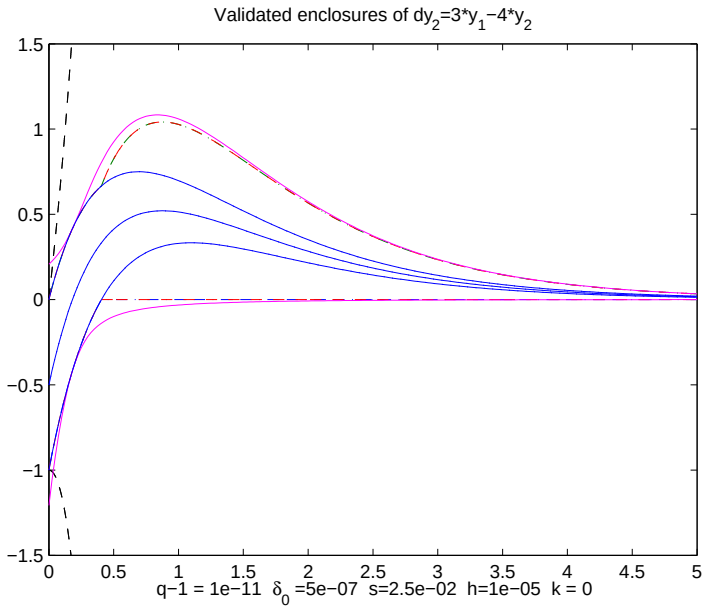

Figure 6.9: Second component of Example 6

6.1.8 Example 8

$$\begin{aligned} y_1{}' &= 1 + y_1^2 y_2 - (y_3 + 1) y_1, \\ y_2{}' &= y_1 y_3 - y_1^2 y_2, \\ y_3{}' &= -y_1 y_3 + 1, \end{aligned}$$

$$t \in [0, .24], \quad y(t_0) \in [0, 1] \times [0, 2] \times [0, 1].$$

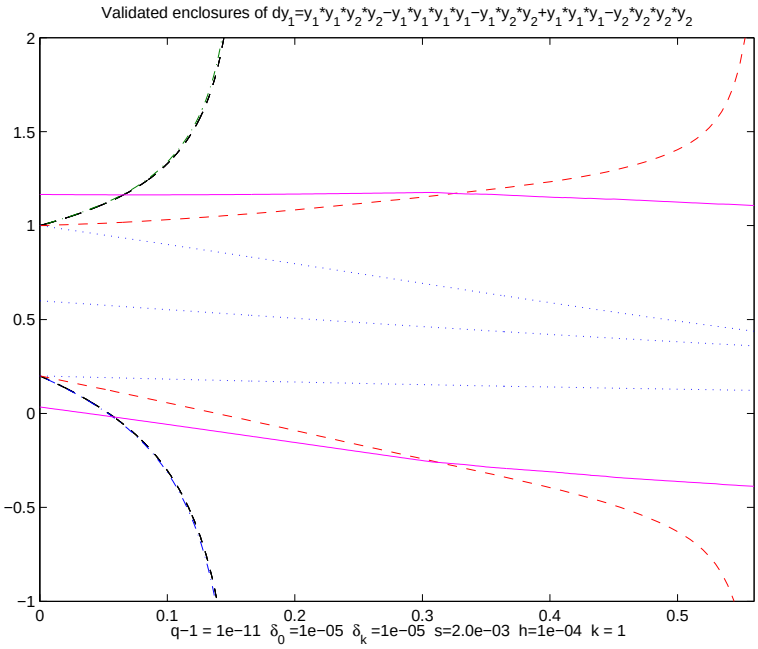

Figure 6.10: First component of Example 7

In first and second components, enclosures computed by VALENCIA-IVP are worst amongst all but in third component, enclosure is better than that of DIVIS. VNODE-LP computes validated enclosures for the best order $p = 5$. VSPODE, both the order of Taylor model and that of order of interval Taylor series method is 17 resulting into better enclosures as compared to VALENCIA-IVP, VNODE-LP and DIVIS. DIVIS again suffers from the initial wrapping step.

Method	W_1	W_2	W_3
VALENCIA-IVP	4.852	5.358	1.585
VNODE-LP	4.841	5.347	1.577
VSPODE	1.839	2.877	1.169
DIVIS	2.531	5.147	2.476
y_{approx}	1.059	1.695	7.357

Table 6.8: Results of Example 8

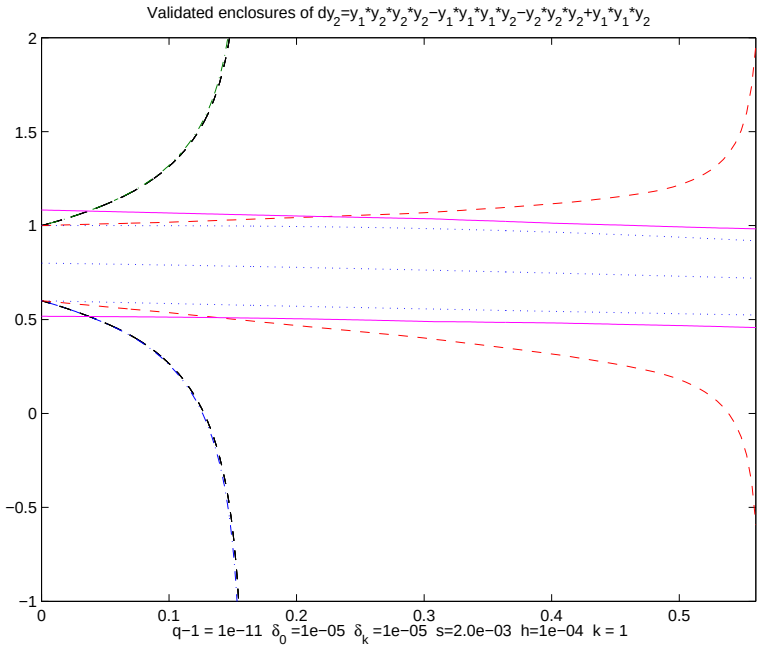

Figure 6.11: Second component of Example 7

6.1.9 Example 9

$$\begin{aligned} y_1{}' &= -2y_2 + y_2y_3 - y_1^3, \\ y_2{}' &= y_1 - y_1y_3 - y_2^3, \\ y_3{}' &= y_1y_2 - y_3^3, \end{aligned}$$

$$t \in [0, .549858], \quad y(t_0) \in [0, 1.5] \times [0, 1.2] \times [0, 1].$$

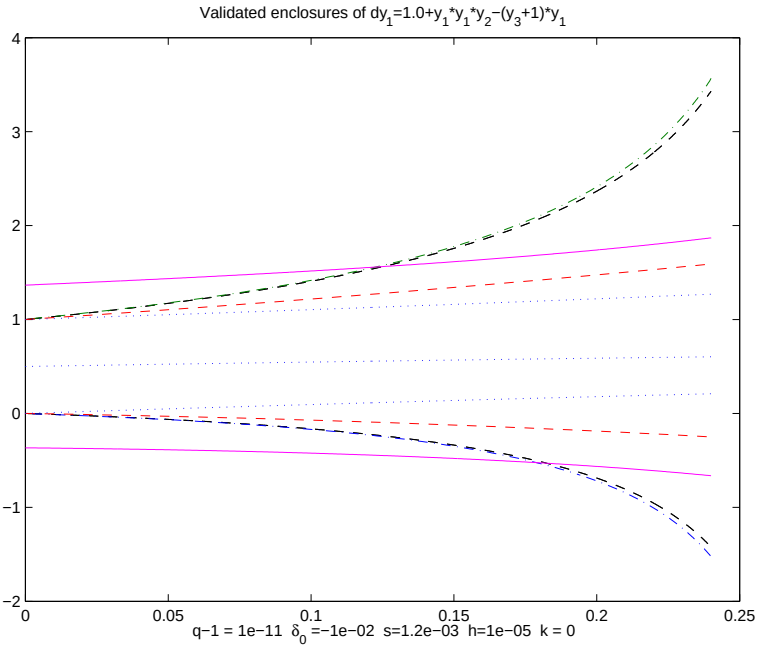

Figure 6.12: First component of Example 8

VALENCIA-IVP computes enclosures until $t_1 = .2576$ with wide bounds $W_1 = 4.0314e + 04$, $W_2 = 8.5279$, $W_3 = 5.7964$.

VNODE-LP computes enclosures until $t_1 = .2520$ with better bounds $W_1 = 7.9141$, $W_2 = 4.7274$, $W_3 = 3.3217$ with order of method $p = 5$, but nevertheless couldn't continue further.

The enclosures computed by VSPODE with order of Taylor model = 17 and order of interval Taylor series method = 17 diverges after t_{end} (which is why we chose this t_{end}).

DIVIS reached t_{end} and can be continued beyond this value of t.

6.2 Conclusion

It can be observed from the above presented examples that DIVIS computes the best enclosures among all methods tested for ODEs or systems of ODEs containing higher order polynomials. We only need to compute the Jacobian of the system of ODEs to construct the preconditioner (5.25), whereas the other solvers need high derivatives and the resulting Taylor series are not so

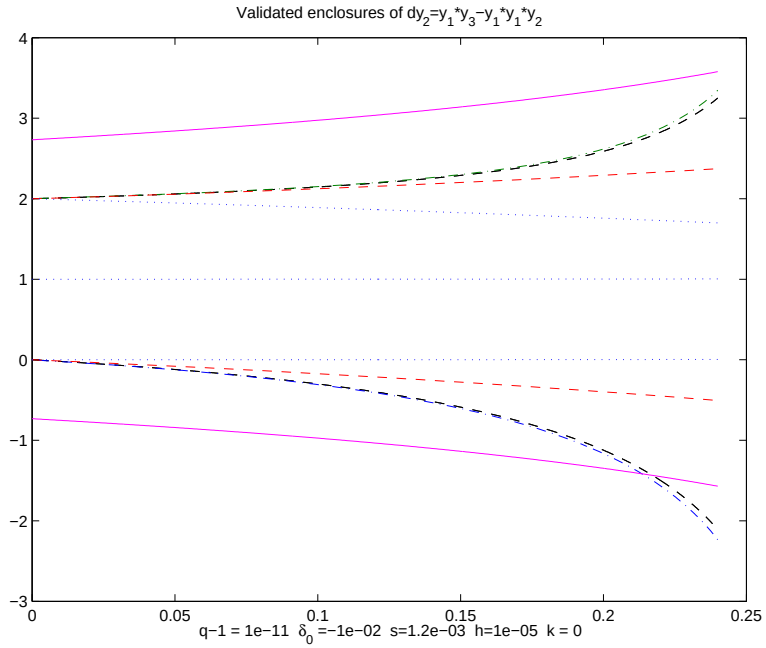

Figure 6.13: Second component of Example 8

Method	W_1	W_2	W_3
VALENCIA-IVP	–	–	–
VNODE-LP	–	–	–
VSPODE	7.3526	3.0457	3.2130
DIVIS	3.2175	1.7181	1.9901
y_{approx}	5.1424×10^{-1}	7.4800×10^{-1}	9.1389×10^{-1}

Table 6.9: Results of Example 9

well-behaved. Since our defect estimates are computed by using local optimization, the execution time is larger, though. Since we are using outer ellipsoidal approximation, our initial enclosing set is already an ellipsoid enclosing the initial box, which leads to an unavoidable increase in the volume of the enclosures. As a result, our bound are slightly worse than those of the other solvers when the latter are highly accurate, such as in Examples 4 and 8. (Things would be opposite when the initial uncertainty is ellipsoidal; in this case the other methods would have this disadvantage.)

We also note that our results are not fully rigorous since we use a heuristic global optimization solver (multiple local search) only. The rigorous results would usually be only slightly worse than these heuristic results because for the wide initial intervals used in the experiments, the additional effect of the rounding errors is minor. However, in a few cases it might be that only a local optimum was found, in which case the true bounds could be signifi-

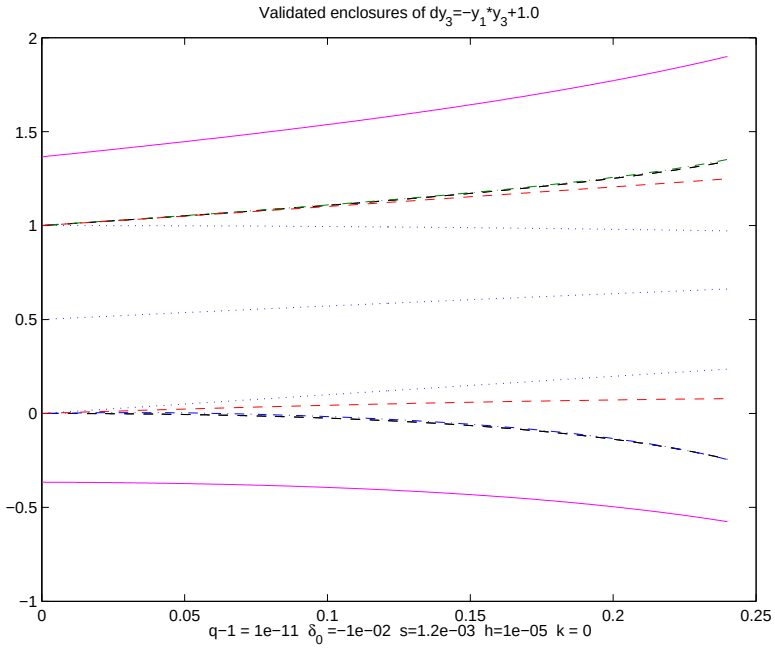

Figure 6.14: Third component of Example 8

cantly worse. Rigorous bounds could be found by replacing our optimization heuristics with a rigorous global solver with full error control.

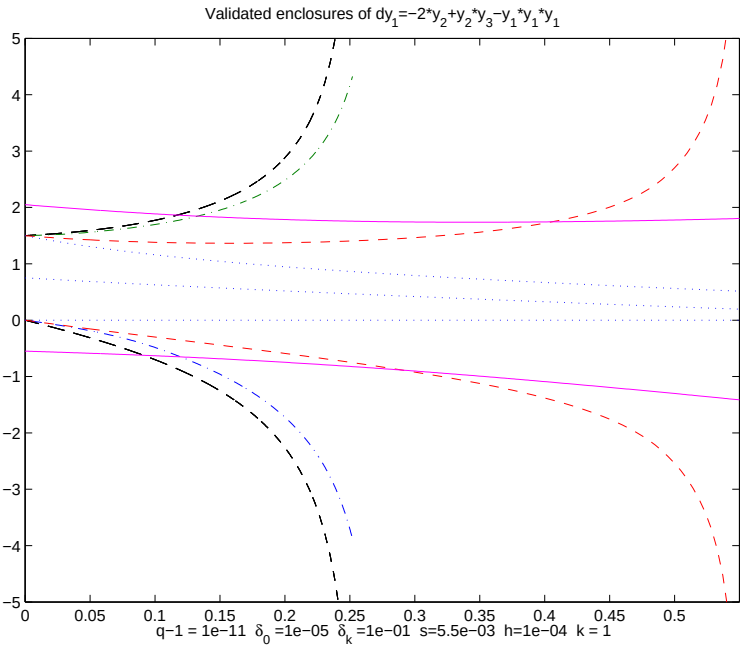

Figure 6.15: First component of Example 9

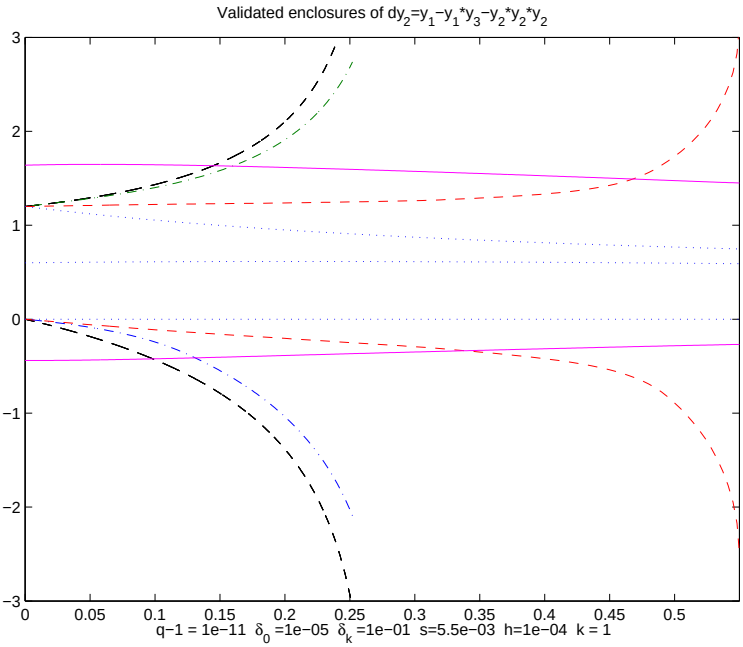

Figure 6.16: Second component of Example 9

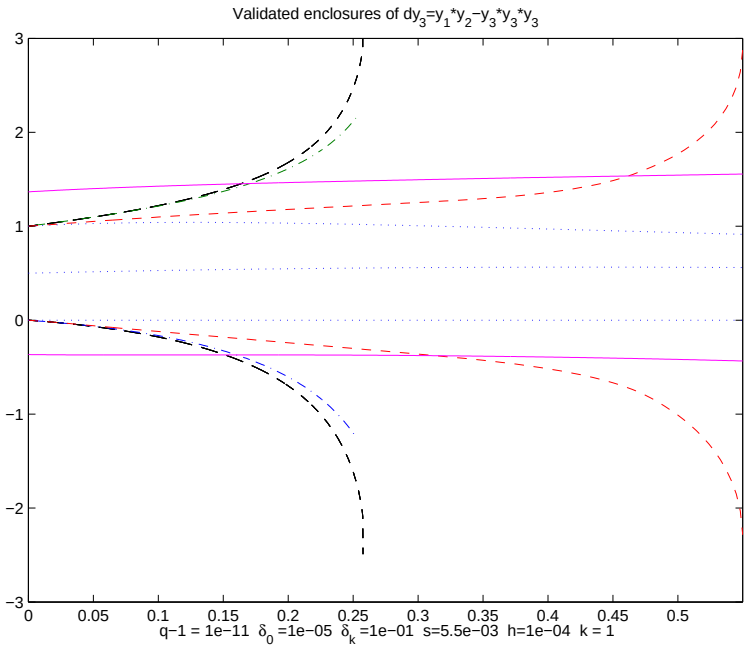

Figure 6.17: Third component of Example 9

Chapter 7

Perspectives and future work

In this dissertation, we have developed a theory to compute validated solutions of systems of ODEs with uncertain initial conditions. A new solver DIVIS has been presented to implement this theory. The purposed method computes defect estimates by using optimization techniques. Then by applying differential inequality, validated state enclosures for IVPs are computed that are compared with VALENCIA-IVP, VNODE-LP and VSPODE. Following attempts will be made to improve the efficiency of this scheme.

7.1 Rounding error control

We are using a local optimizer Ipopt that produces local solutions with multiple starts, therefore, our bounds are not fully rigorous. As mentioned in section 6.2, this problem can be resolved by using rigorous global optimization solver. We also tried COCONUT that is rigorous solver but it was very slow in our case because we need to solve a lot of optimization problems to get the bounds. Improved formulation of optimization problem will probably remove this difficulty of COCONUT solver.

At present, we are using cubic Hermite spline to approximate the solution. The error can be minimized by applying higher-order spline approximation. One can use B-splines corresponding to a function that has function value 1 and derivative 0 at all nodes.

Multiple local search also increases the computational efforts. At the moment, we concentrated on quality of the bounds but not computational cost. That is why, our solver is slower than the other existing validated ODE solvers VALENCIA-IVP, VNODE and VSPODE.

7.2 Solving the system of ODEs with uncertain parameters

To solve the systems of ODEs with uncertain parameters, we need to introduce these parameters as additional variables. Then the system (5.1) becomes,

$$y'(t) \;=\; F(t, y(t), p), \;\; y(0) \in \boldsymbol{y}_0, \;\; p \in \boldsymbol{p}, \quad t \in [0, T], \tag{7.1}$$

where $F \in C^1(D, \mathbb{R}^n)$ is a continuously differentiable function from $D \subseteq \mathbb{R}^n \times \mathbb{R}^l$ to $\mathbb{R}^n$, the function $y \in \mathbb{R}^n$, $p \in \mathbb{R}^l$ are unknown to be solved for, and $\boldsymbol{y}_0$, $\boldsymbol{p}$ are interval boxes. In order to compute the defect estimates, we will solve the optimization problems (5.12) or (5.17) by introducing new variable p as follows:
When bound growth is expected we will solve

$$\begin{aligned} &\max_{p,\epsilon,t} \;\; 2(U(t)\epsilon)^T(U(t)(F(t, u(t)+\epsilon, p) - u(t)') + U(t)'\epsilon) \\ &\text{s.t.} \quad \Delta_0 \leq \; \|U(t)\epsilon\|^2 \; \leq \Delta_+, \\ &\qquad \underline{p} \leq p \leq \overline{p}, \quad t \in [\underline{t}, \overline{t}] \end{aligned} \tag{7.2}$$

and for decaying bound, we will solve

$$\begin{aligned} &\max_{p,\epsilon,t} \;\; 2(U(t)\epsilon)^T(U(t)(F(t, u(t)+\epsilon, p) - u(t)') + U(t)'\epsilon) \\ &\text{s.t.} \quad \Delta_- \leq \; \|U(t)\epsilon\|^2 \; \leq \Delta_0, \\ &\qquad \underline{p} \leq p \leq \overline{p}, \quad t \in [\underline{t}, \overline{t}]. \end{aligned} \tag{7.3}$$

7.3 BVPs

It would be interesting to extend the work done to handle boundary value problems (BVPs).

For BVPs, validated state enclosures can be computed by computing Jacobian for the system of ODEs. For this we consider a two-point BVP of order n on a finite interval $[\underline{t}, \overline{t}]$ given in the form

$$y'(t) = F(t, y(t)), \;\; t \in (\underline{t}, \overline{t}), \;\; y_i(\underline{t}) := \underline{y}_i \text{ if } i \in I, \;\; By(\overline{t}) = b, \tag{7.4}$$

where $y, F \in R^n$. The boundary values are evaluated at two extreme points $\underline{t}$ and $\overline{t}$.

If F is linear, then these BVPs can be transformed into initial value problems (IVPs) by applying the superposition principle for solutions of linear differential equation CAP [15]. In order to solve linear BVPs, for each $i \notin I$, we solve an IVP with

$$y(\underline{t}) = y_0 \text{ which gives } y^{(0)}(t), \;\; y_{0i} = \begin{cases} \underline{y}_i & \text{if } i \in I, \\ x_i & \text{otherwise} \end{cases} \tag{7.5}$$

$y(\underline{t}) = y_0 + e^{(i)}$ and obtain a solution $y^{(i)}(t)$. Then

$$y(t) = y^{(0)}(t) + \sum_{i \notin I} x_i (y^{(i)}(t) - y^{(0)}(t)) \tag{7.6}$$

solves the ODE and the initial condition at $\underline{t}$, and we have

$$y(\overline{t}) = y^{(0)}(\overline{t}) + \sum_{i \notin I} x_i (y^{(i)}(\overline{t}) - y^{(0)}(\overline{t})). \tag{7.7}$$

Therefore, we get solution of the BVP if $By(\overline{t}) = b$. This leads to the linear system

$$\begin{aligned} Ax &= a, \\ A:i &= y^{(i)}(\overline{t}) - y^{(0)}(\overline{t}), \\ a &= y(\overline{t}) - y^{(0)}(\overline{t}). \end{aligned}$$

If A is nonsingular, we find

$$x = A^{-1}a.$$

A rigorous IVP solver produces enclosures

$$\boldsymbol{y}^{(0)}(t) \text{ for } y^{(0)}(t),$$

$$\boldsymbol{y}^{(i)}(t) \text{ for } y^{(i)}(t)$$

and hence enclosure $\boldsymbol{A}$ of A and $\boldsymbol{a}$ of a. If $\boldsymbol{A}$ is regular, then for some $x \in \boldsymbol{A}^{-1}\boldsymbol{a}$, (7.6) solves the BVP.

If F is nonlinear, a shooting method must be applied and in place of $\boldsymbol{A}$, one needs Jacobian enclosure that leads to extra work Lohner, Clark [52, 21]. One can also use homotopy-type methods like Plum [86, 87] who computed enclusion interval for eigenvalues. Watson [108] also applied homotopy-type method to a class of nonlinear two points BVPs, and solved system of nonlinear equations by using shooting method but we concentrate on IVPs.

7.4 Generalization towards differential algebraic equations

Differential algebraic equations arise in many applications especially when modelling constrained mechanical systems, electrical circuits and chemical reaction kinetics etc. We consider an implicit differential equation

$$F(t, y(t), y'(t)) = 0, \;\; t \in [t_0, t_m]. \tag{7.8}$$

If $\partial F/\partial y'$ is nonsingular, then an ODE can be obtained by solving (7.8) for y'. But it becomes impossible in case of singular $\partial F/\partial y'$. In this situation, solution y has to satisfy certain algebraic constraints. Therefore, when $\partial F/\partial y'$

is singular, (7.8) is known as differential algebraic equations. In DAEs, all variables can not be freely initialized. To solve these equations, one needs to find the consistent initial conditions. For such consistent initialization, derivatives of some of the component functions of the DAE are taken into account. The highest order of the derivative required for that purpose is known as differentiation index.

7.4.1 Definition. The nonlinear DAE (7.8) has (differentiation) index μ if μ is the minimal number of differentiations

$$F(t, y(t), y'(t)) = 0, \ \frac{dF(y'(t), y(t), t)}{dt} = 0, \cdots, \frac{d^{\mu}F(y'(t), y(t), t)}{dt} = 0 \quad (7.9)$$

such that the equations (7.9) allow to extract an explicit ordinary differential system $y'(t) = \phi(t, y(t))$ using only algebraic manipulations [96].

A number of schemes have been developed to solve these systems. For detail see [88, 89, 5, 27, 30, 10].

In principle, the scheme presented in this thesis can also be generalized towards differential algebraic equations (DAE). For that purpose, we need to extend the conditional differential inequality to implicit case.

7.5 Automatic choice of parameters

At present, we choose step length and regularizing parameters by hand. We need to make these input variables automatized. This would involve looking into the nature of the problem, Jacobian of the system of ODEs, condition number etc.

Appendix A

This section reports the samples of the AMPL model file, the data file and the command file for the 2-dimensional system of ODEs

$$\begin{aligned} y_1' &= -y_2, \\ y_2' &= y_1, \ t \in [0, 100], \quad y(t_0) \in [0, 1] \times [-1, 0]. \end{aligned}$$

AMPL model

Following is the sample of the AMPL model to solve optimization problems (5.12) and (5.17).

```
#########################
##### example7.mod  #####
#########################
#
set L1;      # set of input deltas
set L;       # no. of output delta_prime
param dim; # dimensions of the system of ODEs
set N:=1..dim; # dimensions
# dimension of the systems of ODEs and the
# preconditioner
set NP:=1..dim^2+dim;
#
######################
# parameters
######################
# coefficient of the cubic Hermite polynomials
#
# coefficient of cubic term
param AA{i in L,j in NP};
#
# coefficient of quadratic term
param BB{i in L,j in NP};
#
```

```
# coefficient of linear term
param CC{i in L,j in NP};
#
# coefficient of constant term
param DD{i in L,j in NP};
#
########################
# coefficient of the cubic Hermite polynomial U
# (solution of the system of preconditioner)
#
param A{i in N,j in N};
param B{i in N,j in N};
param C{i in N,j in N};
param D{i in N,j in N};
#
########################
# coefficient of the cubic Hermite polynomial u
# (solution of the system of ODEs)
#
param a{j in N};
param b{j in N};
param c{j in N};
param d{j in N};
#
# input Deltas
#
param Delta_l{k in L1};
param delta_l_1>=0;
param delta_l>=delta_l_1;
#
########################
#
param TT1{i in L};
param TT2{i in L};
param T1>=0;
param T2>=T1;
#
#
########################
# variables
########################
#
var eps{j in N} ;
#
var t;
#
```

```
########################################
# approximate solution U of the system
# of preconditioner
########################################
var U{i in N,j in N}=A[i,j]*(t-T1)^2*(t-T2)
     +B[i,j]*(t-T1)^2+C[i,j]*(t-T1)+D[i,j];
#
# Ist derivative of U
#
var Uprime{i in N,j in N}=2*A[i,j]*(t-T1)*(t-T2)
    +A[i,j]*(t-T1)^2+2*B[i,j]*(t-T1)+C[i,j];
#
#######################
# approximate solution u of the system of ODEs
#
var u{j in N}=a[j]*(t-T1)^2*(t-T2)+b[j]*(t-T1)^2
    +c[j]*(t-T1)+d[j];
#
# Ist derivative of u
#
var uprime{j in N}=2*a[j]*(t-T1)*(t-T2)
        +a[j]*(t-T1)^2+2*b[j]*(t-T1)+c[j];
var y{j in N}=u[j]+eps[j]; #cubic splines
#
#######################
# system of ODEs ;
#
var F{j in N}
=if j=1 then -y[2]
 else y[1];
#
#######################
# system of ODEs ;
#
var yprime{j in N}
=if j=1 then -u[2]
 else u[1];
#
#######################
var uprime_yprime{j in N}=yprime[j]-uprime[j];
#######################
# simplifying the optimization problem
#
var Y{j in N}=F[j]-uprime[j];
#
#######################
```

```
#
var prod_U_Y{i in N}=sum{j in N}(U[i,j]*Y[j]);
#
########################
#
var prod_Uprime_eps{i in N}=sum{j in N}(Uprime[i,j]*eps[j]);
#
########################
#
var sum_U_Uprime{j in N}=prod_U_Y[j]+prod_Uprime_eps[j];
#
########################
#
var eta{i in N}= sum{j in N}U[i,j]*eps[j];
var eta_normed_squared=sum{i in N}eta[i]^2;
#
########################
#solving optimization problem
# objective;
#
 maximize del_prime:2*sum{j in N}eta[j]*sum_U_Uprime[j];
 subject to e1:eta_normed_squared>=delta_l_1; #delta(l)
 subject to e2:eta_normed_squared<=delta_l; #delta(l+1)
 subject to t1:t>=T1;
 subject to t2:t<=T2;
```

AMPL data

A typical AMPL data file for this model is given as:

```
############################
###### example7.dat  #######
############################
data;
############################

# sets
set L1:=  1 2; # number of input Deltas
set L:=  1; # number of output Delta_prime

########################

# parameters
param dim:= 2; # dimension of the system of ODEs
```

```
##########################

var eps:=
1  -1.64825889   2  0.98911657   ;

##########################

var t:= 99.9999; # initial guess for time span
##########################

param: TT1  TT2:=
1      99.9999  100 ; # time span

##########################

# coefficient of cubic term
param AA: 1 2 3 4 5 6:=
1 -0.10931 -0.02951 -0.02061 -0.03753 0.03811 -0.02049;

# coefficient of quadratic term
param BB: 1 2 3 4 5 6:=
1 -0.08896 0.34218 -0.11753 0.06578 -0.06681 -0.11698 ;

# coefficient of linear term
param CC: 1 2 3 4 5 6:=
1 0.68436 0.17791 0.13541 0.23129 -0.23486 0.13479 ;

# constant coefficient
param DD: 1 2 3 4 5 6:=
1 0.17791 -0.68435 0.23215 -0.13653 0.13866 0.23106 ;

##########################

param: Delta_l:=

1   0.26801291946
2   0.26857557312 ;
;
```

AMPL run file

These model and data files are executed using a separate command file namely ode.sa1 as follows:

```
############################
######### ode.sa1  #########
############################
# calling AMPL model
  model example7.mod;

# calling input data file for AMPL model
  data example7.dat;

  printf "function del_prime=ampl_output()\n"
   >ampl_output.m;

# Assigning suitable positions to coeffients
# of cubic Hermite interpolants of the
# system of precontioner from AMPL data file.

   for {j in N}{for {k in N}{
   let A[k,j] := AA[1,dim+dim*(k-1)+j];
   let B[k,j] := BB[1,dim+dim*(k-1)+j];
   let C[k,j] := CC[1,dim+dim*(k-1)+j];
   let D[k,j] := DD[1,dim+dim*(k-1)+j];
   }

# Assigning suitable positions to coeffients
# of cubic Hermite interpolants of the system
# of ODEs from AMPL data file.

   let a[j] := AA[1,j];
   let b[j] := BB[1,j];
   let c[j] := CC[1,j];
   let d[j] := DD[1,j];
   }

# solving L AMPL models to compute delta prime.

   for {l in L}{
   let T1 := TT1[1]; # t_low
   let T2 := TT2[1]; # t_up
   let delta_l_1:=Delta_l[l]; # delta_0
   let delta_l:= Delta_l[l+1]; # delta_1
```

```
  option substant 1;  #substitution

  option show_stats 1; # show what presolve does
  option solver ipopt;
  option display_precision 15;

  solve;


# checking whether the optimization problem has
# feasible solution or not.

   if match (solve_message, "infeasible") > 0
   then
   {
    printf "del_prime=[];\n">ampl_output.m;
     break;
   }

# checking whether the optimization problem is
# bounded or not.
   else if match (solve_message, "unbounded") > 0
   then
   {
    printf "del_prime=[];\n">ampl_output.m;
     break;
   }

   printf "del_prime(%i)=%12.24f;\n",l,
  del_prime>ampl_output.m
   }
```

Appendix B: The DIVIS package

Installation and use

This section reports the installation and use of our newly developed solver DIVIS. Sample of input files for (7.10)

$$y_1' = -y_2, \tag{7.10}$$
$$y_2' = y_1, \ t \in [0, 100], \quad y(t_0) \in [0, 1] \times [-1, 0]. \tag{7.11}$$

is given as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% example.m %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function dy =example7_deq
%
dy{1}= '-y(2)';
dy{2}= 'y(1)';
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% initial_condition %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [yinf,ysup] = example7_icond
yinf= [0 -1];
ysup= [1 0];
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% time_span %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [t0,splines,tend]= example7_tl
```

```
t0 = 0;
splines=1000;
tend =100;
%
```

To install the software, one needs to copy the divis folder on the computer where it is going to be executed, change to the directory <path>/divis/main, where <path> is the directory in which the divis folder is copied. Then make the following changes.

- Go to ../divis/main/config.m and change the path in conf.home='/users/qaisra/'; to path <path> containing the divis folder.
- Define your ODE model consisting of three MATLAB files according to the sample given above. Execute ../divis/main/driver.m to solve system of ODEs. For user's convenience, the MATLAB file for driver is given below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  driver
%
%  SYNOPSIS:
%
%  driver.m is the main calling sequence that calls
%  MATLAB subroutine batch.m to execute this solver.
%
conf=config;
%
eval(['cd ',conf.main]);
%
option= 'option solver ipopt;% optimization solver
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  file_list is the number of the example to be
%  solved in subdirectory
%
%   ..\divis\test_problems\ex#file_list
%
%  This subdirectory consists of MATLAB files for the
%  system of ODEs and its initial conditions. e.g;
```

```
%  if we choose file_list =1, then this solver will
%  execute example1 in ..\divis\test_problems\ex#1.
%  The folder ex#1 must have three input files
%
%  1. example1_deq.m   %consisting of system of ODEs,
%
%  2. example1_icond.m %consisting of initial box y0,
%
%  3. example1_tl.m    %containing time span [t0,tend].
%
  file_list     =1;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  fac_refine_plot is factor used to make a refined plot
fac_refine_plot =10;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  fac_delta4reg is regularization factors for the
%  preconditioner.
%
%  Uprime=-UFy-U(del_0+del_k(U'U)^k), k=0,1,2.
%
%  In this case, del_0 and del_k are regularization
%  factor so we take fac_delta4reg=[del_0;del_k], k=0,1,2.
%  If no regularization is involved then take
%  fac_delta4reg=[0;0]; In that case preconditioner
%  reduces to Uprime=-UFy.
%
  fac_delta4reg   =[5e-1;5e-1];
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  fac_k4reg is the variable k in the preconditioner.
%
%  Uprime=-UFy-U(del_0+del_k(U'U)^k), k=0,1,2.
%
  fac_k4reg =    1;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
   h=1e-4;   % step length
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  q is the multiplcation factor to compute Delta_1 as
```

```
%
%  Delta_pos=q*Delta0,
%  required to solve the optimization problem
%  for the case when bound growth is expected and
%
%  Delta_neg=Delta0/q,   with q>1
%  for the case when decaying bound is expected.
%
   q               =1e-6;
%
%  If the system is to be solved with preconditioning
%  put precond=1 otherwise precond=0.
%
   precond=1;
%
%
   batch(conf,option,file_list,fac_refine_plot,...
  fac_delta4reg,fac_k4reg,q,h,precond).
%
```

Headers of MATLAB routines

This section reports a list of headers of MATLAB routines in our implementation for computing the error bounds of ODEs. These are arranged in the same sequence in which MATLAB routines are executed. In Figure 5.1, reader can observe which files are automatically generated from input files.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% batch %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%   batch(conf,option,file_list,fac_refine_plot,...
%  fac_delta4reg,fac_k4reg,q,h,precond).
%
%  SYNOPSIS:
%
%  batch.m is a calling sequence that evaluates the
%  system of ODEs with initial conditions. To compute
%  and then plot the component-wise error in the
%  solutions, it calls the following subroutines
%
%  1.   mk_system_preconditioner.m to compute the
```

```
%       Jacobian of system of ODEs required to construct
%       preconditioner.
%
%  2.   print_systems_ODEs_preconditioner.m to print the
%       system of ODEs and the preconditioner. For detail
%       see synopsis of print_systems_ODEs_preconditioner.m.
%
%  3.   print_ample_model.m to print an ampl model to
%       solve the optimization problem.
%
%  4.   print_ampl_runfile.m that prints an AMPL file
%       ode.sa1. to execute AMPL model and data files
%       to solve the optimization problem.
%
%  5.   bds_regularized_precond.m that computes component-
%       wise error in the solutions by applying conditional
%       differential inequality.
%
%
%  INPUT:
%
%   conf:             configuring path
%
%   option:           optimization solver option
%
%   file_list:        number of the problem consisting of the
%                     systems of ODE
%
%   fac_refine_plot: factor used to make a refined plot
%
%   fac_delta4reg:    regularization factors for preconditioner
%                     Uprime=-UFy-U(del_0+del_k(U'U)^k), k=0,1,2.
%                     In this case, del_0 and del_k's are
%                     regularization factor so we take
%                     fac_reg=[del_0;del_k]
%
%   fac_k4reg:        variable  k=0,1,2 in the preconditioner
%                     Uprime=-UFy-U(del_0+del_k(U'U)^k)
%
%   h:                step-length
%
%   q:                multiplcation factor to compute Delta_1
%                     as Delta_1=q*Delta0 with q>1.
%
%   precond           Boolean variable 0 or 1.
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%  mk_system_preconditioner  %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  mk_system_preconditioner(deq,fac_delta4reg,...
%                           fac_k4reg,expath)
%
%  SYNOPSIS:
%
%  mk_system_preconditioner.m prints a MATLAB file
%  compute_Jacobian.m to compute the Jacobian Fy of the
%  system of ODEs and then simplifies the preconditioner
%  Uprime = -UFy-U(del_0+del_k(U'U)^k),
%           for k= 0,1,2 or 3.
%
%  INPUT:
%
%  deq:            system of ODEs
%
%  fac_delta4reg: regularization factors del_0,
%                  del_k for preconditioner
%
%  fac_k4reg:      factor k=0,1,2 in the preconditioner
%                  Uprime=-UFy-U(del_0+del_k(U'U)^k)
%
%  expath:         directory where the file
%                  compute_Jacobian.m is printed.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% num2symbolic %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  deq=num2symbolic(deq)
%
%  SYNOPSIS:
%
%  num2symbolic.m converts the system of ODEs from
%  numerical syntax to the symbolic one required
%  for symbolic toolbox.
%
%  INPUT:
%
```

```
%  deq: system of ODEs with numerical syntax
%       e.g., y'=y(1)
%
%  OUTPUT:
%
%  deq: system of ODEs with symbolic syntax
%       e.g., y'=y1
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% print_systems_ODEs_preconditioner %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  print_systems_ODEs_preconditioner(name,dim,deq,expath)
%
%  SYNOPSIS:
%
%  print_systems_ODEs_preconditioner(name,dim,deq,expath)
%  automatically prints a MATLAB file consisting of
%  the systems of ODEs and the preconditioner. It
%  calls symbolic2num.m to convert the variables
%  of the system of ODES and the preconditioner
%  from symbolic syntax to numerical one.
%
%  INPUT:
%
%  name:     name of the problem
%
%  dim:      dimension of the system of ODEs
%
%  deq:      system of ODEs
%
%  expath:    directory where two MATLAB files,
%             one consisting of the systems of ODEs
%             and the preconditioner and the other
%             with only system of ODEs are
%             automatically printed
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%% symbolic2num %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  Var_eq=symbolic2num
```

```
%
%  SYNOPSIS:
%
%  symbolic2num.m converts the syntax of variables
%  for the systems of ODEs and the preconditioner
%  from symbolic to numerical one to compute the
%  numerical solutions for these systems.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% print_ampl_model %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   CALLING SEQUENCE
%
%   print_ample_model(name,deq,yinf,file_path)
%
%   SYNOPSIS:
%
%   print_ampl_model.m automatically prints a model
%   file to solve a maximization problem in AMPL.
%   This model computes del_prime for the problem
%   y'=F(t,y), y=u+eps. For this purpose,
%   coefficients a1,b1,c1,d1 and a2,b2,c2,d2, of
%   the cubic Hermite splines u and U (solutions of
%   systems of ODEs and the preconditioner respectively)
%   are computed by compute_approx_soln.m.
%
%   The optimization problem has the form
%
%   max     2(eta)'*(U*(F(t,u+eps)-uprime)+Uprime*eps)
%   s.t.    delta_l<=norm(eta)<= delta_l+1,
%   s.t.    t_low <= t <= t_up,
%   where
%   u      = a(t-t_low)^2(t-t_up)+b(t-t_low)^2
%            +c(t-t_low)+d,
%   uprime = 2a(t-t_low)(t-t_up)+a(t-t_low)^2
%            +2b(t-t_low)+c
%   u      = A(t-t_low)^2(t-t_up)+B(t-t_low)^2
%            +C(t-t_low)+D,
%   Uprime = 2A(t-t_low)(t-t_up)+A(t-t_low)^2
%             +2B(t-t_low)+C.
%
%   eps is input parameter for this problem and is
%   computed separately in subroutine
%   compute_t_l_delta_l.m and eta=U(t)*eps.
%
```

```
%   This subroutine calls mat2ampl.m to convert the MATLAB
%   syntax of variables for the systems of ODEs and the
%   preconditioner into AMPL syntax.
%
%   INPUT:
%
%     name:         problem name
%
%     deq:          system of ODEs
%
%     dim:          dimension of the system of ODEs
%
%     file_path:    path of directory to print data file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%% mat2ampl %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  deq=mat2ampl(deq)
%
%  SYNOPSIS:
%
%  mat2ampl.m converts the MATLAB syntax of variables
%  for the systems of ODEs and the preconditioner
%  into AMPL syntax required to solve AMPL model.
%
%   INPUT:
%
%   deq:          system of ODEs  % MATLAB syntax
%                 e.g., y'=y(1)
%   OUTPUT:
%
%   deq:          system of ODEs  % AMPL syntax
%                 e.g., y'=y[1]
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%% print_ampl_runfile %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   CALLING SEQUENCE
%
%   print_ampl_runfile(name,file_path)
%
%   SYNOPSIS:
```

```
%
%   print_ampl_runfile.m automatically prints a file
%   ode.sa1 that calls ampl model and data files to
%   solve the optimization problem. The output
%   delta_prime is printed in a file ampl_output.m
%
%   INPUT:
%
%  name:          problem name
%
%  file_path:     path of directory to print the file
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% bds_regularized_precond %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  bds_regularized_precond(name,M,dim,n_steps,h,...
%                          q_up,file_path)
%
%  SYNOPSIS:
%
%  bds_regularized_precond.m computes the error bounds
%  of the system of ODEs. For each interval,
%  [t(i), t(i+1)], i=1:n_steps, we plot approximate
%  solutions at extreme points of initial box that are
%  computed from compute_approx_soln.m. To solve
%  optimization problems in AMPL, we compute the
%  coefficients of cubic Hermite polynomials a,b,c,d
%  and A,B,C,D for the systems of ODEs and the
%  preconditioner respectively by using
%  compute_coeff_hermite.m. A list of defect estimates
%  Delta's and their corresponding t's is computed from
%  compute_t_l_Delta_l.m.
%  We construct tlist, Delta_list from these Delta's and
%  t's. Using the coefficients a,b,c,d and  A,B,C,D, of
%  continuously differentiable solutions ylist, Ulist of
%  the system of ODEs and the preconditioner are evaluated
%  at tlist. Finally the lower bound ylist-cor and upper
%  bound ylist+cor of the component-wise error in the
%  solution of the system of ODEs are plotted, where
%  cor = Vnorm.*sqrt(Delta_list), Vnorm = norm(inv(Ulist)).
%
%   INPUT:
%
```

```
%   problem:    name of problem
%
%   M:          factor to refine the plot
%
%   dim:        dimension of the system of ODEs
%
%   n_steps:    number of steps to compute error bounds
%
%   h:          step length
%
%   q:          multiplcation factor to compute Delta_1 as
%               Delta_pos=q*Delta0, required to solve the
%               optimization problem
%
%   file_path:  path of directory where MATLAB files
%               consisting of system of ODEs, the
%               preconditioner, initial conditions are
%               printed
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% compute_approx_soln %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   CALLING SEQUENCE:
%
%  [tspline ,y0v,yinf,ysup,y,Y,yprime,yinf_prime,...
%  ysup_prime,Yprime,Delta0]=compute_approx_solutions(name,...
%  t0,tend,n_steps)
%
%   SYNOPSIS:
%
%  compute_approx_solutions.m computes approximate
%  solutions of the system of ODEs and the preconditioner
%  at mid point of the initial box by using ODE solver ode45.
%  For that purpose, initial conditions y_inf,y_sup for the
%  system of ODEs are evaluated. Initial Delta, is computed
%  by Delta0=dim/4. This choice of Delta0 makes it sure
%  that initial box lies within the ellipsoid.
%
%  Approximate solutions to the system of ODEs are also
%  computed at extreme points of the initial box using
%  ode45. Then systems of ODEs and the  preconditioner are
%  evaluated at these solutions required to compute the
%  coefficients of the cubic Hermite splines.
%
%
```

```
%   INPUT:
%
%   name:         name of the problem
%
%   t0:           starting time
%
%   tend:         ending time
%
%   n_steps:      number of steps to compute error bounds
%
%   OUTPUT:
%
%   tspline       spline nodes
%
%   y0v:          initial condition for the systems of ODES
%                 and the preconditioner
%
%   yinf:         approximate solution computed at yinf
%
%   ysupe:        approximate solution computed at ysup
%
%   y:            approximate solution computed at mid point
%                 u0=(yinf+ysup)/2
%
%   Y:            approximate solution to the system of the
%                 preconditioner
%
%   yprime:       system of ODEs evaluated  at t and y
%
%
%   yinf_prime:   system of ODEs evaluated at t and yinf
%
%
%   ysup_prime:   system of ODE evaluated at t and ysup
%
%
%   Yprime:       system of the precontioner evaluated at t
%                 and Y
%
%   condY:       condition number of the system of preconditioner
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% compute_coeff_hermite %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
```

```
%
%  [coeff_ode,coeff_precond,coeff_ysup,coeff_yinf]=...
%   compute_coeff_hermite(t_low,t_up,y_approx,...
%   Y_approx,yprime,Yprime,dim)
%
%  SYNOPSIS:
%
%  compute_coeff_hermite.m computes the coefficients of
%  cubic Hermite polynomial for the systems of ODEs and
%  the preconditioner
%
%  INPUT:
%
%  t_low:        starting time
%
%  t_up:         end time
%
%  y:            approximate solution of the system of ODEs
%                computed at mid point u0 of the initial box
%                [yinf,ysup]
%
%  y_inf:        approximate solution of the system of ODEs
%                computed at yinf
%
%  y_sup:        approximate solution of the system of ODEs
%                computed at ysup
%
%  Y:            approximate solution of the system of
%                preconditioner computed at mid point u0
%
%  yprime:       Value of the system of ODEs evaluated at
%                mid point of u0
%
%  yinf_prime:   Value of the system of ODEs evaluated at  yinf
%
%  ysup_prime:   Value of the system of ODEs evaluated at ysup
%
%  Yprime:       Value of the system of the system of the
%                preconditioner evaluated at mid point
%
%  dim:          dimension of the system of ODEs
%
%
%  OUTPUT:
%
%  coeff_ode:     coefficients of the cubic Hermite polynomial
```

```
%                 of the system of ODEs solved at u0
%
%  coeff_precond: coefficients of the cubic Hermite polynomial
%                 of the system of preconditioner solved at u0
%
%  coeff_ysup:    coefficients of the cubic Hermite polynomial
%                 of the system of ODEs solved at yinf
%
%  coeff_yinf:    coefficients of the cubic Hermite polynomial
%                 of the system of ODEs solved at ysup
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% compute_t_l_Delta_l %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE:
%
%  [coeff_ode_t1,coeff_yinf_t1,coeff_ysup_t1,...
%   coeff_precond_t1,t_l,Delta_t_l]=compute_t_l_Delta_l(name,...
%   q,U,dim,coeff_ode,coeff_yinf,coeff_ysup,coeff_precond,...
%   t_low,t_up,y0v,Delta0,file_path)
%
%  SYNOPSIS:
%
%  compute_t_l_Delta_l.m computes lists of t_l and Delta_l
%  for each polynomial piece required to plot component wise
%  error in the solution. To do so, an optimization problem
%  is solved by compute_delprime.m to compute Delta_prime_1.
%  Depending upon the sign of Delta_prime_1, lists of t_l
%  and Delta_l are computed. For example,
%  if Delta_prime_1>0,
%   compute t_l and Delta_l by t_l_Delta_l4pos_delprime.m.
%  If Delta_prime_1<0,
%  apply t_l_Delta_l4neg_delprime.m to compute t_l, Delta_l.
%  If Delta_prime_1=0, then take t_l = [t0,t_up] and
%  Delta_l=[Delta0 Delta0].
%  There is a possibility that t_l can not reach t_up. In that
%  case, choose t_low=t_1=t_l(end) and compute coefficients of
%  the cubic hermite polynomials y and Y at new t_low and again
%  solve the optimization problem in AMPL for
%  Delta0=Delta_l(end) to compute Delta_prime_1. Again depending
%  upon the sign of Delta_prime_1, compute t_l and Delta_l.
%  Continue until t_l(end)=t_up.
%
%[coeff_ode_t1,coeff_yinf_t1,coeff_ysup_t1,coeff_precond_t1,...
%   t_l,Delta_t_l]=compute_t_l_Delta_l(name,q,U,dim,...
```

```
%   coeff_ode,coeff_yinf,coeff_ysup,coeff_precond,t_low,t_up,...
%   y0v,Delta0,file_path)
%   INPUT:
%
%   name:           problem name
%
%   q:              multiplcation factor to compute Delta_1 as
%                   Delta_1=q*Delta0 with q>1
%
%   U:              solution of of the system of preconditioner
%                   at midpoint u0
%
%   dim:            dimension of the system of ODEs
%
%   t_low:          starting time
%
%   t_up:           end time
%
%   coeff_ode:      coefficients of the cubic Hermite polynomial
%                   of the system of ODEs solved at u0
%
%   coeff_precond:  coefficients of the cubic Hermite polynomial
%                   of the system of preconditioner
%
%   coeff_ysup:     coefficients of the cubic Hermite polynomial
%                   of the system of ODEs solved at yinf
%
%   coeff_yinf:     coefficients of the cubic Hermite polynomial
%                   of the system of ODEs solved at ysup
%
%   y0v:            initial conditions for the system of ODEs
%                   and the preconditioner
%
%   Delta0:         starting input Delta
%
%   file_path:      path of the directory
%
%
%   OUTPUT:
%
%   coeff_ode_t1:   coefficients of cubic Hermite polynomials
%                   of the system of ODEs computed at
%                   [t_1,t_up]
%
%   coeff_yinf_t1:  coefficients of cubic Hermite polynomials
%                   evaluated at [t_1,t_up] for the system of
```

```
%                   ODEs solved at yinf
%
%   coeff_ysup_t1:  coefficients of cubic Hermite polynomials
%                   evaluated at [t_1,t_up] for the system of
%                   ODEs solved at ysup
%
%  coeff_precond_t1:coefficients of cubic Hermite polynomials
%                   for the system of preconditioner evaluated
%                   at [t_1,t_up]
%
%   t_l:            list of changing time points
%
%   Delta_l:        list of changing Delta
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% compute_delprime %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE:
%
%  Del_prime_1 = compute_delprime(name,dim,coeff,t_span,...
%                       y0v,Delta0,Delta_1,U,file_path
%
%  SYNOPSIS:
%
%  compute_delprime.m computes Delta_prime_1 by solving
%  optimization problems in AMPL. A list of eta's is
%  constructed by taking tim=5,
%  eta_pos =sqrt(Delta0+(Delta_1-Delta0)/tim*[0:tim]),
%  eta_neg=-eta_pos, eta = [eta_pos eta_neg]' and
%   eps= (U\eta)'.
%  Due to local optimization solver Ipopt, for each eps,
%  an optimization problem
%
%   max     2(eta)'*(U*(F(t,u+eps)-uprime)+Uprime*eps)
%   s.t.    Delta0<=norm(eta)<= Delta_1
%   s.t.    t_low <= t <= t_up
%
%  is solved in AMPL to compute Delta_prime. Then worst
%  case is taking into account by choosing
%  Delta_prime_1=max(Delta_prime).
%
%
%
%   INPUT:
%
```

```
%   name:         problem name
%
%   dim:          dimension of the system of ODEs
%
%   coeff:        coefficients of cubic Hermite polynomials
%  of the system of ODEs and the preconditioner
%
%   tspan:        time span [t_low,t_up]
%
%   y0v:          initial conditions for the systems of ODEs and
%                 the preconditioner
%
%   Delta0:       starting input Delta
%
%   Delta_1:      Delta_1 = Delta0*q, or Delta_1 = Delta0/q, q>1
%
%   U:            solution of of the system of preconditioner
%                 at t_low
%
%
%   file_path:    directory path where the file exist
%
%
%   OUTPUT:
%
%   Delta_prime_1:  Worst solution of optimization problem
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% print_ampl_data %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   CALLING SEQUENCE
%
%   print_ampl_data(name,eps,dim,coeff,tspan,y0v,...
%    Delta_in,file_path)
%
%   SYNOPSIS:
%
%   print_ampl_data.m prints a data file for ampl model of ODE.
%   It prints coefficients A, B, C and D of the cubic Hermite
%   polynomials of the solution of the systems of ODEs and the
%   preconditioner arranged in tabular form by coef_table.m,
%   time span for this problem, input Deltas, Delta0,
%   and initial guess for the variable eps.
%
%   INPUT:
```

```
%
%   name:         problem name
%
%   eps:          initial guess for the variable eps in
%                 optimization problem
%
%   dim:          dimension of the system of ODEs
%
%   coeff:        coefficients of cubic Hermite polynomials
%
%   tspan:        time span for ith interval
%
%   y0v:          initial conditions for the systems of ODEs
%                 and the preconditioner
%
%   Delta_in:     list of input Deltas, (Delta_0, Delta_1)
%
%   file_path:    path of directory to print data file
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%% coef_table %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE
%
%  [A, B,C,D]=coef_table(coeff,y0v)
%
%  SYNOPSIS:
%
%  coef_table.m arranges coefficients of the cubic Hermite
%  polynomial of the system of ODEs and the preconditioner
%  computed by compute_coeff_hermite.m in the format
%  required for AMPL data.
%
%
%  INPUT
%
%  coeff:   coefficients of the cubic Hermite polynomials
%           of the system of ODEs and the preconditioner
%
%  y0v:     initial conditions for the system of ODEs
%
%  OUTPUT
%
%  A:       coefficient of cubic term
%  B:       coefficient of quadratic term
```

```
%  C:       coefficient of linear term
%  D:       coefficient of constant term
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% t_l_Delta_l4pos_delprime %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE:
%
%  [t_l, Delta_l]=t_l_Delta_l4pos_delprime(name,Delta0,...
%  q,Delta_1,t_low,t0,t_up,Delta_prime,U,coeff,...
%  dim,y0v,file_path)
%
%  SYNOPSIS:
%
%  When Delta_prime>0, t_l_Delta_l4pos_delprime.m computes
%  list of t_l and Delta_l, qused, and fac, where qused is
%  the last value of q used to compute Delta_1 and is taken
%  as initial q for Delta_1 in next iteration and
%  fac = (t_1-t0)/(t_up-t_low) is used for initial guess of
%  Delta_1 for next iteration, that is,
%    if fac >= .2,
%       Delta_1 = Delta0*q;
%    else
%       Delta_1 = Delta0/q;
%    end
%   Using input Delta0, Delta_1,t0,
%  and t_up, applying cond_diff_inequality.m to compute t_1
%  and Delta_t_1. Now check (t_1-t0) >= (t_up-t0)/5. If yes,
%  then take t_l=t_1, Delta_l = Delta_t_1 otherwise apply
%  iterative_case4pos_del_prime.m to compute t_l and Delta_l.
%
%  INPUT:
%
%  name:       problem name
%
%  Delta0:     starting input Delta.
%
%  q:          multiplcation factor to compute Delta_1 as
%              Delta_1=q*Delta0 with q>1
%
%  Delta_1:    Delta_1= Delta0*q, or Delta_1 = Delta0/q, q>1
%
%  U:        solution of of the preconditioner at t_low
%
%  dim:        dimension of the system of ODEs
```

```
%
%  t_low:       starting time
%
%  t0:          t_low<t0<t_up,
%
%  t_up:        end time
%
%  coeff:       coefficients of cubic Hermite polynomials of
%
%               the systems of ODEs
%
%  y0v:         initial conditions for the system of ODEs
%               and the preconditioner
%
%
%  Delta_prime: ampl output
%
%  file_path:  path where the file exist
%
%
%   OUTPUT:
%
%   t_l:          list of changing time points
%
%   Delta_l:      list of changing Delta
%
%   fac:        a factor (t_1-t0)/(t_up-t_low)  used for
%               initial guess of Delta_1 for next iteration
%
%   qused:      last value of q used in iterative scheme.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% iterative_case4pos_del_prime %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE:
%
%  [t_l,Delta_t_l,fac,qused] = ...
%    iterative_case4pos_del_prime(name,t_low,t0,t_up,U,...
%    q,Delta0,y0v,coeff,dim,file_path)
%
%  SYNOPSIS:
%
%  iterative_case4pos_del_prime computes t_l, Delta_l and qused,
%  and fac, where qused is the last value of q used to compute
%  Delta_1 and is taken as initial q for Delta_1 in next
```

```
%  iteration and fac = (t_1-t0)/(t_up-t_low) is used for
%  initial guess of Delta_1 for next iteration, that is,
%    if fac >= .2,
%       Delta_1 = Delta0*q;
%    else
%       Delta_1 = Delta0/q;
%    end
%  If Delta_prime_1>0, compute t_1, Delta_1 as an output of
%  cond_diff_inequality.m.
%  If t_1 couldn't reach t_up, then check whether this t_1 is
%  greater or equal to 20% of the interval [t0 t_up] or not.
%  If yes, return t_l=[t0,t_1] and Delta_l=[Delta0,Delta_t_1].
%
%  But if it is not the case, increase Delta_1 by increasing
%  the value of q and then with Delta_1 = Delta0*q, solve the
%  optimization problem. If Delta_prime_1>0, then proceed as
%  above to compute t_1, otherwise return t_l=[t0,t_1] and
%  Delta_l=[Delta0, Delta0].
%  Continue this process until t_1(end)=t_up or Delta_prime<=0.
%
%
%  INPUT:
%
%  name:        problem name
%
%  q:           multiplcation factor to compute Delta_1 as
%                Delta_1=q*Delta0 with q>1
%
%  U:           solution of of the system of preconditioner
%                computed at t_low
%
%  dim:         dimension of the system of ODEs
%
%  t_low:       starting time
%
%  t_up:        ending time
%
%  t0:           t_low<t0<t_up,
%
%  coeff:       coefficients of cubic Hermite polynomials
%
%  y0v:         initial conditions for the system of ODEs
%
%  Delta0:      starting input Delta.
%
%  file_path:   directory path where the file exist
```

```
%
%
%   OUTPUT:
%
%   t_l:       list of changing time points
%
%   Delta_l:   list of changing Delta
%
%   fac:       a factor (t_1-t0)/(t_up-t_low)  used for
%              initial guess of Delta_1 for next iteration
%
%   qused:     last value of q used in iterative scheme.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% t_l_Delta_l4neg_delprime %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE:
%
%  [t_l,Delta_t_l,fac,q] = t_l_Delta_l4neg_delprime(name,...
%         Delta0,q,t_low,t0,t_up,U,coeff,dim,y0v,file_path)
%
%  SYNOPSIS:
%
%  t_l_Delta_l4neg_delprime.m computes list of t_l and Delta_l
%  when AMPL output Delta_prime is negative.
%  An AMPL data file is printed with new Delta_1=Delta0/q,
%  to solve an optimization problem
%
%   max     2(eta)'*(U*(F(t,u+eps)-uprime)+Uprime*eps)
%   s.t.    Delta_1<=norm(eta)<= Delta0
%   s.t.    t_low <= t <= t_up
%
%  If ampl output Delta_prime_1<0, then compute changing time
%  point t_1 and changing Deltat_1 by cond_diff_inequality.m
%  otherwise return t_l=[t0,t_1] and Delta_l=[Delta0, Delta0].
%
%  If t_1<t_up, then check whether (t_1-t0) >= (t_up-t0)/5 or
%  not. If yes, return t_l=[t0,t_1] and
%  Delta_l=[Delta0,Delta_t_1] otherwise compute t_l and
%  Delta_l by applying iterative_case4neg_del_prime.m.
%
%
%   INPUT:
%
%   name:       problem name
```

```
%
%   Delta0 :    starting input Delta.
%
%   q:          multiplcation factor to compute Delta_1
%
%   t_low :     starting time
%
%   t0:         t_low<t0<t_up
%
%   t_up:       end time
%
%   U:          solution of of the system of preconditioner
%               at t0.
%
%   coeff:      coefficients of cubic Hermite polynomials
%               for the system of ODEs and the preconditioner
%
%   dim:        dimension of the system of ODEs
%
%   y0v:        initial conditions for the system of ODEs
%
%   file_path:  directory path where the file exist
%
%
%   OUTPUT:
%
%   t_l:        list of changing time points
%
%   Delta_l:    list of changing Delta
%
%   fac:        a factor (t_1-t0)/(t_up-t_low)  used for
%               initial guess of Delta_1 for next iteration
%
%   qused:      last value of q used in iterative scheme.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% iterative_case4neg_del_prime %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE:
%
%  [t_l,Delta_t_l,fac,q] = iterative_case4neg_del_prime(name,...
%    t_low,t0,t_up,q,Delta0,y0v,U,coeff,dim,file_path)
%
%%  SYNOPSIS:
%
```

```
%  iterative_case4neg_del_prime computes t_l, Delta_l, qused,
%  and fac.
%  If Delta_prime_1 < 0 and t_1 < t_up then decrease Delta_1
%  by decreasing the value of q and then for Delta_1 = Delta0/q,
%   solve the optimization problem as in case of
%  t_l_Delta_l4neg_delprime.m.
%
%  If ampl output Delta_prime_1<0, compute changing time point
%  t_1 and changing Deltat_1 by applying cond_diff_inequality.m
%  otherwise return t_l=[t0,t_1] and Delta_l=[Delta0, Delta0].
%
%  If t_1<t_up, then check whether (t_1-t0) >= (t_up-t0)/5 or
%  not. If yes, return t_l=[t0,t_1] and
%  Delta_l=[Delta0,Delta_t_1] otherwise decrease Delta_1
%  by decreasing the value of q and proceed as above to solve
%  the optimization problem.
%
%  Continue this process until t_1(end)=t_up or Delta_prime<=0.
%
%
%  INPUT:
%
%  name:         problem name
%
%  t_low:        starting time
%
%  t0:           t_low<t0<t_up,
%
%  t_up:         ending time
%
%  q:            multiplcation factor to compute Delta_1 as
%                Delta_1=q*Delta0 with q>1
%
%  Delta0:       starting input Delta.
%
%  y0v:          initial conditions for the system of ODEs
%
%  U:            solution of of the system of preconditioner
%                computed at t_low
%
%  coeff:        coefficients of cubic Hermite polynomials
%
%  dim:          dimension of the system of ODEs
%
%  file_path:    directory path where the file exist
%
```

```
%
%   OUTPUT:
%
%   t_l:       list of changing time points
%
%   Delta_l:   list of changing Delta
%
%   fac:       a factor (t_1-t0)/(t_up-t_low)  used for
%              initial guess of Delta_1 for next iteration
%
%   qused:     last value of q used in iterative scheme.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% cond_diff_inequality %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    CALLING SEQUENCE:
%
%    [t_1, Delta_t_1] = cond_diff_inequality(Delta_1,Delta0,...
%                         Delta_prime, t_low, t_up)
%
%    SYNOPSIS:
%
%    cond_diff_inequality.m computes t_1 as a changing time
%    point,Delta_1 as changing Delta at t_1. With starting
%    time t_low, Delta0, Delta_1 and Delta_prime, compute t_1.
%    Check if t_1 >= t_up, then compute Delta_t_1 as a function
%    of t_up. But if t_1 < t_up, use t_new to compute Delta_t_1.
%
%    INPUT:
%
%    Delta0:      starting Delta
%
%    Delta_1:     Delta0*q, q>1
%
%    Delta_prime: ampl output
%
%    t_low:       starting time
%
%    t_up:        end time
%
%
%    OUTPUT:
%
%    t_1:         a changing time point
%
```

```
%    Delta_t_1:   changing Delta
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% compute_polynomial_intermediate_points %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CALLING SEQUENCE:
%
%  [u,uprime,u_inf,uprime_inf,u_sup,uprime_sup,U,Uprime]=,...
%   compute_polynomial_intermediate_points(t_low,t_1,t_up,...
%   a,b,c,d,a_inf,b_inf,c_inf,d_inf,a_sup,b_sup,c_sup,...
%   d_sup,A,B,C,D,dim)
%
%  SYNOPSIS:
%
%  compute_polynomial_intermediate_points.m computes cubic
%  Hermite polynomials for the system of ODEs and the
%  preconditioner and their derivatives at t_1 and t_up.
%
%   INPUT:
%
%   t_low:       starting time
%
%   t_1:         time variable at which polynomials are
%                evaluated.
%
%   t_up:        end time
%
%   a:           coefficient of the cubic term of Hermite
%                polynomial of the system of ODEs solved at
%                mid point u0 of the interval [yinf,ysup]
%
%   b:           coefficient of the quadratic term of Hermite
%                polynomial of the system of ODEs solved at
%                mid point u0
%
%   c:           coefficient of the linear term of Hermite
%                polynomial of the system of ODEs solved at
%                mid point u0
%
%   d:           constant coefficient of the cubic Hermite
%                polynomial of the system of ODEs solved at
%                mid point u0.
%
%   a_inf:       coefficient of the cubic term of Hermite
%                polynomial of the system of ODEs solved at
```

```
%                yinf
%
%   b_inf:       coefficient of the quadratic term of Hermite
%                polynomial of the system of ODEs solved at
%                yinf
%
%   c_inf:       coefficient of the linear term of Hermite
%                polynomial of the system of ODEs solved at
%                yinf
%
%   d_inf:       constant coefficient of the cubic Hermite
%                polynomial of the system of ODEs solved at
%                yinf
%
%   a_sup:       coefficient of the cubic term of Hermite
%                polynomial of the system of ODEs solved at
%                ysup
%
%   b_sup:       coefficient of the quadratic term of Hermite
%                polynomial of the system of ODEs solved at
%                ysup
%
%   c_sup:       coefficient of the linear term of Hermite
%                polynomial of the system of ODEs solved at
%                ysup
%
%   d_sup:       constant coefficient of the cubic Hermite
%                polynomial of the system of ODEs solved at
%                ysup
%
%   A:           coefficient of the cubic term of Hermite
%                polynomial of the system of preconditioner
%
%   B:           coefficient of the quadratic term of Hermite
%                polynomial of the system of preconditioner
%
%   C:           coefficient of the linear term of Hermite
%                polynomial of the system of preconditioner
%
%   D:           constant coefficient of the cubic Hermite
%                polynomial of the system of preconditioner
%
%   dim:         dimension of the system of ODEs
%
%
%  OUTPUT:
```

```
%
%  u_inf:          continuously differentiable solution at
%                  extreme points  of the interval [t_1,t_up]
%                  in the form of cubic Hermite polynomials for
%                  the system of ODEs solved at yinf
%
%   u:             continuously differentiable solution at
%                  extreme points of the interval [t_1,t_up]
%                  in the  form of cubic Hermite polynomials
%                  for the system of ODEs solved at the
%                  midpoint of initial box [yinf,ysup]
%
%   u_sup:         continuously differentiable solution at
%                  extreme points of the interval [t_1,t_up]
%                  in the form of cubic Hermite polynomials
%                  for the system of ODEs solved at ysup
%
%   U:             continuously differentiable solution for
%                   the system of preconditioner approximated
%                  at extreme points of the interval [t_1,t_up]
%                  in the form of cubic Hermite polynomials
%
%
%   uprime_inf:    First derivative of the cubic Hermite
%                  polynomial u_inf approximated at [t_1,t_up]
%
%   uprime:        First derivative of the cubic Hermite
%                  polynomial u approximated at [t_1,t_up]
%
%   uprime_sup:    First derivative of the cubic Hermite
%                  polynomial u_sup approximated at [t_1,t_up]
%
%   Uprime:        First derivative of the cubic Hermite
%                  polynomial U approximated at [t_1,t_up]
```

Bibliography

[1] E. Adams. *On methods for the construction of the boundaries of sets of solutions for differential equations or finite-dimensional approximations with input sets*. Springer Berlin, 1980.

[2] E. Auer and W. Luther. SmartMOBILE and its Applications to Guaranteed Modeling and Simulation of Mechanical Systems. *Informatics in Control, Automation and Robotics*, 24:139–151, 2009.

[3] E. Auer and W. Luther. Uses of New Sensitivity and Dae Solving Methods in SmartMobile for Verified Analysis of Mechanical Systems. *International Journal of Applied Mathematics and Computer Science*, 19(3):455–467, 2009.

[4] E. Auer, A. Rauh, E. P.Hofer, and W. Luther. *Validated Modeling of Mechanical Systems with SmartMOBILE : Improvement of Performance by ValEncIA-IVP*, volume 5045 of *Lecture Notes in Computer Science*. Springer, 2008.

[5] R. Barrio. Performance of the Taylor series method for ODEs/DAEs . *Applied Mathematics and Computation*, 163(2):525–545, 2005.

[6] D. Barton, I. M. Willers, and R. V. M. Zahar. The automatic solution of the system of ODEs by the method of Taylor series. *The Comp. Journal*, 14(3), 1970.

[7] D.P. Bertsekas and I.B. Rhodes. Recursive state estimation for a set-membership description of uncertainty. volume 16, pages 117–128, 1968.

[8] M. Berz. COSY INFINITY version 8 referenced manual. East Lansing, Mich., 1997.

[9] M. Berz and G. Hoffstätter. Exact bounds on the long term stability of weakly nonlinear systems applied to the design of large storage rings. *Interval Computations*, 2:68–89, 1994.

[10] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on higher-order Taylor models. *Reliable Computing*, 4:361–369, 1998.

[11] A. Björck. *Numerical Methods for Least Square Problems*. Siam, 1996.

[12] J. A. Braun and R. E. Moore. A program for the solution of differential equations using interval arithmetic (difeq) for the cdc 3600 and 1604, 1968. MRC Technical Summary 901, University of Wisconsin, Madison.

[13] Ole Stauning C. Bendtsen. FADBAD a Flexible C++ package for Automatic Differentiation. Technical report, Department of Mathematical Modeling, Technical University of Denmark, DK-2800, Lyngby, Denmark, 1996.

[14] Ole Stauning C. Bendtsen. TADIFF, a flexible C++ package for Automatic Differentiation using Taylor series. Technical report, Department of Mathematical Modeling, Technical University of Denmark, DK-2800, Lyngby, Denmark, 1997.

[15] F. F . Cap. *Mathematical Methods in Physics and Engineering with Mathematica*. CRC Press, 2003.

[16] F. L. Chernousko. *State Estimation for Dynamic System*. CRC Press, 1993.

[17] F. L. Chernousko. *What is ellipsoidal modeling and how to use it for control and state estimation?* Springer, Wien, 1999.

[18] F. L. Chernousko. Optimal ellipsoidal estimation of dynamic systems subject to uncertain disturbances. *Cybernetics and Systems Analysis*, 38(2):85–89, 2002.

[19] F. L. Chernousko. Ellipsoidal state estimation for dynamical systems, 2005. Invited Talks from the Fourth World Congress of Nonlinear Analysts (WCNA 2004).

[20] F. L. Chernousko. Properties of optimal ellipsoids approximating reachable sets of uncertain systems. *Mathematical and Computer Modeling of Dynamical Systems: Methods, Tools and Applications in Engineering and Related Sciences*, 11:135–147, 2005.

[21] K. D. Clark and L. R. Petzold. Numerical solution of boundary value problems in differential-algebraic systems. *SIAM J. ScI. STAT. COMPUT.*, 10(5):915–938, 1989.

[22] G. Corliss and Y. F. Chang. Solving ODEs using Taylor series. *ACM Transactions on Mathematical Software*, 8(2):114–144, 1982.

[23] G. F. Corliss. *Where is validated ODE solving going?*, volume 89 of *Mathematical Research*. Akademic Verlag Berlin, 1996.

[24] George F. Corliss and Robert Rihm. Validating an a priori enclosure using high-order Taylor series. In *In Scientific Computing, Computer Arithmetic, and Validated Numerics*, pages 228–238. Akademie Verlag, 1996.

[25] S. Dietrich. *Adaptive Verifizierte Lösung gewoöhnlicher Differentialgleichungen*. Universität Karlsruhe (TH), Germany, 2003. PhD Thesis.

[26] P. Eijgenraam. *The solution of initial value problems using interval arithmetic*. PhD thesis, Stichting Mathematisch Centrum, Amsterdam, 1981.

[27] G. Fábián, D.A. van Beek, and J.E. Rooda. Substitute equations for index reduction and discontinuity handling. Vienna, 2000.

[28] C. F. Van Loan G. H. Golub. *Matrix computations*. JHU Press, 1996.

[29] F. R. Gantmacher. *The Theory of Matrices*. Chelsea Publishing Company, NewYork, 1960.

[30] C. W. Gear and L. R . Petzold. Ode methods for the solution of differential/algebraic systems. *SIAM Journal on Numerical Analysis*, 21(4):716–728, 1984.

[31] T. H. Gronwall. Note on the derivatives with respect to a parameter of the solutions of a system of differential equations. *Annals of Mathematics*, 20:292–296, 1919.

[32] J. Hoefkens, M. Berz, and K. Makino. *Efficient Higher-Order Methods for ODEs and DAEs*. Springer, 2002.

[33] *http : //en.wikipedia.org/wiki/Interval_arithmetic#Dependency_problem*.

[34] S. Hunger. Intervallanalytische Defektabsschätzung bei Anfangswertaufgaben für Systeme gewöhnlicher Differentialgleichungen,. *Schriften der Gesellschaft fur Mathematik und Datenverarbeitung, Bonn*, 14, 1971.

[35] L. W. Jackson. Interval arithmetic error-bounding algorithms. *Siam Journal of Numerical Analysis*, 12(2):223–238, 1975.

[36] E. Kamke. *Differentialgleichungen reeller Funktionen*. Leipzig: Akademische Verlagsgesellschaft,, 1945.

[37] O. Knüppel. PROFIL/BIAS a fast interval library. *Computing*, 53:277–287, 1994.

[38] D. E. Knuth. Literate programming. Technical report, 1983.

[39] D. E. Knuth. Literate programming. 1992.

[40] N. N. Krasovsky. Theory of control of motion, 1968. Nauka Moscow.

[41] N. N. Krasovsky. Game problems of meeting of motion, 1970. Nauka Moscow.

[42] F. Krückberg. *Ordinary differential equations*. Clarendon Press, Oxford, 1969.

[43] W. Kuhn. Rigorous error bounds for IVP based on defect estimates, 1997. -.

[44] A.B. Kurzhanski. Control and observation in conditions of uncertainty, 1977. Nauka, Moscow.

[45] K. l. Nickel. Bounds for the set of solutions of functional-differential equations. Technical report, 1983.

[46] V. Lakshmikantham and S. Leela. *Differential and Integral Inequalities: Theory and Applications*, volume 55. New York Academic Press, 1969.

[47] E. B. Lee and L. Markus. *Foundations of optimal control theory*. Wiley New York, 1967.

[48] Y. Lin, J. A. Enszer, and M. A. Stadtherr. Enclosing all solutions of two-point boundary value problems for ODEs. *Computers and Chemical Engineering*, 32:1714–1725, 2008.

[49] Y. Lin and M. A. Stadther. Deterministic global optimization for parameter estimation of dynamical systems. *Ind and Eng Chem Res*, 45(25):8438–8448, 2006.

[50] Y. Lin and M. A. Stadther. Validated solution of ODEs with parametric uncertainties. *Computer-Aided Chem. Engrg*, 21:167–172, 2006.

[51] R. J. Lohner. Enclosing the solution of ordinary initial and boundary value problems. *Computer Arithmetic: Scientific Computation and Programming Languages*, pages 255–286, March 28-April 1 1987.

[52] R. J. Lohner. *Computations of guaranteed enclosures for the solutions of ordinary initial and boundary value problems*. Clarendon Press, Oxford, UK, 1992.

[53] R. J. Lohner. *Step size and order control in the verified solution of IVP with ODEs*. Stanford University, Calif., March 28-April 1 1995.

[54] R.J. Lohner. *Computation of guaranteed enclosures for the solution of ordinary initial and boundary value problems*. Clarendon Press Oxford, 1992.

[55] K. Makino and M. Berz. COSY INFINITY version 8. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 427:338–343, 1999.

[56] K. Makino and M. Berz. *Taylor model based verified integration for the Volterra equations and the Lorenz system*, May 23-25 2002. SIAM Workshop on Validated Computing Toronto.

[57] K. Makino and M. Berz. COSY INFINITY version 9. In *Proceedings of the 8th International Computational Accelerator Physics Conference - ICAP 2004*, volume 558, pages 346–350, 2006.

[58] M. E. Miller. Interval arithmetic programs and references, 1965. Lockheed Missiles and Space Div. Memo.

[59] R. E. Moore. Automatic error analysis in digital computation, 1959. Technical Report Space Div. Report LMSD84821, Lockheed Missiles and Space Co., Sunnyvale, CA, USA.

[60] R. E. Moore. Interval arithmetic and automatic error analysis in digital computing., 1962. Ph.D. dissertation, Department of Mathematics, Stanford University, Stanford, CA, USA.

[61] R. E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers. *Error in Digital Computation*, 1:61–130, 1965.

[62] R. E. Moore. Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solution of ordinary differential equations. *Error in Digital Computation*, 2:103–140, 1965. Proceedings of a symposium conducted by the Mathematics Research Center, United States Army, at the University of Wisconsin, Madison, April 2628, 1965.

[63] R. E. Moore. Interval analysis. Englewood Cliffs, N. J., 1966.

[64] R. E. Moore. A survey of interval methods for differential equations. pages 1529–1535, Las Vegas, 1984. In Proceedings of the 23rd Conference on Decision an dControl.

[65] R. E. Moore. Numerical solution of differential equations to prescribed accuracy. *Computers and Mathematics with Applications*, 28:253–261, 1994.

[66] R. E. Moore, J. A. Davison, H. R. Jaschke, and S. Shayer. Difeq integration routine - user's manual, 1964. Technical Report LMSC6-90-64-6, Lockheed Missiles and Space Co.

[67] N. S. Nedialkov. VNODE-LP A Validated Solver for Initial Value Problems in Ordinary Differential Equations. Technical report, Department of Computing and Software, McMaster University, Hamilton, Canada,, 2006.

[68] N. S. Nedialkov and K. R. Jackson. *A new perspective on the wrapping effect in interval methods for initial value problems for ordinary differential equations*. Springer-Verlag, Vienna, 2001.

[69] N. S. Nedialkov and K. R. Jackson. The design and implementation of a validated object-oriented solver for IVPs for ODEs. Technical report, Software Quality Research Laboratory, Department of Computing and Software, McMaster University, Hamilton, Canada, L8S 4K1, 2002.

[70] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7:449–465, 2001.

[71] Ned Nedialkov. Interval Tools for ODEs and DAEs. IEEE Computer Society, Los Alamitos (2007), Duisburg, Germany, 2006.

[72] N.S. Nedialkov. Interval method for initial value problems in linear ODEs. Supported in part by the Natural Sciences and Engineering Research Council of Canada.

[73] N.S. Nedialkov. *Computing Rigorous Bounds on the Solution of an IVP for an ODE*. PhD thesis, University of Toronto, 1999.

[74] N.S. Nedialkov and K.R. Jackson. The design and implementation of an object-oriented validated ode solver.

[75] N.S. Nedialkov and K.R. Jackson. An Interval Hermite-Obreschkoff method for computing rigorous bounds on the solution of an IVP for an ODE. *Reliable Computing*, 5(3):289–310, 1999.

[76] N.S. Nedialkov, K.R. Jackson, and J.D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7(6):449–465, December 2001.

[77] M. Neher. Enclosing power series solution of ODEs. *Z. angew. Math. Mech.*, 77, 1992.

[78] M. Neher. Ein Einschliessungsverfahren fur $y^n = \Sigma_{i=0}{}^{n-1} p_i(x) y^i + p(x)$. *Z. angew. Math. Mech.*, 75:47–48, 1995.

[79] M. Neher. An enclosure method for linear ODEs with analytical coefficients. *Z. angew. Math. Mech.*, 76, 1996.

[80] M. Neher. From interval analysis to Taylor Models - an overview. 2005. In Proc. IMACS'05, 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation.

[81] M. Neher, K.R. Jackson, and N.S. Nedialkov. On Taylor model based integration of ODEs. 2005.

[82] R. D. Neidinger. Differential equations are recurrence relations in apl. *ACM SIGAPL APL Quote Quad*, 23:165 – 174, 1992.

[83] A. Neumaier. Global, rigorous and realistic bounds for the solution of dissipative differential equations. Part I: Theory. *Computing*, 52:315–336, 1994.

[84] A. Neumaier. Taylor forms - use and limits. *Reliable Computing*, 9:43–79, 2002.

[85] B. G. Pachpatte. *Inequalities for Differential and Integral Equations*. Academic Press, 1998.

[86] M. Plum. Eigenvalue inclusions for second-order ordinary differential operators by a numerical homotopy method. *ZAMP*, 41:205–226, 1990.

[87] M. Plum. *Existence and inclusion for two point boundary value problems by numerical means*. Computational Ordinary Differential Equations. Clarendon Press, Oxford, 1992.

[88] J. D. Pryce. Solving high-index DAEs by Taylor series. *Numerical Algorithms*, 19:195–211, 1998.

[89] J. D. Pryce. A simple structural analysis method for DAEs. *BIT*, 41(2):364–394, 2001.

[90] A. Rauh, E. Auer, and E. P. Hofer. A novel interval method for validating state enclosures of the solution of initial value problems. volume 19, pages 381–397. Versita, Warsaw.

[91] A. Rauh, E. Auer, J. Minisini, and Eberhard P. Hofer. Extensions of ValEncIA-IVP for Reduction of Overestimation, for Simulation of Differential Algebraic Systems, and for Dynamical Optimization. *PAMM*, 7:1023001–1023002, 2007.

[92] A. Rauh, E. P. Hofer, and E. Auer. VALENCIA-IVP: A comparison with other IVP solvers.

[93] A. Reiter. Interval arithmetic package (INTERVAL), 1965. Technical Report, Univ. of Wisconsin Mathematics Research Center.

[94] A. Reiter. Programming interval arithmetic and applications. 1967. In Proceedings of the 1967 Army Numerical Analysis Conference, ARO-D Report 67-3.

[95] F. M. Schlaepfer and F.C. Schweppe. Continuous time state estimation under disturbances bounded by convex sets. volume 17, pages 197–205, 1972.

[96] S. Schulz. Four Lectures on Differential-Algebraic Equations, 2003. Report Series 497.

[97] F.C. Schweppe. Recursive state estimation: unknown but bounded errors and system inputs. volume 13, pages 22–28, 1968.

[98] F.C. Schweppe. *Uncertain Dynamic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[99] A.B. Singer and P.I. Barton. Bounding the solutions of parameter dependent nonlinear ordinary differential equations. *SIAM J. Sci. Comput*, 27:2167–2182, 2006.

[100] Robert D. Skeel. Thirteen Ways to Estimate Global Error. *Numer. Math*, 48:1–20, 1986.

[101] O. Stauning. Enclosing solutions of ordinary differential equations: with application. Technical report, Department of Mathematical Modeling, Technical University of Denmark, DK-2800, Lyngby, Denmark, 1996.

[102] O. Stauning. Automatic validation of numerical solutions, September 1997. -.

[103] G. Strang. *Introduction to Linear Algebra*. Third edition. Wellesley-Cambridge Press, 2003.

[104] W. Tucker. *The Lorenz Attractor Exists*. PhD thesis, Uppsala , Sweden, 1998. -.

[105] W. Tucker. A rigorous ODE solver and Smale's 14th problem. *Found. Comput.Math*, 2, 2002.

[106] W. Tucker. Computing accurate Poincaré maps, 2002. -.

[107] W. Walter. *Differential and Integral Inequalities*. Springer Verlag Berlin Heidelberg New York, Karlsruhe, August 1970.

[108] L. T. Watson. An algorithm that is globally convergent with probability one for a class of nonlinear two-point boundary value problem. *SIAM Journal on Numerical Analysis*, 16(3):394–401, 1979.

Curriculum Vitæ

Qaisra Fazal

qaisra.fazal@univie.ac.at

Professional Experience

- **Duration:** June 2002 - April 2006.
  Position: Lecturer.
  Organization: Department of Mathematics, GC. University Lahore, Pakistan.

- **Duration:** Sep. 2003 - Aug. 2005.
  Position: Visiting Lecturer.
  Organization: National University of Computer and Emerging Sciences, Lahore, Pakistan.

Education

- Master of Philosophy (M.Phil) in Computational Mathematics from University of Punjab, Lahore Pakistan (2003).

- Master of Science (M.SC) in Computational Mathematics from Bahauddin Zakariya University, Multan, Pakistan (1998).

- Bachelor of Science (B.SC) in Math. Major and Physics from Bahauddin Zakariya University, Multan, Pakistan (1995).

Publications

- *Optimization Techniques for Error Bounds of ODEs*, PhD dissertation in preparation under the supervision of Prof. Arnold Neumaier, Institute for Mathematics, University of Vienna, Austria, 2011.

- Error Bounds for Initial Value Problems by Optimization, (with A. Neumaier), in preparation.
- Reconstruction of Conic Sections using Rational Spline, M.Phil Dissertation, under the supervision of Prof. Shahid S. Siddiqi, University of Punjab, Lahore Pakistan, 2003.

Research Interest

- Numerical Methods Applied to Differential Equations
- Mathematical Modeling of Applied Problems.
- Numerical Linear Algebra,
- Numerical Algorithms for optimization.
- Spline Theory

Personal Information

Date of Birth	November 10, 1973
Nationality	Pakistani
Academic Grad	Mag.

Declaration

The information furnished in this document are true to my knowledge as on June 27, 2011.

Qaisra Fazal.