Rigorous defect control and the numerical solution of ordinary differential equations

RIGOROUS DEFECT CONTROL AND THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

BY

JOHN M. ERNSTHAUSEN

A THESIS

SUBMITTED TO THE SCHOOL OF COMPUTATIONAL SCIENCE AND ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MSC OF COMPUTATIONAL SCIENCE AND ENGINEERING

MSc of Computational Science and Engineering(2017) McMaster University

(School of Computational Science and Engineering) Hamilton, Ontario, Canada

TITLE:	Rigorous defect control and the numerical solution of ordinary differential equations
AUTHOR:	John M. Ernsthausen School of Computational Science and Engineering McMaster University, Hamilton, Ontario, Canada
SUPERVISOR:	Nedialko S. Nedialkov
NUMBER OF PAGES:	xv, 134

Dedication

This thesis is dedicated to software craftsmanship and understanding through computer experiments.

Abstract

Modern numerical ordinary differential equation initial-value problem (ODE-IVP) solvers compute a piecewise polynomial approximate solution to the mathematical problem. Evaluating the mathematical problem at this approximate solution defines the defect. Corless and Corliss proposed rigorous defect control of numerical ODE-IVP.

This thesis automates rigorous defect control for explicit, first-order, nonlinear ODE-IVP. Defect control is residual-based backward error analysis for ODE, a special case of Wilkinson's backward error analysis. This thesis describes a complete software implementation of the Corless and Corliss algorithm and extensive numerical studies. Basic time-stepping software is adapted to defect control and implemented.

Advances in software developed for validated computing applications and advances in programming languages supporting operator overloading enable the computation of a tight rigorous enclosure of the defect evaluated at the approximate solution with Taylor models. Rigorously bounding a norm of the defect, the Corless and Corliss algorithm controls to mathematical certainty the norm of the defect to be less than a user specified tolerance over the integration interval. The validated computing software used in this thesis happens to compute a rigorous supremum norm.

The defect of an approximate solution to the mathematical problem is associated with a new problem, the perturbed reference problem. This approximate solution is

often the product of a numerical procedure. Nonetheless, it solves exactly the new problem including all errors. Defect control accepts the approximate solution whenever the sup-norm of the defect is less than a user specified tolerance. A user must be satisfied that the new problem is an acceptable model.

Acknowledgements

I extend my gratitude and appreciation with a few professional acknowledgements. I owe the success of my Masters of Science project to Prof. Ned Nedialkov, my mentor and supervisor. Thank you for believing in me. I would also like to thank my thesis committee Prof. Jacques Carette (Chair) and Prof. David Earn whose comments and suggestions have improved the presentation of this document. Our Computational Science and Engineering graduate director Bartek Protas with our administrative assistants Tina Thorogood and Diana Holmes helped me to navigate the requirements. I appreciate that Gary Guangning Tan shared his thesis template and his time in deep conversation about our academic projects. Shawn X. Li and I engaged the real world in a software development internship at the McMaster University spin-off EnviroSim Associates Ltd. I appreciate Shawn's technical insight, pragmatic approach, and help in my first build of SOLLYA. I am grateful to Reza Zolfaghari for his comments and suggestions on a draft of this thesis.

Rob Corless presented a seminar talk, "Optimal Backward Error and the Leaky Bucket", at McMaster University on 8 October 2015. This talk inspired my thesis topic.

I am grateful to the government of Canada for the privilege to study in Canada. I appreciate the hospitality of the Canadian people.

I have a couple of personal acknowledgements. I'm grateful to Jaleen Grove for a super environment for academic work. Thank you for opening your home to me. I'm grateful to Michelle Fielding. I couldn't overstate all you have done for me. Thank you Michelle. I pray manifold blessings return to you!

Notation and abbreviations

Notation

u	Approximate solution
$C^k\big(\Omega, \mathbb{R}^d\big)$	k-times, continuously differentiable, mappings from domain Ω into range $\mathbb{R}^d$
Δu	Defect evaluated using approximate solution u
δu	Deviation evaluated using approximate solution u
$(p, \boldsymbol{r})$	Taylor model of degree k
	with Taylor polynomial p of degree k and interval remainder bound $\boldsymbol{r}$
t_0	Initial integration time
t_{end}	Final integration time
TOL	Tolerance
$u'\big(t\big)$	First derivative of u with respect to t
$\frac{d}{dt}u\big(t\big)$	First derivative of u with respect to t
$\mathbb{R}$	The field of real numbers
$\mathbb{R}^d$	The d-dimensional normed linear (vector) space over the field of real numbers
$\|v\|$	The norm of a vector $v \in \mathbb{R}^d$
$\|v\|_{\mathcal{J}}$	The supremum norm of a mapping $t \in \mathcal{J} \mapsto v(t) \in \mathbb{R}$
$\|v\|_{\mathcal{J},\infty}$	The infinity norm of a mapping $t \in \mathcal{J} \mapsto v(t) \in \mathbb{R}^d$

δ	A rigorous upper bound on $\|v\|_{\mathcal{J},\infty}$
$\boldsymbol{r} \stackrel{\text{def}}{=} [\,\underline{r}, \overline{r}\,]$	An interval
$\underline{r}$	Left endpoint of an interval $\boldsymbol{r} \stackrel{\text{def}}{=} [\,\underline{r}, \overline{r}\,]$
$\overline{r}$	Right endpoint of an interval $\boldsymbol{r} \stackrel{\text{def}}{=} [\,\underline{r}, \overline{r}\,]$

Acronyms and Abbreviations

ADAMS	ADAMS method
API	Application Programming Interface
CRK	Continuous Runge–Kutta
C++	Object oriented C language
DAE	Differential Algebraic Equation
multistep	Linear multistep Method
ODE	Ordinary Differential Equation
ODE-IVP	Ordinary Differential Equation Initial-Value Problem
PID	Proportional-Integral-Derivative
RDC CRK	Relaxed Defect Control
Runge–Kutta	Runge–Kutta method
RPA	Rigorous Polynomial Approximation
SDC CRK	Strict Defect Control
SDCV CRK	Strict Defect Control with Validity check
TDD	Test Driven Development
Taylor series	Taylor series method
UML	Unified Modelling Language

Software and Problem Solving Environments

FADBAD++	Automatic differentiation package [5, 92]
INTLAB	The INTLAB package [83]
MATLAB	The MATLAB problem solving environment [63]
ODETS	Numerical ODE-IVP solver developed in this thesis
SOLLYA	The SOLLYA package [12, 13]
TADIFF	Automatic differentiation package [6]

Contents

Abstract **v**

Acknowledgements **vii**

Notation and abbreviations **ix**

1 Introduction **1**

2 Background **15**

2.1 The mathematical ODE-IVP problem 16

2.2 Residual-based backward error analysis 19

2.3 Defect control 22

2.4 Backward error analysis 25

2.5 Asymptotic defect control 29

2.5.1 The asymptotic defect control problem 30

2.5.2 Discrete Runge–Kutta methods 31

2.5.3 Continuous Runge–Kutta methods 32

2.5.4 Constructing continuous Runge–Kutta methods I 33

2.5.5 Constructing continuous Runge–Kutta methods II 35

2.6 Guaranteed defect control 42
2.7 Stepsize control 45
2.8 Global error and condition 48

3 Taylor models **53**

3.1 Interval arithmetic 56
3.2 Taylor models 64
3.3 Computing the supremum norm 66
3.4 Class Tmodel 67

4 Automating rigorous defect control **75**

4.1 Input and driver 76
4.2 Phase I: Compute an approximate solution 76
4.3 Phase II: Bound the defect 80
4.4 Phase III: Accept/reject step 84
4.5 Initial stepsize 85
4.6 ODETS software 86
4.6.1 Build requirements 87
4.6.2 Requirements and Specification by Example 87
4.6.3 Algorithm overview 88
4.6.4 The class structure of ODETS 91

5 Numerical results and discussions **95**

5.1 Test cases 95
5.2 Adaptive time-stepping study 96
5.3 Performance study 106

5.4 Defect width study 108
5.5 Initial value problems solved 110
5.5.1 Three DETEST examples 111
5.5.2 Restricted two-body model 111
5.5.3 Forced Brusselator 112
5.5.4 Discussion 113

6 Conclusions and future work **115**
6.1 Conclusions 115
6.2 Future work 119

Chapter 1

Introduction

We investigate rigorous defect control for explicit, first-order, nonlinear ordinary differential equation initial-value problem (ODE-IVP) from its mathematical foundation to its complete software implementation. This thesis is a software development project that automates rigorous defect control for the numerical solution of ODE-IVP, a stepsize control strategy based on the defect. We document an extensive battery of numerical studies designed to exhaustively test our software and record the results from these studies. The research outcomes identify stepsize control as a future need.

Corless and Corliss proposed rigorous defect control of ODE-IVP over 25 years ago [17]. Defect control is residual-based backward error analysis for ODE, a special case of Wilkinson's backward error analysis [18], which we shall explain in Chapter 2. Residual-based backward error analysis is a remarkably straightforward procedure and a powerful point of view.

Modern numerical ODE-IVP solvers compute a piecewise polynomial *approximate*

solution [34, 40, 85]

$$t \in [t_0, t_{\text{end}}] \mapsto u\big(t\big) = \big(u_1\big(t\big), \ldots, u_d\big(t\big)\big) \in \mathbb{R}^d \tag{1.1}$$

to the mathematical ODE-IVP or *reference problem*

$$x'\big(t\big) = f\big(t, x\big(t\big)\big), \quad x\big(t_0\big) = x_0 \in \mathbb{R}^d, \quad t \in [t_0, t_{\text{end}}]. \tag{1.2}$$

As the notation suggests, the computed approximate solution is globally continuously differentiable. Compute at this approximate solution

$$\Delta u\big(t\big) \stackrel{\text{def}}{=} u'\big(t\big) - f\big(t, u\big(t\big)\big), \quad \Delta u\big(t_0\big) \stackrel{\text{def}}{=} u\big(t_0\big) - x_0, \quad t \in [t_0, t_{\text{end}}]. \tag{1.3}$$

We call a continuous Δu a *defect*. Whenever Δu is only piecewise smooth, we call it a *deviation*, and we distinguish it from the continuous defect with the notation δu. The mathematical notion of defect can be traced back at least to Cauchy [18, p. 509].

For the sake of concreteness, we use $\|\cdot\|_\infty$ to represent the infinity norm for vectors in $\mathbb{R}^d$ and the induced matrix norm for matrices throughout this thesis. For the sake of mathematical completeness, we use the infinity norm in functional spaces. The infinity norm of a continuous path $t \in \mathcal{J} \mapsto x\big(t\big) = \big(x_1\big(t\big), \ldots, x_d\big(t\big)\big) \in \mathbb{R}^d$ over a closed, bounded interval $\mathcal{J} \subset \mathbb{R}$ is

$$\|x\|_{\mathcal{J},\infty} \stackrel{\text{def}}{=} \max_{i=1,\ldots,d} \max_{t \in \mathcal{J}} |x_i(t)|. \tag{1.4}$$

We may write the norm (1.4) as $\|x\|_\infty$ or simply $\|x\|$ whenever identifying the underlying vector space is clear from the context.

Let's consider *defect control*. Let TOL be a given user specified tolerance. Locally, a rigorous defect control approach selects a stepsize such that the sup-norm of the defect over the step is bounded by TOL. This approach introduces a mesh to partition the interval $[t_0, t_{\text{end}}]$

$$t_0 < t_1 \cdots < t_N = t_{\text{end}} \tag{1.5}$$

and generates a discrete approximation $x_n = u(t_n)$ at each associated mesh point in (1.5). We refer to the sequence of *continuation points* (t_n, x_n) as a *skeletal solution*,

$$\big\{\big(t_n, x_n\big)\big\}_{n=0}^{N}, \qquad t_0 < t_1 < \ldots < t_N = t_{\text{end}}. \tag{1.6}$$

At a continuation point (t_n, x_n), the algorithm, as yet to be specified, computes an approximate solution u to the local numerical ODE-IVP

$$u'\big(t\big) = f\big(t, u\big(t\big)\big), \quad u\big(t_n\big) = x_n, \quad \|\Delta u\|_{[t_n, t_{n+1}],\infty} \leq \text{TOL}. \tag{1.7}$$

A *local perturbed reference problem* is a perturbation of the reference problem by the defect at the approximate solution,

$$u'\big(t\big) = f\big(t, u\big(t\big)\big) + \Delta u\big(t\big), \quad u\big(t_n\big) = x_n + \Delta u\big(t_n\big), \quad t \in [t_n, t_{n+1}] \tag{1.8}$$

that associates the local defect with the local problem (1.7). The local approximate solution is the exact solution of the perturbed reference problem (1.8), even if it is tainted with representation and approximation errors.

A reference problem, a perturbed reference problem, and an engineered problem

are concepts from backward error analysis which we shall explain in Chapter 2.

Rigorous defect control identifies a solution so that $\|\Delta u\|_{[t_n,t_{n+1}],\infty} \leq \text{TOL}$ holds to mathematical certainty. Consequently, concatenating this sequence of local approximate solutions into the continuous numerical solution (1.1) joined at the skeletal points (1.6), the approximate solution solves exactly the global numerical ODE-IVP

$$u'\big(t\big)= f\big(t,u\big(t\big)\big)+\Delta u\big(t\big), \quad u\big(t_0\big)= x_0+\Delta u\big(t_0\big), \quad \|\Delta u\|_{[t_0,t_{\text{end}}],\infty} \leq \text{TOL}. \tag{1.9}$$

In rigorous defect control, the norm of the defect is controlled rigorously to be within a user specified tolerance. The user must validate that the perturbed reference problem is a nearby problem. A nearby problem by definition is as valid an ODE-IVP model as the reference problem, a model that is usually an approximation in modelling space anyhow.

Evaluating the reference problem (1.2) at the approximate solution u defines the defect (1.3) and the perturbed reference problem (1.9). While we never construct the perturbed reference problem in practice, we know its exact solution u.

The mathematical community often thinks of the mathematical reference problem as determined. However, the mathematical model may in fact denote an approximation to the reality it seeks to model. Backward error analysis relaxes the requirement that the reference problem be solved, and it instead solves exactly a nearby perturbed reference problem. Solving the engineered problem provides the exact solution of the perturbed reference problem. If the defect is smaller than the perturbations inherent in the modelling context, then the solution provided by the engineered problem can be considered completely satisfactory [18].

Defect control will control a measure of the defect to be less than TOL so that

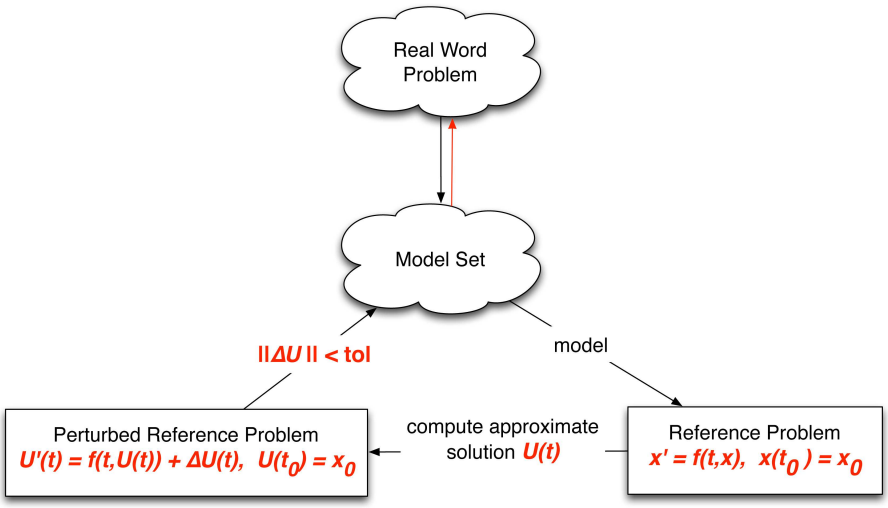


Figure 1.1: Justification for the validity of the perturbed reference problem.

defect control is residual based backward error analysis. We illustrate this situation in Figure 1.1.

From a practical point of view, MATLAB solvers ODE45 and ODE113 provide numerical solutions to ODE-IVP without concern about the defect. The local forward error is computed differently for the Runge–Kutta code ODE45 than for the ADAMS code ODE113. The defect easily distinguishes the quality of the numerical solutions. For example, consider the logistic model.

Logistic model: We applied ODE45 and ODE113 to the logistic model

$$x'(t) = x(t) - x(t)x(t), \quad x(0) = 0.2, \quad t \in [0, 5]. \tag{1.10}$$

We solve and plot the ODE45 numerical solution of (1.10) with MATLAB code

```
f = @(t,x) x - x.*x;
options = odeset( 'Reltol', 1.0e-3, 'Abstol', 1.0e-6 );
sol = ode45( f, [0,5], 0.2, options );
t = RefineMesh( sol.x, 15 );
[ z, dotz ] = deval( sol, t );
figure(1),plot(t,z)
```

We solve and plot the ODE113 numerical solution of (1.10) with MATLAB code

```
f = @(t,x) x - x.*x;
options = odeset( 'Reltol', 1.0e-3, 'Abstol', 1.0e-6 );
sol = ode113( f, [0,5], 0.2, options );
t = RefineMesh( sol.x, 15 );
[ z, dotz ] = deval( sol, t );
figure(1),plot(t,z)
```

The plots agree to graphical accuracy, the overlay is not shown. These MATLAB codes follow Corless and Fillion [18, Section 12.2] and method RefineMesh from Corless and Fillion [18, Section 12.2.2] inserts additional evaluation points between the continuation points.

```
function [ refinedMesh ] = RefineMesh( coarseMesh, nRefine )
  if nargin == 1
    nRefine = 4;
  end
  n = length( coarseMesh );
  [m1, m2] = size( coarseMesh );
  h = diff( coarseMesh );
  refinedMesh = repmat( coarseMesh(1:end-1).', 1, nRefine );
  refinedMesh = (refinedMesh+(h.')*[0:nRefine-1]/nRefine).';
  refinedMesh = [refinedMesh(:);coarseMesh(end)];
  if m1<m2
    refinedMesh = refinedMesh.';
  end
end
```

We compute the absolute defect for the logistic model:

```
f = @(t,x) x - x.*x;
options = odeset( 'Reltol', 1.0e-6, 'Abstol', 1.0e-11 );
sol = ode113( f, [0,5], 0.2, options );
t = RefineMesh( sol.x, 15 );
[ z, dotz ] = deval( sol, t );
deltaz = dotz - f( t, z );
figure(1),semilogy(t,abs(deltaz),'o:','MarkerSize',2,'MarkerFaceColor','r')
set(gca,'fontsize',16),axis([0, 5, 0.0, 1.0])
xlabel('t'),ylabel('absolute residual'),xticks([0:1:5])
```

The tolerances control the defect. We compute the relative defect by replacing deltaz in the MATLAB code with the relative defect.

```
deltaz = dotz./f( t, z ) - 1;
```

Plots indicate that ODE45 is less faithful than ODE113 when solving the logistic model, an assessment which is possible by comparing, in this case, deviations. The defect easily distinguishes the quality of the numerical solutions!

Defect plots indicate that the defect rendered for a general problem by MATLAB solvers ODE45 and ODE113 will not be continuous. Enright and Yan [38] construct continuous output Runge–Kutta (CRK) methods, and the defect at the approximate solution generated with these methods render a continuous defect, which we explain in Section 2.5.

Standard ODE-IVP solvers estimate the local forward error. Forward error estimation often depends on the solver as it is part of the solver's design, and the error estimate depends on the particular problem. Local forward error estimators can underestimate the true local error. The solver accepts this overshoot error estimate, because the error estimator indicates the estimate is within tolerance when the true error is

greater than tolerance. That's being fooled. Interval methods for ODE-IVP solvers compute rigorous bounds on the solution, but it is challenging to keep the bound tight [73, 75, 79]. Enright illustrates that replacing traditional error estimators with defect control separates concerns for universal comparison of approximate solutions from the engineered problems applied to compute them [24]. We offered this illustration on the logistic model.

Taylor models: Our approach to rigorous defect control uses Taylor models and interval arithmetic to compute a rigorous, tight upper and lower enclosure for the true, mathematical defect. Hence, defect control cannot be fooled. A Taylor model is an interval analysis technique based on Taylor arithmetic with remainder. Neumaier discusses the properties and merits of Taylor forms in his review article [80]. Taylor forms use the truncated Taylor series approximation. Taylor models implement Taylor forms in a special way that aims to reduce overestimation of the rigorous remainder bound [58, 67, 68, 82].

A *Taylor model of degree k* represents a function v as a couple $\big(p, \boldsymbol{r}\big)$:

$$\big(p, \boldsymbol{r}\big) \quad \text{means} \quad v(t) - p(t) \in \boldsymbol{r} \stackrel{\text{def}}{=} [\,\underline{r}, \overline{r}\,] \quad \text{for all} \quad t \in [a, b]. \tag{1.11}$$

The function $t \in [a, b] \mapsto v\big(t\big) \in \mathbb{R}$ is a member of the function class $C^{k+1}\big([a, b], \mathbb{R}\big)$. We select approximations $p \in \mathcal{P}$ to v from a $\big(k+1\big)$-dimensional subspace $\mathcal{P}$ of $C^{k+1}\big([a, b], \mathbb{R}\big)$, the polynomials of degree k. For example, in Figure 1.2, we depict the Taylor model $\big(p, \boldsymbol{r}\big)$ of degree 4 for $\sin(t)$ on $[0, \pi/2]$. Here $p(t) = t - 0.1667\ t^3$ is the Taylor series approximation to $\sin(t)$ at 0 of degree 4 and the remainder bound is $\boldsymbol{r} = [-0.0797, 0.0797]$. The graph and Taylor model are computed using INTLAB [83].

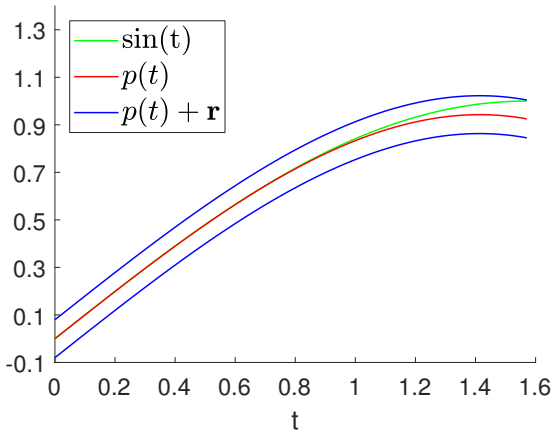


Figure 1.2: Taylor model of degree 4 for $\sin(t)$

The approximation statement $\sin(t) \in p(t) + r$ on $[0, \pi/2]$ in (1.11) holds, and it is observed graphically in Figure 1.2.

Joldes [61] generalized Taylor models to Rigorous Polynomial Approximation (RPA) and designed a new tool based on Chebyshev series approximations. The approximation subspace are the polynomials of degree k in a certain polynomial basis. The most common representation for a polynomial p is a monomial basis shifted by t_0; that is, $p(t) = \sum_{i=0}^{k} c_i \big(t - t_0\big)^i$. Other bases are available such as Newton, Bernstein, and Chebyshev. We apply her theoretical results, and we extend her freely available codes distributed in the SOLLYA package [12, 13]. The SOLLYA tool implements RPA in one independent variable, and it implements a rigorous supremum norm.

Defect control literature: Hull [59] and Stetter [93, 94] established conditions that guarantee the existence of a continuously differentiable interpolant u that satisfies a perturbed reference problem and the indirect asymptotic relationship

$$\|\Delta u\| \leq \sigma(t) \text{ TOL},$$

for some scalar mapping σ on $[t_0, t_{\text{end}}]$, depending on the method as well as the problem, such that $|\sigma| \leq c$ where c is independent of the problem and the method. For Stetter, the path u and the defect Δu may not be computable.

Stetter considers the asymptotic form of σ for interpolation schemes that can be associated with ADAMS method [45]. In Hanson and Enright [50], Stetter's analysis is extended to interpolation schemes used in existing software, and numerical results are presented to verify the Stetter σ condition is satisfied over a realistic range of tolerances, opening an investigation into defect control.

Enright's asymptotic defect control is an efficient and effective defect control strategy, and it happens to be the first serious numerical implementation of defect control [24]. Asymptotic defect control extends to delay differential equations [30, 31, 51, 87, 97], Volterra integro-differential equations [84], ODE boundary value problems [35, 63, 88], and differential-algebraic equations [66, 81].

Enright with his coworkers and students developed asymptotic defect control. The pioneering articles Hanson and Enright [50], Higham [53, 54, 55], Enright [23, 24, 25, 26, 29], Enright and Hayes [32], and Enright and Yan [38], support the statements in the previous paragraph, and they illustrate defect control designed for implementation into high performance computing software [33] and into problem solving environments [27, 28]. Enright focused on defect control based on the infinity-norm. Kierzenka and Shampine [63] developed an inexpensive estimate of the defect in 2-norm.

Enright starts with a discrete Runge–Kutta method and designs a CRK method for non-stiff problems [38] based on procedures for continuous extensions developed in [34, 40, 85]. A CRK method generates a continuously differentiable, piecewise polynomial approximate solution to the ODE-IVP that is defined for all values of the

independent variable in the range of interest. Enright constructs special CRK methods of order p that have local error of order $p+1$ where the maximizer for the maximum magnitude of the defect is computable ahead of time and problem independent. The CRK method and the estimate of the maximum magnitude of the defect are tightly connected. Classic textbooks [2, 45, 46] discuss the construction and justification of traditional discrete Runge–Kutta methods.

Enright's asymptotic defect control has a 30 year development history, and the author acknowledges that the explanation in Section 2.5 omits interesting aspects of that history such as a theoretical cost per step analysis and a cost comparison analysis for the CRK methods which Enright designed. Enright quantifies the reliability of CRK methods through several novel measures which our discussion omits. Enright developed asymptotic defect control for several general-purpose numerical methods in classes of ordinary differential equations such as differential-algebraic equations, Volterra integro-differential equations, delay differential equations, and ODE boundary value problems which we omit from our discussion. The interested reader is invited to investigate the cited literature. On the other hand, the discussion in Section 2.5 offers a detailed explanation of state-of-the-art asymptotic defect control. Enright and Yan [38] explain the construction of a general CRK method.

Stepsize control literature: Defect control requires stepsize control, and, in this thesis, basic time-stepping software is adapted to defect control and implemented. The Lund PID stepsize controller generalizes the familiar elementary stepsize controller [46, 90, 91]. Bergsma [7] investigated three stepsize controllers based on the radius of convergence for Taylor series methods.

Our approach: Our approximate solution begins as a Taylor expansion of degree k. We construct the Hermite approximate solution from the Taylor approximate solution, the interpolant of the solution and derivative at both endpoints of the integration interval. The Hermite interpolant is the exact solution of the perturbed reference problem associated with the defect of the Hermite interpolant. Thus, we locally know the Taylor model of degree $\big(k+2\big)$ for the approximate solution u, $\big(u, \mathbf{0}\big)$. We run the Taylor model for the Hermite approximate solution through the code list of the defect $u' - f\big(t, u\big(t\big)\big)$ in SOLLYA to compute the Taylor model of degree $\big(k+2\big)$ for its defect, this requires that the ODE right hand side $f \circ s \in C^{k+3}\big(\mathbb{R}, \mathbb{R}^d\big)$ where $s\big(t\big) = \big(t, u\big(t\big)\big)$. The real-valued coefficients of the approximation polynomial and the ODE right hand side must be represented in machine format $\mathcal{F}$, and representing the coefficients in machine format introduces rounding and truncation errors. Nonetheless, the Taylor model of the defect accounts for all errors.

Let's review the process. Rigorous defect control occurs over three phases:

Phase I. Compute a Hermite approximate solution based on the Taylor series solution which is necessary for the defect of the Hermite approximate solution to be continuous, a desirable property. This step involves floating-point arithmetic.

Phase II. Bound the defect of the Hermite approximate solution. This phase rigorously bounds the sup-norm of the defect, and it involves Taylor models which are based on interval arithmetic.

Phase III. Accept or reject the predicted step based on the sup-norm of the defect evaluated at the Hermite approximate solution less than TOL. This step involves only floating-point arithmetic.

A rejected step can reuse the Taylor series solution on the newly predicted step.

Thesis outline: Chapter 2 discusses background topics related to rigorous defect control including an exposition of the mathematical ODE-IVP, backward error analysis, asymptotic defect control, and the rationale for guaranteed ODE defect control proposed by Corless and Corliss. Chapter 3 introduces Taylor models. Each component of the defect is a mapping over a closed bounded interval into the real numbers, and we represent it as a Taylor model so that we can tightly and rigorously bound the sup-norm of the mapping. The SOLLYA package [12, 13] does the heavy lifting in our software implementation. Chapter 4 implements rigorous defect control. We discuss the algorithm in three phases. Chapter 5 discusses our numerical experiments. We hope our Taylor model software will be generally used. We chose our example problems because they challenge existing stepsize controllers. Chapter 6 concludes the thesis with a Conclusions and future work chapter. Several open problems are discussed in this thesis, including the need for establishing improved time-stepping software in the future. Finally, we provide a detailed Bibliography on the literature involving defect control.

Chapter 2

Background

This chapter collects basic definitions, theorems, and background material on defect control. The material is meant to help the reader understand the mathematics behind the algorithms implemented in our software. The reader may confidently omit this chapter on a first reading.

Defect control is residual-based backward error analysis, and, replacing traditional forward error with the defect, it uses the defect in stepsize control. Rigorous defect control captures all error information. In this chapter, we discuss two approaches to defect control, asymptotic defect control and guaranteed defect control.

The Gröbner-Alexeev nonlinear variation-of-constants formula [18, p. 537] relates the defect to the global error through the derivative of the true solution of the reference problem with respect to its initial condition along the computed exact solution of the perturbed reference problem, whenever the vector field has a continuous first derivative. The Gröbner-Alexeev formula enables us to justify faithfully ensuring a measure of the defect less than TOL for problems of large global error. With a small amount of extra computational effort, the global error and an estimate of the condition

number are accessible with defect control [29].

2.1 The mathematical ODE-IVP problem

A given mathematical ODE-IVP

$$x'(t) = f\big(t, x(t)\big), \quad x(t_0) = x_0, \quad t \in [t_0, t_{\text{end}}] \tag{2.1}$$

with $x \in \mathbb{R}^d$ is an example of a *reference problem*, the model problem.

A *k-path x over the interval* $[t_0, t_{end}]$ is a mapping $x \in C^k([t_0, t_{\text{end}}], \mathbb{R}^d)$; that is, x has d components x_i which are k-times, continuously differentiable, real-valued functions over the interval $[t_0, t_{\text{end}}]$. We interchangeably denote the first ordinary derivative of the path x by $x'(t)$ and by $\frac{d}{dt}x(t)$.

The *solution* of the reference problem (2.1) has one independent variable t, must be continuously differentiable on the domain of definition $[t_0, t_{\text{end}}]$, must take values in real d-dimensional space $\mathbb{R}^d$, must satisfy the differential equation, and must satisfy the initial condition at t_0. More succinctly, the solution x of an ODE initial-value problem is a 1-path over $[t_0, t_{\text{end}}]$ that satisfies the defferential equation and the initial condition. We require the existence and uniqueness of solution to build our arguments.

Theorem 1 (Existence and uniqueness of ODE-IVP solution). *Define the solution ball*

$$\mathcal{B} \stackrel{\text{def}}{=} \big\{ \big(t, x\big) \mid t^* \le t \le t^* + T, \quad \|x - x^*\|_\infty \le M \big\} \subset \mathbb{R} \times \mathbb{R}^d$$

for any positive, real constants T and M, and any fixed but arbitrary point $\big(t^*, x^*\big)$.

Restricting T and M, if necessary, suppose the vector field f is continuous on $\mathcal{B}$ and $\|f\|$ is bounded on $\mathcal{B}$ by A. Further, suppose f satisfies

$$\|f\big(t_1, x_1\big) - f\big(t_2, x_2\big)\| \leq L \, \|x_1 - x_2\| \quad \textit{for all} \quad \big(t_1, x_1\big), \ \big(t_2, x_2\big) \in \mathcal{B},$$

a Lipschitz condition in x uniformly in t near the arbitrary but fixed point $\big(t^*, x^*\big)$ with Lipschitz constant L. If $T - t^* \leq M/A$, then there exists a unique, continuously differentiable solution x of the local ODE-IVP reference problem

$$x'\big(t\big) = f\big(t, x(t)\big), \quad x\big(t^*\big) = x^*, \quad t \in [t^*, t^* + T] \tag{2.2}$$

such that $(t, x(t)) \in \mathcal{B}$ for all $t \in [t^*, t^* + T]$.

A statement of Theorem 1 and its proof can be found in [45, Lemma 7.1, p.32 and Theorem 7.3, pp. 33–34].

The following assumption is required to extend the local solution.

Assumption 2. *Suppose the assumptions of Theorem 1 hold at $\big(t_0, x_0\big)$. The local solution through x_0 at t_0 can be extended to the interval $[t_0, t_{end}]$.*

Under the assumptions of Theorem 1, the vector field is continuous and bounded on $\mathcal{B}$. If the local solution u^n to (2.2) evaluated at the final time $t^* + T$ satisfies $\big(t^* + T, u^n\big(t^* + T\big)\big) \in \mathcal{B}$, then the local solution can be continued to the right of $t^* + T$. Define $\big(t_n, x_n\big) \stackrel{\text{def}}{=} \big(t^* + T, u^n\big(t^* + T\big)\big)$. It is natural to call $\big(t_n, x_n\big)$ a *continuation point*. The global solution $t \in [t_0, t_{\text{end}}] \mapsto u(t) \in \mathbb{R}^d$ is a *pasting* or *continuation* of the local solutions u^n.

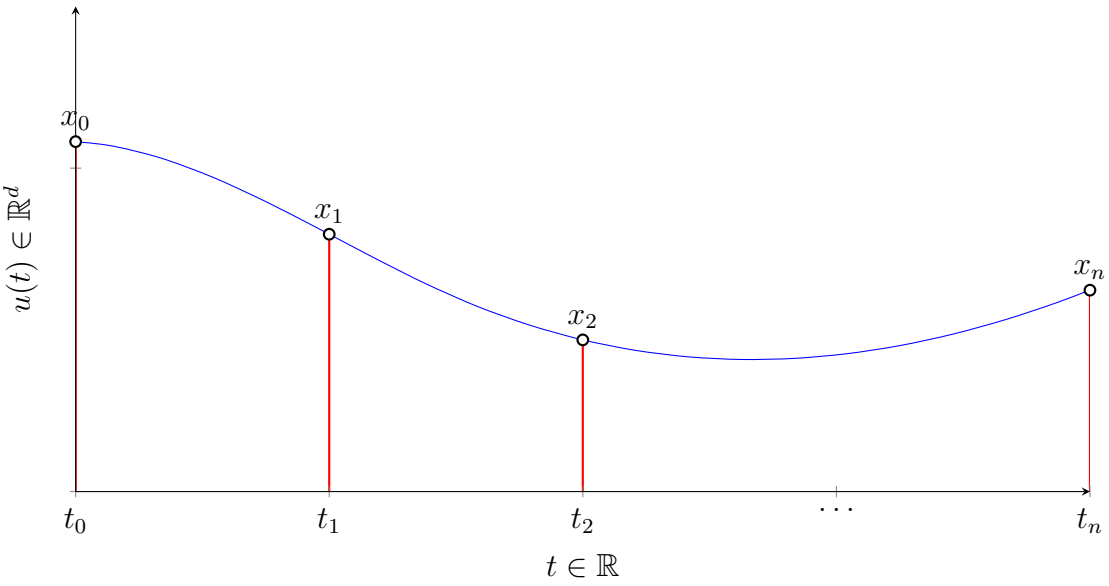


Figure 2.1: Graphically illustrate relationship between the global solution u and the skeletal solution (2.4) as well as the relationship between global solution and local solution.

The construction introduces a mesh to partition the interval $[t_0, t_{\text{end}}]$

$$t_0 < t_1 \cdots < t_N = t_{\text{end}}. \tag{2.3}$$

The sequence of *continuation points* (t_n, x_n) on the mesh (2.3) generates a *skeletal solution*,

$$\{(t_n, x_n)\}_{n=0}^{N}, \qquad t_0 < t_1 < \ldots < t_N = t_{\text{end}}. \tag{2.4}$$

We illustrate the continuation process of constructing a global solution from local solution as well as the relationship between the local solution and the skeletal solution in Figure 2.1.

The continuation process outlined in this section translates into a strategy for the

numerical solution of ODE-IVP. This process is followed by most classical numerical solvers. Classic texts [2, 22, 45, 46] discuss the construction and justification of the discrete algorithms sometimes used to design continuous output methods. We will not review them here.

Modern numerical solvers construct a convenient, computable approximate solution which may not be the true solution [34, 40, 85]. Examples of modern solvers are the continuous output methods such as the continuous Runge–Kutta method, multistep method, and Taylor series method. A continuous output method is a modern solver that is designed to output a global, continuously differentiable approximate solution with the same order as the underlying numerical method [34, 85].

The next section considers a general presentation of the residual and discusses the modern solver as an engineered problem. The defect is the residual for an ODE.

2.2 Residual-based backward error analysis

We consider the residual in this section, a remarkably straightforward procedure and a powerful point of view. The mapping φ specifies a reference problem

$$d \in \mathcal{D} \mapsto \varphi\big(d\big) \in \mathcal{S} \tag{2.5}$$

whose inputs d in data space $\mathcal{D}$ are mapped into outputs s in solution space $\mathcal{S}$, its codomain [18]. Points in output space can be sets, enabling us to consider solutions that are not unique. Whenever the reference problem (2.5) can be expressed with an

implicit function of data and output, we may write

$$d \in \mathcal{D} \mapsto \varphi\big(d\big) = \big\{ s \mid \rho\big(d, s\big) = 0 \big\} \in \mathcal{S}. \tag{2.6}$$

The function ρ is a *defining function* and $\rho\big(d, s\big) = 0$ is a *defining equation.*

Suppose a given reference problem has true solution $s = \varphi(d)$ for given input d which satisfies the defining equation $\rho\big(d, s\big) = 0$, but it is difficult to determine the true solution. The reference problem can be deliberately modified to construct a convenient, computable approximate solution, $\widehat{s}$. The computed solution $\widehat{s}$ is the exact solution of a *modified problem* $\widehat{\varphi}(d)$; that is, $\widehat{s} = \widehat{\varphi}(d)$. In the parlance of Wilkinson's backward error analysis, we call the conveniently computable modified problem $\widehat{\varphi}(d)$ an *engineered problem.* The *residual* r at $\widehat{s}$, the computed solution of a modified problem, is defined by

$$r = r(\widehat{s}) = \rho\big(d, \widehat{s}\big). \tag{2.7}$$

We procedurally obtain the residual by substituting the exact solution of a modified problem $\widehat{\varphi}(d)$ into the defining equation. We can rearrange Equation (2.7) to realize a new defining equation $\widetilde{\rho}\big(d, \widehat{s}\big) \stackrel{\text{def}}{=} \rho\big(d, \widehat{s}\big) - r(\widehat{s}) = 0$. When such a construction is possible, its associated problem

$$\widetilde{\varphi}\big(d\big) = \big\{ s \mid \widetilde{\rho}\big(d, s\big) = 0 \big\} \tag{2.8}$$

is the *reverse-engineered problem.* The easily computed solution generated by the engineered problem is an exact solution to the reverse-engineered problem. See Corless

and Fillion [18, p. 29] for an example when such a construction is not possible.

Given specific data d, a specific modified problem is $\widehat{\varphi}(d)$. We write a generic modified problem as $\widehat{\varphi}$. We follow this convention for reference problem, modified problem, and engineered problem throughout this thesis.

Example 3. *The logistic model*

$$x'(t) = x - x^2, \quad x(0) = 1/2$$

considered in Corless and Corliss [17] is an ODE-IVP *reference problem. Suppose an engineered problem returns the approximate solution*

$$u(t) = \frac{1}{2} + \frac{1}{4}t - \frac{1}{96}t^3, \quad t \in [0, h_0].$$

Compute the defect. The defect in initial condition is zero and

$$\Delta u(t) = u'(t) - u(t) + u(t)^2 = \frac{1}{32}t^2\left(1 - \frac{1}{6}t^2 + \frac{1}{288}t^4\right).$$

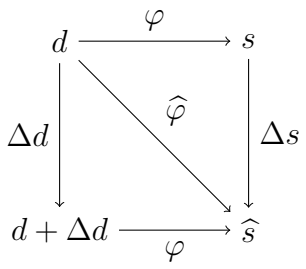
The defect is the residual (2.7), and it is the defining function. The perturbed reference problem is

$$x'(t) = x - x^2 + \frac{1}{32}t^2\left(1 - \frac{1}{6}t^2 + \frac{1}{288}t^4\right), \quad x(0) = 1/2.$$

and u exactly solves this perturbed reference problem.

The reference problem (2.6) and the reverse-engineered problem (2.8) are different in the input data, and the difference is the residual. This observation, illustrated in

the following diagram, is the thrust of Wilkinson's backward error analysis; that is, we depict the backward error Δd in the diagram (2.9). [1]



(2.9)

Reflecting back refers to reflecting the forward error Δs back into the backward error Δd; that is, we find a Δd such that $\varphi\big(d+\Delta d\big)= \widehat{\varphi}\big(d\big)$. The smallest reflection is called the *backward error*. In Section 2.4, we precisely define the backward error, and we prove that the backward error in residual-based backward error analysis is the residual.

We apply the theory developed in this section to control the backward error during the numerical solution of ODE-IVP, a process called defect control. Defect control is the topic of the next section.

2.3 Defect control

Given the mathematical ODE-IVP reference problem (2.1) and a tolerance parameter TOL, suppose Assumption 2 holds. Construct the global numerical solution in a continuation procedure analogous to the construction done in Section 2.1 for the exact solution; that is, suppose approximate solutions $\widehat{u}^j$ and steps T_j, for $j = 1, \ldots, n$, are

[1] Inspired by Higham [57, p. 7], but labeled to reflect our notation.

computed to the local ODE-IVP reference problem,

$$x'(t) = f\big(t, x(t)\big), \quad x(t_{j-1}) = x_{j-1}, \quad t \in [t_{j-1}, t_{j-1} + T_j]$$

with an engineered problem $\widehat{\varphi}$. Set the final time $t_n \stackrel{\text{def}}{=} t_{n-1} + T_n$, and evaluate the next continuation point $\big(t_n, x_n\big) \stackrel{\text{def}}{=} \big(t_n, \widehat{u}^n(t_n)\big)$. Evaluating the local reference problem at v denoting $\widehat{u}^n$ defines the defect Δv at v which is computed as

$$\Delta v(t) \stackrel{\text{def}}{=} v'(t) - f\big(t, v(t)\big), \quad t \in [t_{n-1}, t_n] \quad \text{and} \quad \Delta v(t_{n-1}) \stackrel{\text{def}}{=} v(t_{n-1}) - x_{n-1}. \tag{2.10}$$

The approximate solution must be differentiable to compute the defect. The defect describes the extent to which the first derivative of the approximate solution v' fails to satisfy tangency with f, and it describes the extent to which v fails to satisfy the initial condition. Certainly, $\Delta v(t_j) = 0$ for all j with a possible exception at $j = 0$. The data space for the local reference problem is

$$\mathcal{D} \stackrel{\text{def}}{=} \big\{ d \mid d = \big(f, t_{n-1}, t_n, x_{n-1}\big) \big\}, \quad t_n \stackrel{\text{def}}{=} t_{n-1} + T_n,$$

and the local reference problem in terms of the defect as the defining equation is

$$d \in \mathcal{D} \mapsto \varphi(d) \stackrel{\text{def}}{=} \big\{ v(d) \mid \Delta v(t) = 0 \ \text{ for } \ t \in [t_{n-1}, t_n], \ \ \Delta v(t_{n-1}) = 0 \big\}. \tag{2.11}$$

We write the approximate solution v as $v\big(d\big)$ to emphasize the dependence of it on the reference problem data. From (2.8), the reversed-engineered problem is

$$\widehat{\varphi}\big(d\big)= \left\{ v \mid \begin{array}{l} v'\big(t\big)-f\big(t,v\big(t\big)\big)-\Delta v\big(t\big)= 0 \ \text{ for } \ t\in[t_{n-1},t_n], \\ v\big(t_{n-1}\big)-x_{n-1}-\Delta v\big(t_{n-1}\big)= 0 \end{array} \right\}. \tag{2.12}$$

The approximate solution from the engineered problem v is the exact solution of the reverse-engineered problem (2.12).

We adopt Enright's defect control idea: Control the magnitude of the true defect $\|\Delta v\|$ by controlling the stepsize T_n. Defect control is stepsize control [17]. We solve exactly the local numerical ODE-IVP

$$v'\big(t\big)= f\big(t,v\big(t\big)\big)+\Delta v\big(t\big), \quad v\big(t_{n-1}\big)= x_{n-1}+\Delta v\big(t_n\big), \quad \|\Delta v\|_{[t_{n-1},t_n],\infty}\leq \text{TOL}. \tag{2.13}$$

We want to take a step as large as possible that rigorously satisfies $\|\Delta v\|\leq \text{TOL}$. This is theoretically possible. Indeed, we have $\|v(t_{n-1})-x_{n-1}\|=\|\Delta v(t_{n-1})\|<\text{TOL}$. The defect is continuous – it is a polynomial. By continuity, there is some maximal stepsize $0<T\leq t_{\text{end}}$ such that $\|\Delta v(t)\|\leq \text{TOL}$, for $t\in[t_{n-1},t_{n-1}+T]$.

A local approximate solution might not induce a global continuous defect. A *deviation* is a defect that is only locally continuous, and a deviation is distinguished in notation from a defect with a lower case delta, δ. We chose the absolute operator Δu for our discussion as it is somewhat easier to evaluate. The relative operator δu can be used, and it is preferable over the absolute error when the modelling context demands the proper choice of scale [18, p. 21]. The relative operator evaluates the

relative defect, and it is defined componentwise

$$\delta_i v(t) \stackrel{\text{def}}{=} \frac{\frac{d}{dt}v_i(t) - f_i(t, v(t))}{f_i(t, v(t))} = \frac{\frac{d}{dt}v_i(t)}{f_i(t, v(t))} - 1 \tag{2.14}$$

whenever each component of the vector field f is not zero.

The next section pursues a deeper understanding of backward error analysis.

2.4 Backward error analysis

In this section, we consider a powerful framework for error analysis which is the theoretical foundation for defect control. We highlight relevant results from graduate textbooks by Corless and Fillion [18, Chapter 1] and Higham [57, Chapter 1].

An *algorithm* is a finite sequence of basic operations leading to an approximate solution $\widehat{s}$ of the mathematical problem $s = \varphi(d)$. An algorithm may deliberately alter the reference problem to make computation easier or even possible, and the process defines a *modified problem* or *engineered problem* $\widehat{\varphi}$. The approximate solution $\widehat{s}$ is the exact solution of an engineered problem $\widehat{\varphi}$; that is, $\widehat{s} = \widehat{\varphi}(d)$. Whenever solutions of a modified problem capture the expected characteristics of the reference problem, we say the engineered problem is a *nearby problem*.

Example 4. *Apply the quadratic formula to find all roots of a quadratic polynomial*

$$\rho\big(x, \eta\big) = \eta^2 - 2x\eta + x^2$$

for an arbitrary, but fixed real x. *The reference problem is, given an arbitrary but*

fixed real value for x,

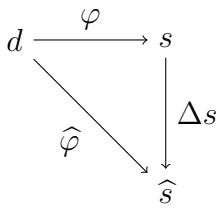
$$d = \big(1, -2x, x^2\big) \mapsto s = \big\{\eta \mid \rho\big(x, \eta\big) = 0\big\}.$$

The solution algorithm applies the quadratic formula to find the roots

$$s = \big\{\eta \in \mathbb{R} \mid \eta^2 - 2x\eta + x^2 = 0,\ x \in \mathbb{R}\big\} = \big\{x\big\} \stackrel{\text{def}}{=} x.$$

By convention, we flatten the set for singleton as in the last equality. Check that x satisfies the defining equation. The modified problem, here the quadratic formula, in exact arithmetic exhibits the exact solution.

We depict the relationship between the reference problem φ and its true solution s along with the modified problem $\widehat{\varphi}$ and its solution $\widehat{s}$ in diagram (2.15):



(2.15)

We need norms to compare solutions and measure error. Error can be measured in absolute error or relative error, and the solutions being compared are expected to be members of a normed linear space.

An example of a normed linear space is the space of continuously differentiable 1-paths under the infinity-norm. If $\mathcal{P} \stackrel{\text{def}}{=} \big\{x \mid x \in C^1\big([0,1], \mathbb{R}^d\big)\big\}$ and $\|\cdot\|$ is the infinity-norm in Equation (1.4), then $\big(\mathcal{P}, \|\cdot\|\big)$ is a normed linear space. The space

of polynomials of degree at most k is a $\big(k+1\big)$-dimensional normed linear space. In fact, it is a finite-dimensional subspace of $\mathcal{P}$.

Consider members $\widehat{v}$ and v from the space of continuously differentiable 1-paths under the infinity-norm, $(\mathcal{P}, \|\cdot\|)$. The *absolute error* Δv is

$$\Delta v = \widehat{v} - v.$$

If $v_i \neq 0$, then the *relative error* δv is defined componentwise as

$$\big(\delta v\big)_i = \frac{\widehat{v_i} - v_i}{v_i}.$$

We have the equality

$$\widehat{v} = v + \Delta v = v\big(1 + \delta v\big) \quad \text{where} \quad (v\ \delta v)_i \stackrel{\text{def}}{=} v_i\ \delta v_i$$

from the absolute and relative error statements whenever $v_i \neq 0$.

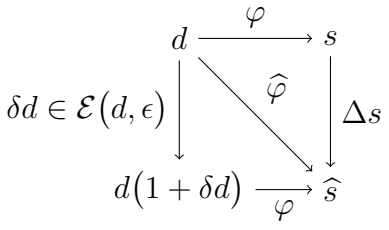
Given any real number ϵ positive, an *absolute ϵ-neighborhood* near v is defined by

$$\big\{\widehat{v} = v + \Delta v \quad \text{such that} \quad \|\Delta v\| < \epsilon\big\}$$

whereas a *relative ϵ-neighborhood* near v is defined by

$$\big\{\widehat{v} = v\big(1 + \delta v\big) \quad \text{such that} \quad \|\delta v\| < \epsilon\big\}.$$

We depict the (relative) backward error δd in the diagram (2.16). [2]



(2.16)

The absolute forward error Δs compares solutions s and $\widehat{s}$

$$\Delta s = s - \widehat{s} = \varphi(d) - \widehat{\varphi}(d).$$

It measures the distance between solutions. The backward error measures distance between problems. Given data d and a tolerance ϵ, the backward error is the smallest (relative) perturbation δd in the equality set $\mathcal{E}\big(d, \epsilon\big)$,

$$\mathcal{E}\big(d, \epsilon\big) \stackrel{\text{def}}{=} \big\{\delta d \mid \widehat{d} = d\big(1 + \delta d\big), \quad \|\delta d\| < \epsilon, \quad \varphi\big(\ \widehat{d}\ \big) = \widehat{\varphi}(d)\big\}.$$

In general, the equality set $\mathcal{E}\big(d, \epsilon\big)$ depends on d and ϵ. The *backward error* is

$$\mathrm{be}\big(d, \epsilon\big) \stackrel{\text{def}}{=} \inf_{\delta d \in \mathcal{E}(d,\epsilon)} \|\delta d\| \tag{2.17}$$

in the extended sense; that is, the backward error is ∞ whenever $\mathcal{E}\big(d, \epsilon\big)$ is empty. The process of bounding the backward error is called *backward error analysis* and inclusively considers all data errors and computation errors simultaneously.

Let the problem data d and tolerance ϵ be given, and choose an engineered problem

[2] Inspired by Higham [57, p. 7], but labeled to reflect our notation.

$\widehat{\varphi}$ to construct numerical solution $\widehat{s} \stackrel{\text{def}}{=} \widehat{\varphi}(d)$. The backward error in residual-based backward error analysis is the smallest residual. Indeed, choose a fixed but arbitrary perturbation $\delta\widehat{s}$ from the equality set $\mathcal{E}\big(d, \epsilon\big)$. By definition, $\widehat{d} \stackrel{\text{def}}{=} d\big(1 + \delta\widehat{s}\big)$ is a member of the relative ϵ-neighborhood near d such that $\varphi\big(\widehat{d}\big) = \widehat{\varphi}(d)$. By construction, a solution of the engineered problem $\widehat{\varphi}(d)$ is a solution of the reverse-engineered problem whose residual $\delta\widehat{s}$ is defined in (2.7). The definition of backward error (2.17) is the smallest residual from the solutions of engineered problem $\widehat{\varphi}(d)$ satisfying the tolerance condition.

In defect control, the reverse-engineered problem has a unique solution according to Theorem 1. It follows that the backward error in defect control is the defect. The following two sections consider two approaches to defect control.

2.5 Asymptotic defect control

In this section, we explain Enright's asymptotic defect control for a computable, continuous output Runge–Kutta (CRK) method, where the approximate solution and the defect are computable. The design of the method enabled Enright to identify an asymptotically correct maximizer of the defect ahead of time so that an inexpensive, asymptotic estimate of the maximum defect can be computed for the purpose of defect control on each integration step. The estimate is controlled with an adaptive time-stepping algorithm to remain less than a user specified tolerance. Results show that the estimate is near optimal on a wide range of non-stiff problems, but it is not rigorous in our sense.

2.5.1 The asymptotic defect control problem

Enright requires the fixed, but arbitrary, tangent vector field f to be Lipschitz to justify his asymptotic approach. Theorem 1 holds under the Lipschitz assumption, but we must assume Assumption 2 holds. Given a user specified tolerance TOL, the engineered problem in this section is the continuous output Runge–Kutta (CRK) method, and we will define these methods in this section. The numerical solution (1.1) to the mathematical reference problem (1.2) is a piecewise polynomial, continuously differentiable, approximate solution u constructed from local polynomial, approximate solutions u^n to the local reference problem (2.2). Denote the exact solution to the n^{th} local reference problem by x^n. In our discussion, we freely refer to the right endpoint of the interval $[t_{n-1}, t_{n-1} + h_n]$ as $t_n \stackrel{\text{def}}{=} t_{n-1} + h_n$. The local approximate solution is always continuously differentiable.

Enright's approximate solution u^n for his CRK method is a fixed *optimal order interpolant* which means that u^n agrees with the local solution x^n to $O\big(h_n^{p+1}\big)$ and its derivative agrees with the derivative of the local solution to $O\big(h_n^p\big)$ on $[t_{n-1}, t_n]$.

Enright implemented CRK methods. He applied a stepsize control strategy to control his estimate est_n by adjusting stepsize h_n to assure that $\text{est}_n \leq \text{TOL}$ on each step. An adaptive time-stepping algorithm selects the stepsize sequence h_n to improve regularity and computational stability [90, 91]. Stepsize control is the topic of Section 2.7. The distribution of points in the mesh (1.5) where $x_n \stackrel{\text{def}}{=} u^n(t_n)$ defines a skeletal solution $\big\{\big(t_n, x_n\big)\big\}_{n=0}^{N}$ as defined in (1.6). Starting at the continuation point $\big(t_{n-1}, x_{n-1}\big)$ with stepsize h_n, let's follow Enright to determine the next continuation point and the next local approximate solution u^n. In a bootstrapping technique, he extends discrete Runge–Kutta formula to a suitable continuous extension [29].

2.5.2 Discrete Runge–Kutta methods

A classical p^{th}-order, s-stage, discrete Runge–Kutta formula determines intermediate approximations to the solution at times $t_{n-1} + c_r h_n$

$$X_r = x_{n-1} + h_n \sum_{j=1}^{s} a_{r,j} k_j, \quad k_r = f\big(t_{n-1} + c_r h_n, X_r\big), \quad r = 1, \ldots, s. \tag{2.18}$$

The discrete approximation to the solution at t_n is

$$x_n = x_{n-1} + h_n \sum_{j=1}^{s} \omega_j k_j. \tag{2.19}$$

This discrete approximate solution is order $p+1$ which means that (2.19) satisfies

$$x_n = x_{n-1} + h_n \sum_{j=1}^{s} \omega_j k_j = x^n(t_n) + O\big(h_n^{p+1}\big). \tag{2.20}$$

Some discrete Runge-Kutta methods have a $\big(p-1, p\big)$ formula pair, the order $p+1$ approximation (2.20) and a second, inexpensive, order p approximation

$$\widetilde{x}_n = x_{n-1} + h_n \sum_{j=1}^{s} \widetilde{\omega}_j k_j = x^n(t_n) + O\big(h_n^{p}\big).$$

The step is taken with the higher order discrete approximate solution $\big(t_n, x_n\big)$ in a practice called *local extrapolation*, and this practice is known to improve the accuracy of the solution.

The "first same as last" design practice saves one function evaluation. The evaluations of $f\big(t_{n-1}, x_{n-1}\big)$ and $f\big(t_n, x_n\big)$ in the stages will save one function evaluation on the next step. Sometimes the first same as last design practice can be used in

the second order p approximation and, as a consequence, will make the second order p approximation less computationally expensive. This is the case for the embedded Runge–Kutta methods RKF 2(3) and DOPRI5 [45, pp. 170-171].

2.5.3 Continuous Runge–Kutta methods

Enright introduces $\widetilde{s}$ additional stages k_{s+1}, k_{s+2}, ..., $k_{s+\widetilde{s}}$ to construct the local approximate solution u^n on $[t_{n-1}, t_n]$. The intermediate points

$$X_{s+r} = x_{n-1} + h_n \sum_{j=1}^{s+\widetilde{s}} a_{s+r,j} k_j, \quad k_{s+r} = f\big(t_{n-1} + c_{s+r} h_n, X_{s+r}\big), \quad r = 1, \ldots, \widetilde{s} \tag{2.21}$$

enable him to construct polynomials of degree at most $p+1$,

$$\tau \in [0,1] \mapsto b_j\big(\tau\big) \in \mathbb{R}, \qquad b_j\big(\tau\big) = \sum_{r=0}^{p+1} \beta_{j,r} \tau^r, \tag{2.22}$$

that define this CRK method

$$u^n\big(t\big) = x_{n-1} + h_n \sum_{j=1}^{s+\widetilde{s}} b_j\big(\tau\big(t\big)\big) k_j, \quad t \in [t_{n-1}, t_n] \mapsto \tau\big(t\big) \stackrel{\text{def}}{=} \big(t - t_{n-1}\big)/h_n. \tag{2.23}$$

Observe $\tau\big(t\big) \in [0,1]$. The additional $\widetilde{s}$ stages and the polynomial coefficients $\beta_{j,r}$ are not uniquely determined by the underlying discrete Runge–Kutta method.

Enright chose coefficients so that the global numerical solution u is continuously differentiable on $[t_0, t_{\text{end}}]$. Enright's coefficients satisfy constraints

$$b_j\big(1\big) = \omega_j, \quad b_{s+r}\big(1\big) = 0, \quad k_1 = f\big(t_{n-1}, x_{n-1}\big), \quad k_{s+1} = f\big(t_n, x_n\big).$$

for $j = 1, \ldots, s$ and $r = 1, \ldots, \widetilde{s}$. The next continuation point $\big(t_n, x_n\big)$ is now defined.

2.5.4 Constructing continuous Runge–Kutta methods I

We follow Enright's earliest construction in this section [24, 26].

Enright computes a polynomial $\tau \in [0,1] \mapsto G(\tau)$ of degree at most $p+1$ satisfying

$$u\big(t\big) - x\big(t\big) = u^n\big(t\big) - x^n\big(t\big) = G(\tau) h_n^{p+1} + O\big(h_n^{p+2}\big), \qquad t \in [t_{n-1}, t_n], \tag{2.24}$$

as a linear combination

$$G(\tau) = q_1(\tau) F_1 + q_2(\tau) F_2 + \cdots + q_m(\tau) F_m \tag{2.25}$$

of m fixed, computable polynomials q_r of degree at most $p+1$ that depend only on the CRK formula and F_r constants that depend only on the problem [24, p. 294].

Evaluate the true (absolute) defect at u. Find

$$\Delta u\big(t\big) \stackrel{\text{def}}{=} u'\big(t\big) - f\big(t, u\big(t\big)\big) \stackrel{\text{def}}{=} u^{n\prime}\big(t\big) - f\big(t, u^n\big(t\big)\big), \quad t \in [t_{n-1}, t_n], \tag{2.26}$$

and

$$\Delta u^n\big(t_{n-1}\big) \stackrel{\text{def}}{=} \begin{cases} u^1\big(t_0\big) - x_0, & n = 1, \\ 0, & n > 1. \end{cases} \tag{2.27}$$

The second part (2.27) of the defect definition depends upon machine format and floating point truncation errors. We accept this error. Find $u^n\big(t_n\big) - x_n = 0$ for $n > 1$

holds exactly as defined by the continuation processes and the design of CRK method. Enright bounds (2.26) by constructing an asymptotic bound on its infinity norm. The defect at the true solution satisfies

$$\Delta x(t) \stackrel{\text{def}}{=} \big(x^{n\prime}(t) - f\big(t, x^n(t)\big)\big) \equiv 0, \qquad t \in [t_{n-1}, t_n].$$

Thus, the defect at the approximate solution u satisfies

$$\Delta u(t) = \big(u^{n\prime}(t) - f\big(t, u^n(t)\big)\big) - \big(x^{n\prime}(t) - f\big(t, x^n(t)\big)\big), \qquad t \in [t_{n-1}, t_n].$$

Enright uses (2.24) and the Lipschitzian property of the tangent vector field to observe

$$\Delta u(t) = \big(u^{n\prime}(t) - x^{n\prime}(t)\big) + O\big(h_n^{p+1}\big) = \frac{d}{d\tau} G(\tau) h_n^p + O\big(h_n^{p+1}\big)$$

which is the mathematical basis for effective asymptotic defect control [26, p. 161-162]. Of course, the point evaluation of $G(\tau)$ at t requires the mapping

$$t \in [t_{n-1}, t_n] \mapsto \tau(t) \stackrel{\text{def}}{=} \big(t - t_{n-1}\big)/h_n$$

which depends upon h_n. Enright is not appealing to this argument. Instead, the maximizer of $\frac{d}{d\tau}G$ is at a fixed τ^* independent of h_n. This is a main point.

Observe from (2.24) and (2.25) that asymptotically the defect is a linear combination of the same m polynomials q_r over each integration subinterval $[t_{n-1}, t_n]$. The linear combination of problem dependent F_j at coefficients $\frac{d}{d\tau} q_j(\tau^*)$ could result in a zero of $\frac{d}{d\tau}G$ at τ^*. Enright handles this potential source of error as part of the stepsize selection strategy, and he chooses τ^* following the procedure in [24] to insure that τ^*

is near the maximum value of each $\frac{d}{d\tau}q_j$. Enright estimates his (absolute) asymptotic defect control estimates $\|\Delta u^n\|_{[t_{n-1},t_n],\infty}$ by

$$\text{est}_n = \|u^{n\prime}(\tau^*) - f\big(t, u^n(\tau^*)\big)\|_\infty.$$

In the special case $m = 1$, the defect will (almost always) asymptotically be a multiple of q_1. In particular, the maximum value of the defect will occur at the maximizer $\tau^* \in [0, 1]$ of q_1 which is a fixed local extrema independent of the problem. This defect control strategy is *strict defect control* (SDC CRK).

In case $m > 1$, Enright determines an evaluation point τ^* that is not near any zero of $q_1, q_2, \ldots q_m$ following the procedure in [24, p. 296]. This defect control strategy is *relaxed defect control* (RDC CRK). Due to the linear combination (2.25), the evaluation point depends on the problem and stepsize, making it difficult to choose a fixed evaluation point that returns the defect maximizer over a wide class of applications.

2.5.5 Constructing continuous Runge–Kutta methods II

We follow Enright and Yan [38] in this section. Enright does RDC CRK for comparison purposes, but he found sufficient numerical evidence to conclude that SDC CRK is reliable. Here reliable is not rigorous. We restrict our attention to SDC CRK.

A crucial observation in Enright's construction of SDC CRK methods compares all interpolants in the same generalized Lagrange basis, a recent nomenclature introduced in the mathematical literature to subsume the Lagrange basis concept for situations requiring higher derivatives. Enright constructs an ordered, generalized Lagrange bases $\big\{\widehat{Q}_j\big\}_{j=0}^{p+1}$ for the purpose of interpolating values at t_{n-1}, t_n, and derivatives at t_{n-1}, t_n, $t_{n-1} + \mu_r h_n$, for $r = 1, \ldots, p-2$ where $t \in [t_{n-1}, t_n] \mapsto \widehat{Q}_j(t) \in \mathbb{R}$. $\widehat{Q}_0$ is the

unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$\begin{aligned}&\widehat{Q}_0(t_{n-1}) = 1, \quad \widehat{Q}_0(t_n) = 0, \quad \widehat{Q}'_0(t_{n-1}) = 0, \quad \widehat{Q}'_0(t_n) = 0,\\ &\widehat{Q}'_0(t_{n-1} + \mu_r h_n) = 0.\end{aligned} \tag{2.28}$$

$\widehat{Q}_1$ is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$\begin{aligned}&\widehat{Q}_1(t_{n-1}) = 0, \quad \widehat{Q}_1(t_n) = 1, \quad \widehat{Q}'_1(t_{n-1}) = 0, \quad \widehat{Q}'_1(t_n) = 0,\\ &\widehat{Q}'_1(t_{n-1} + \mu_r h_n) = 0.\end{aligned} \tag{2.29}$$

$\widehat{Q}_2$ is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$\begin{aligned}&\widehat{Q}_2(t_{n-1}) = 0, \quad \widehat{Q}_2(t_n) = 0, \quad \widehat{Q}'_2(t_{n-1}) = 1, \quad \widehat{Q}'_2(t_n) = 0,\\ &\widehat{Q}'_2(t_{n-1} + \mu_r h_n) = 0.\end{aligned} \tag{2.30}$$

$\widehat{Q}_3$ is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$\begin{aligned}&\widehat{Q}_3(t_{n-1}) = 0, \quad \widehat{Q}_3(t_n) = 0, \quad \widehat{Q}'_3(t_{n-1}) = 0, \quad \widehat{Q}'_3(t_n) = 1,\\ &\widehat{Q}'_3(t_{n-1} + \mu_r h_n) = 0.\end{aligned} \tag{2.31}$$

$\widehat{Q}_{3+k}$ is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$\begin{aligned}&\widehat{Q}_{3+k}(t_{n-1}) = 0, \quad \widehat{Q}_{3+k}(t_n) = 0, \quad \widehat{Q}'_{3+k}(t_{n-1}) = 0, \quad \widehat{Q}'_{3+k}(t_n) = 0,\\ &\widehat{Q}'_{3+k}(t_{n-1} + \mu_k h_n) = 1, \quad \widehat{Q}'_{3+k}(t_{n-1} + \mu_r h_n) = 0, \quad r \neq k.\end{aligned} \tag{2.32}$$

The derivative in equations (2.28)-(2.32) is $\frac{d}{dt}$. Enright found it convenient to scale the basis and change variable to normalize the domain to the unit interval $[0, 1]$. With this change of variable, the $\widehat{Q}_j$ will be polynomials in τ and will be independent of the step

n and stepsize h_n. Let $Q_0 = \widehat{Q}_0$, $Q_1 = \widehat{Q}_1$, and $Q_r = \big(1/h_n\big)\widehat{Q}_r$, for $r = 2, \ldots, \big(p-2\big)$. The $\widehat{Q}_r$ will satisfy equations (2.28)-(2.32), if the Q_r are defined by the following equations where $'$ denotes $\frac{d}{d\tau}$ rather than $\frac{d}{dt}$. From (2.23) and the chain rule, Enright observes $\frac{d}{dt}Q_r\big(\tau\big(t\big)\big) = \big(1/h_n\big)\frac{d}{d\tau}Q_r(\tau)$. Q_0 is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$Q_0(0) = 1, \quad Q_0(1) = 0, \quad Q_0'(0) = 0, \quad Q_0'(1) = 0, \quad Q_0'(\mu_r) = 0. \tag{2.33}$$

Q_1 is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$Q_1(0) = 0, \quad Q_1(1) = 1, \quad Q_1'(0) = 0, \quad Q_1'(1) = 0, \quad Q_1'(\mu_r) = 0. \tag{2.34}$$

Q_2 is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$Q_2(0) = 0, \quad Q_2(1) = 0, \quad Q_2'(0) = 1, \quad Q_2'(1) = 0, \quad Q_2'(\mu_r) = 0. \tag{2.35}$$

Q_3 is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$Q_3(0) = 0, \quad Q_3(1) = 0, \quad Q_3'(0) = 0, \quad Q_3'(1) = 1, \quad Q_3'(\mu_r) = 0. \tag{2.36}$$

Q_{3+k} is the unique polynomial of degree at most $p+1$ satisfying $p+2$ equations,

$$\begin{aligned} &Q_{3+k}(0) = 0, \quad Q_{3+k}(1) = 0, \quad Q_{3+k}'(0) = 0, \quad Q_{3+k}'(1) = 0, \\ &Q_{3+k}'(\mu_k) = 1, \quad Q_{3+k}'(\mu_r) = 0, \quad r \neq k. \end{aligned} \tag{2.37}$$

Enright solved the linear system (2.33)-(2.37) in Maple. Let $q_r \stackrel{\text{def}}{=} \frac{d}{d\tau}Q_r$.

Enright makes several critical design choices. We slightly abuse notation by dropping the n in u^n and x^n to simplify notation to u and x. The intended meaning should be clear from the domain.

The exact solution x to the local reference problem (2.2) was introduced in the problem statement of this section. In a critical design choice, Enright introduces the interpolant $\widetilde{x}$ of degree at most $p+1$ that interpolates the exact local solution x at t_{n-1}, t_n and the derivative of the exact local solution at t_{n-1}, t_n, $t_{n-1}+\mu_r h_n$, for $r = 1, \ldots, p-2$. This polynomial $\widetilde{x}$ has interpolation error

$$x\big(t_{n-1}+\tau h_n\big) - \widetilde{x}\big(t_{n-1}+\tau h_n\big) = \frac{x^{\big(p+2\big)}\big(\eta\big)}{\big(p+2\big)!} h_n^{p+2}\ \tau^2\ \big(\tau-1\big)^2\ \prod_{r=1}^{p-2}\big(\tau-\mu_r\big), \quad (2.38)$$

for some $\eta \in [t_{n-1}, t_n]$ with $\tau \in [0,1]$. The evaluation points $\mu_1, \cdots, \mu_{p-2}$ denote the method parameters $c_{s+1}, \cdots, c_{\bar{s}}$. The generalized Lagrange basis interpolates at these values, and, consequently, Enright compares local interpolants at these values.

The stages (2.18) and (2.21) apply to general implicit Runge–Kutta formula. Enright extensively tested explicit Runge–Kutta methods for non-stiff ODE although he formulated and tested implicit Runge–Kutta formula to handle differential algebraic equations [38]. A Runge–Kutta method is explicit whenever $c_1 = 0$ and $a_{r,j} = 0$ for $j \geq r$. An explicit Runge–Kutta method is computationally less expensive than an implicit Runge–Kutta method, because there are no nonlinear equations to solve.

A critical design choice is to use the "first same as last" design. The conditions $k_1 = f\big(t_{n-1}, x_{n-1}\big)$ and $k_{s+1} = f\big(t_n, x_n\big)$ enable one function evaluation to be reused after the first step.

The Runge–Kutta method admits an interpolant $\widehat{u}$ that agrees with the true local solution x to optimal order $O\big(h_n^{p+1}\big)$ and agrees with the derivative of the true local

approximate solution to optimal order $O\big(h_n^p\big)$. The Runge–Kutta method requires $s+1$ stage evaluations k_1, k_2, ..., k_{s+1} where $k_1 = f\big(t_{n-1}, x_{n-1}\big)$ and $k_{s+1} = f\big(t_n, x_n\big)$. Enright introduces $\bar{s}$ stages k_{s+1}, k_{s+2}, ..., $k_{s+\bar{s}}$ to construct a RDC CRK approximate solution u that agrees with the true local solution x to optimal order $O\big(h_n^{p+2}\big)$. The associated defect of u has order $O\big(h_n^{p+1}\big)$. Enright introduces $p-2$ additional stages $k_{s+\bar{s}+1}$, $k_{s+\bar{s}+2}$, ..., $k_{s+\bar{s}+p-2}$ to construct the local SDC CRK approximate solution $\bar{u}$ such that the contribution to the associated defect of $\bar{u}$ has order $O\big(h_n^{p+2}\big)$. This bootstrapping approach adjusts the interpolation points of the generalized Lagrange basis $\mu_1, \cdots, \mu_{p-2}$ to identify derivatives $\big\{q_r\big\}$ of $\big\{Q_r\big\}$ such that

$$D_r \stackrel{\text{def}}{=} \max_{\tau\in[0,1]} |q_r\big(\tau\big)|, \quad r = 1, 2, \ldots, p-1 \tag{2.39}$$

satisfy $D_1 \leq 2$ and $D_r/D_1 < 1$.

In a generalized Lagrange basis satisfying the constraints (2.39), Enright and Yan [38, Section 2] asymptotically expand the approximate solutions of the previous paragraph to mathematically and computationally justify that the defect satisfies the asymptotic form

$$\Delta\bar{u}\big(t_{n-1}+\tau h_n\big) = q_1\big(\tau\big)F_1h_n^p + \big(\widehat{q}_1\big(\tau\big)\widehat{F}_1 + \ldots + \widehat{q}_L\big(\tau\big)\widehat{F}_L\big)h_n^{p+1} + O\big(h_n^{p+2}\big), \tag{2.40}$$

where $F_1h_n^p = x\big(t_n\big) - x_n$ is the discrete local error associated with the current step, and the linear span of the polynomials $[q_2, q_3, \cdots, q_{p+1}]$ is contained in the linear span of the polynomials $[\widehat{q}_1, \widehat{q}_2, \cdots, \widehat{q}_L]$. Enright applies the analysis in this paragraph to ODE45 from MATLAB, and he derived a new SDC CRK method SDC CRK5.

The maximizer of q_1 is τ^*. Since q_1 is independent of the problem, Enright observed

as $h_n \to 0$ that the profile of $\Delta\bar{u}$, and consequently the norm of the local maximum defect, is proportional to the fixed polynomial q_1, independent of the problem and step. Enright concludes

$$\|\Delta\bar{u}\|_{[t_{n-1},t_n],\infty} = \|\Delta\bar{u}\big(t_{n-1} + \tau^* h_n\big)\|_\infty + O\big(h_n^{p+1}\big)$$

The asymptotic form of the defect (2.40) will demand stepsizes h_n such that

$$\|h_n \widehat{q}_r\|_\infty \ll \|q_1\|_\infty \tag{2.41}$$

The design constraints (2.39) ensure that the maximum value of q_1 is less than 2 so that the stepsize (2.41) can be large enough.

The following situation rarely happens but can occur. The discrete local error $|F_1|$ is very small in magnitude. In this case, the first term in the asymptotic form of the defect (2.40) is nearly zero, and the $O\big(h_n^{p+1}\big)$ terms contribute to the defect. On these isolated steps, Enright reports that the actual maximum defect was smaller than TOL, but the estimated value was not close to the true maximum defect value. Enright introduces a validity check to detect this situation and a modified defect control strategy SDCV CRK.

The validity check involves two additional defect evaluations, and hence two extra function evaluations will be necessary on each step. Observe that $q_1\big(\tau\big)$ is zero at $\tau = 0$, $\tau = 1$, and $|q_1\big(\tau^*\big)|$ is maximum by definition of τ^*. There exists by continuity a first $\tau_1 < \tau^*$ and a first $\tau_2 > \tau^*$ such that

$$q_1\big(\tau_1\big) = q_1\big(\tau_2\big) = q_1\big(\tau^*\big)/2$$

which motivates the definitions

$$R_1 = \frac{\Delta\bar{u}\big(t_{n-1} + \tau_1 h_n\big)}{\Delta\bar{u}\big(t_{n-1} + \tau^* h_n\big)} \quad \text{and} \quad R_2 = \frac{\Delta\bar{u}\big(t_{n-1} + \tau_2 h_n\big)}{\Delta\bar{u}\big(t_{n-1} + \tau^* h_n\big)}$$

Evidently, R_1 and R_2 approach $1/2$ as $h_n \to 0$. This justifies the Enright and Yan validity check that R_1 and R_2 be in the interval $[0.3, 0.7]$. If the validity check is satisfied on all steps, then the user can have increased confidence in the reliability of the integration. Enright modified the SDC CRK strategy so that, when this check is not satisfied, two additional defect evaluations are performed to determine a more appropriate estimate of the maximum defect. These two extra evaluations together with the two validity check evaluations and the first defect evaluation at the maximizer τ^* of q_1 give five evaluations of the defect. Enright in his defect control strategy SDCV CRK chooses the largest absolute value of these values to be the estimate of the maximum defect. The choices of these points to evaluate the defect depends upon the SDC CRK method. The choices made in Enright and Yan [38] can be found in their Figure 3 for SDC CRK5, Figure 4 for SDC CRK6, and Figure 5 for SDC CRK8.

Finally, the estimate of the maximum defect in SDC CRK methods based on the maximizer of q_1 can be unreliable on problems where local errors do not dominate round-off errors in the computations associated with the step. Enright and Yan graphically depict the computational artifact and its resolution in Figure 2.2. Enright determined a credible indication for round-off errors dominating local errors is that the defect value near $\tau = 1$ has magnitude comparable to the estimate $\|\Delta\bar{u}\big(t_{n-1} + \tau^* h_n\big)\|_\infty$. The strategy recommended in Enright and Yan [38, p. 250] to handle this case is to "halt the integration with a warning suggesting a higher precision implementation of the method or a lower order method be used".

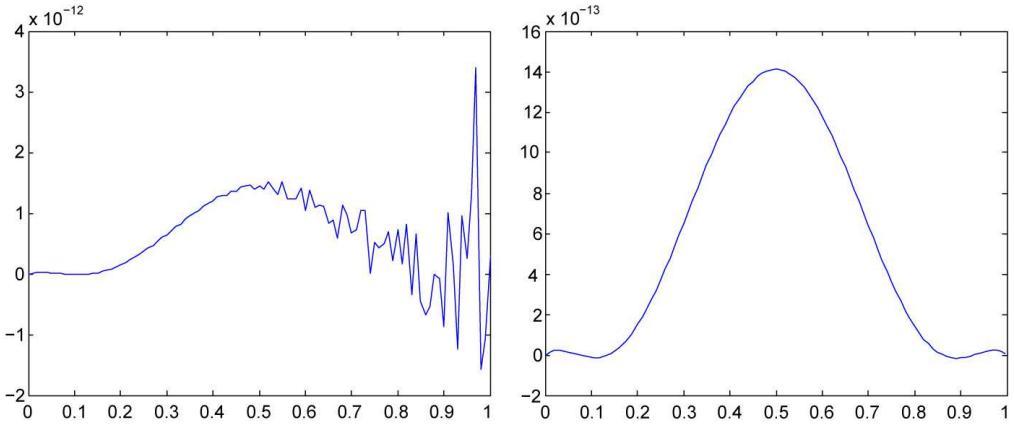


Figure 2.2: Plot of defect vs τ (scaled by its local extremum) for SDC CRK6 on a typical step where roundoff-error is comparable to truncation error as $\tau \to 1.0$. Plot on left is when the defect is evaluated in double precision. Plot on right is when the same defect is evaluated using extended precision. This figure is from [38, p. 250, Fig. 2].

Asymptotic defect control estimates the range of the defect by evaluating the defect at the asymptotic maximizer τ^* and at possibly several more points $t_{n-1} + \tau_j h_n$ with $\tau_j \in [0, 1]$. The next section applies interval analysis to defect control and rigorously bounds the range of the defect on $[t_{n-1}, t_n]$.

2.6 Guaranteed defect control

Guaranteed defect control is not only defect control, it is rigorous defect control. It computes a rigorous enclosure of the defect at the approximate solution, but, unlike its asymptotic defect control counterpart, it does not consider the engineered problem's details or how the approximate solution was constructed. The approximate solution need not be piecewise polynomial, but it could be.

Guaranteed defect control requires an estimate of some appropriate norm, usually the infinity norm [17]. One way to estimate the infinity norm is to obtain a rigorous

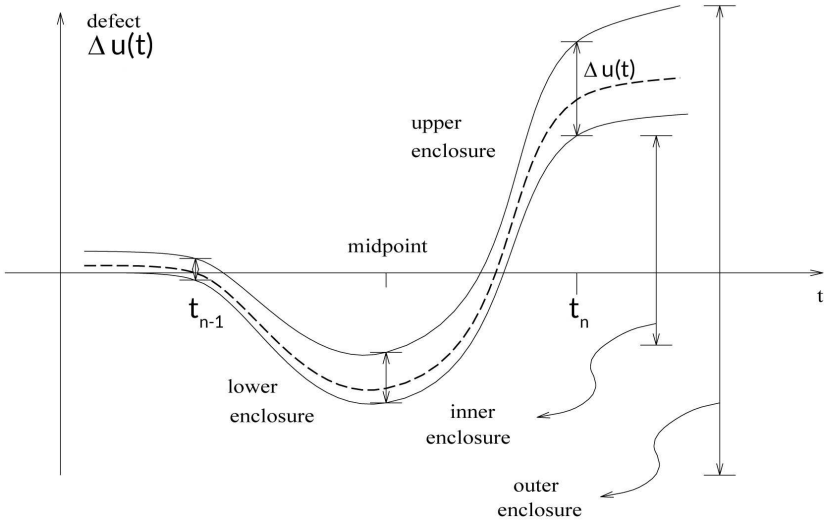


Figure 2.3: Inner and outer enclosures of the defect. This figure is from [17, Figure 4].

bound on the range of the defect. Corless and Corliss [17] recognise that the problem of bounding the range of a function is a well studied problem in interval analysis. The Corliss tutorial [20] and the Corliss outlook article [21] on defect control orient the reader to relevant interval techniques applicable to defect control. Corless and Corliss suggest to use tight bounds on the range of the defect using natural interval extensions, monotonicity, concavity, mean value forms, and Taylor forms [17]. As part of the computation of a tight enclosure for $\|\Delta u\|_{[t_{n-1},t_n],\infty}$, they suggest to evaluate Δu at t_{n-1}, t_n, the midpoint $\big(t_n - t_{n-1}\big)/2$, and on the entire interval of the integration step. This approach enables the computation of both inner and outer enclosures to confirm the tightness of the enclosures as illustrated in Figure 2.3.

The dotted line in Figure 2.3 represents the true defect Δu. The lower and upper enclosures of Δu are represented as curves. The outer enclosure must contain the lower and upper enclosure over the integration interval $[t_{n-1}, t_n]$, but it may include some overestimation. The inner enclosure is an interval that may include some

underestimation. The set difference between the outer and the inner enclosures is an indication of the tightness with which the range of Δu has been enclosed.

The Corless and Corliss [17] defect control algorithm enumerated in Algorithm 1 computes an (outer) enclosure δ of the true defect over the integration interval $[t_{n-1}, t_n]$. The directives "reduce" or "increase" invoke an adaptive time-stepping algorithm, a special piece of software discussed in Section 2.7, but the stepsize control strategy for determining a step h_n for which $\|\Delta u\|_{[t_{n-1},t_n],\infty} \leq \text{TOL}$ is not specified in their algorithm.

Algorithm 1: Defect-controlled algorithm

Input: f, t_0, t_{end}, x_0, TOL
Output: The nodes count N,
Nodes $t_0, t_1, \ldots, t_N = t_{\text{end}}$,
Continuously differentiable u which exactly solves $u'(t) = f(t, u(t)) + \Delta u(t)$,
Guarantee that $\|\Delta u\|_{[t_{n-1},t_n],\infty} \leq \text{TOL}$ for all $t \in [t_0, t_{\text{end}}]$.

```
1  h := Initial trial step
   forall Steps n = 0, ..., N do
2  |  break if t_n > t_end
   |  Compute u^n(t), a continuous approximate solution on t_{n-1} ≤ t ≤ t_{n-1} + h
   |  Define the defect Δu^n(t) := u^n'(t) − f(t, u^n(t))
   |  Evaluate enclosure δ of Δu^n(t)
   |  if |δ| > TOL then
3  |  |  reduce h and repeat
   |  |
4  |  else if |δ| ≪ TOL then
5  |  |  increase h and repeat
6  |  else
7  |  |  accept step h_n ← h and t_n ← t_{n-1} + h_n
8  |  end
9  end
```

Corless and Corliss build their algorithm based on the premise that "The modeler can choose a stepsize to control the size of the defect and to guarantee that no error larger than those already made in the modeling process will be introduced by the solution process [17]."

Guaranteed defect control solves the local numerical ODE-IVP (2.13) to mathematical certainty. Consequently, the approximate solution exactly solves the global numerical ODE-IVP (1.9). Rigorously controlling a measure of the defect to be less than a user specified tolerance is rigorous defect control.

The adaptive time-stepping algorithm in a numerical ODE method controls stepsize based on an error estimate, and it indirectly controls the stability of the numerical integration. An adaptive time-stepping algorithm is a "separate piece of software that should be carefully analyzed and implemented [90]", and they are considered in the next section.

2.7 Stepsize control

The "elementary" stepsize controller [89] is a heuristic based on the assumption that error r is proportional to h^k,

$$\|r_n\| = \varphi_{n-1}\ h_{n-1}^k \tag{2.42}$$

for stepsize h, and the assumption φ is slowly varying as the process moves from step $n-1$ to step n; that is, assume

$$\varphi_n \approx \varphi_{n-1}. \tag{2.43}$$

The elementary controller seeks to achieve $\|r_n\| = \text{TOL}$. We want the predicted step to be as large as possible and maintain $\|r_n\| \leq \text{TOL}$ for all steps $h < h_n$ with $\|r_n\| = \text{TOL}$

for step h_n. The elementary controller is designed so the approximation

$$\frac{\text{TOL}}{\|r_{n-1}\|} = \frac{\|r_n\|}{\|r_{n-1}\|} = \frac{\varphi_n \; h_n^k}{\varphi_{n-1} \; h_{n-1}^k} \approx \left(\frac{h_n}{h_{n-1}}\right)^k \tag{2.44}$$

holds. Taking the approximation as exact, rearrange (2.44) to find

$$\frac{h_n}{h_{n-1}} = \left(\frac{\text{TOL}}{\|r_{n-1}\|}\right)^{1/k}.$$

A safety factor $\theta < 1$ is introduced, and we have the stepsize controller

$$h_n = \left(\frac{\theta \; \text{TOL}}{\|r_{n-1}\|}\right)^{1/k} \; h_{n-1}. \tag{2.45}$$

Here k is p or $p+1$ where p is the order of the local error estimator r_n. If the order of convergence for the method is p, then take $k = p+1$ for an error-per-step control. Take $k = p$ for error-per-unit-step control [90, p. 2].

The stepsize selector (2.45) is designed to predict a next stepsize based on the evaluation of the asymptotic estimate r_{n-1} of the local error. It does not matter if the current step was accepted or rejected the stepsize estimator (2.45) is used in both cases. Good codes have the safety factors different in rejected and accepted steps.

The assumptions for the elementary stepsize control fit into Enright's asymptotic defect control model. Enright [24] and Higham [54] use the elementary controller where, after each step, whether successful or not successful, the newly proposed stepsize h_{n+1} is chosen according to (2.45) with $\theta = 0.9$.

Guaranteed defect control constructs an approximate solution for the reference problem. We know the approximate solution exactly solves the perturbed reference

problem. We draw an analogy between the traditional use of (2.45) in forward error control and defect control to infer a stepsize controller. We use Taylor models to represent the defect over each step. We want to control the defect with stepsize. The r_{n-1} in (2.45) is the defect, and the k is the degree of the Taylor model.

Hairer and Wanner [46] presented PID controllers from a continuous controller point of view which aided understanding. Here P is the proportional controller, I is the integral controller, and D is the differential controller. Hairer and Wanner explain the ability to simultaneously invoke these aspects and their influence on the stability of the process. Gustafsson, Lundh, and Söderlind presented the PI controller [44]. In a sequence of papers [89, 90, 91], Söderlind unified the controller theory around the asymptotic assumption and extended the theory to include digital filters.

Authors investigated PI controllers for Runge–Kutta methods obtaining a good deal of numerical experience, see articles by Gustafsson, Lundh, and Söderlind [44], Gustafsson [42] on explicit Runge–Kutta, Gustafsson [43] on implicit Runge–Kutta, as well as articles deSwart and Söderlind [95], Hall [47, 48], Hall and Higham [49], and Higham and Hall [56] on constructing error estimators for implicit Runge–Kutta.

An alternative approach to controllers consider the radius of convergence of the series, see Chang and Corless [10] and Bergsma [7]. Bergsma [7] investigated three special stepsize controllers based on radius of convergence for Taylor series methods for their suitable performance in reentry problems from aerospace engineering.

The next section considers estimating the condition number for an ODE-IVP and introduces a technique that estimates the global error. Enright [29, p. 9] proves that this estimate of the global error can be used to improve the accuracy of the numerical solution.

2.8 Global error and condition

The analysis in this section follows Enright [29, p. 8-9]. Assume s is the true solution to the mathematical ODE-IVP reference problem (1.2), and assume the approximate solution u defined in (1.1) satisfies the perturbed reference problem (1.9). The absolute forward error is $\epsilon(t) = s(t) - u(t)$.

Enright constructs a companion ODE-IVP, which is a reference problem for the absolute error. He proposes to solve the companion ODE-IVP with the same algorithm applied to obtain the approximate solution u of the engineered problem, and calls its approximate solution E. Evaluate ΔE, the defect for the companion ODE-IVP at E. Enright points out three major consequences of this analysis. Let's investigate.

The first consequence: Rearrange the absolute forward error relationship. Find $s(t) = \epsilon(t) + u(t)$ upon solving the absolute forward error identity for s. Differentiate the absolute forward error. The companion ODE-IVP is

$$\epsilon'(t) = c\big(t, \epsilon(t)\big), \quad \epsilon(t_0) = 0, \quad t \in [t_0, t_{\text{end}}] \tag{2.46}$$

where vector field c is defined as

$$c\big(t, \epsilon(t)\big) \stackrel{\text{def}}{=} f\big(t, \epsilon(t) + u(t)\big) - u'(t). \tag{2.47}$$

Because $s(t) = \epsilon(t) + u(t)$ and s is the solution to the mathematical ODE-IVP reference problem, Enright's definition of c follows from the analysis

$$\epsilon'(t) + u'(t) = s'(t) = f\big(t, s(t)\big) = f\big(t, \epsilon(t) + u(t)\big).$$

Enright recommends to solve the companion ODE-IVP with the same algorithm applied to obtain the approximate solution u of the engineered problem, and call its approximate solution E. Evaluate ΔE, the defect for the companion ODE-IVP at E. Find E exactly solves

$$E'\big(t\big)= c\big(t, E\big(t\big)\big)+\Delta E\big(t\big), \quad E\big(t_0\big)= 0, \quad \|\Delta E\|_{[t_0,t_{\rm end}],\infty} \leq \tau, \tag{2.48}$$

the perturbed reference problem for the companion for the ODE-IVP. This determines τ, given the algorithm and step sizes used to compute u.

Enright demonstrates on test problems that $\tau \geq \|\Delta E\|_{[t_0,t_{\rm end}],\infty}$ is much smaller than TOL. Enright determines τ by sampling $\|\Delta E\big(t\big)\|_\infty$ on each step. In Table 2.1, we report U, the maximum global error associated with u in units of TOL. We also report UPE, the maximum global error associated with $u + E$ in units of TOL. Enright determined both global error by computing the true global error at 100 sample points per step, and we are reporting in Table 2.1 his results found in article [29]. Enright's results show a remarkable improvement in the global error between these two approximate solutions in these two case studies. The Lorenz system at TOL $= 10^{-8}$ and orders 6 and 8 is an exception.

The second consequence: The approximate solution $u + E$ satisfies the perturbed reference problem for the companion ODE-IVP with defect ΔE. In symbols, find

$$\begin{aligned} E'\big(t\big)+u'\big(t\big) &= c\big(t, E\big(t\big)\big)+\Delta E\big(t\big)+u'\big(t\big) \\ &= f\big(t, E\big(t\big)+u\big(t\big)\big)-u'\big(t\big)+\Delta E\big(t\big)+u'\big(t\big) \\ &= f\big(t, E\big(t\big)+u\big(t\big)\big)+\Delta E\big(t\big) \end{aligned} \tag{2.49}$$

by the definition of the vector field c. The approximate solution $u + E$ satisfies the

order	tol	Lorenz $t_{\text{end}} = 15$ U/UPE	pred. prey $t_{\text{end}} = 40$ U/UPE
5	10^{-2}	4400 / 470	3.7 / 0.002
	10^{-4}	190000 / 50	7.3 / 0.009
	10^{-6}	190000 / 170	11.4 / 0.004
	10^{-8}	1800000 / 6800	14.4 / 0.041
6	10^{-2}	4200 / 290	2.2 / 0.0006
	10^{-4}	280000 / 310	4.6 / 0.001
	10^{-6}	150000 / 320	2.5 / 0.001
	10^{-8}	150000 /20000	3.5 / 0.008
8	10^{-2}	5500 / 70	9.5 / 0.0009
	10^{-4}	16000 / 8.2	6.1 / 0.002
	10^{-6}	14000 / 420	6.1 / 0.003
	10^{-8}	20000 /48000	14.4 / 2.0

Table 2.1: Compare the true global error sampled at 100 points per step between the approximate solution u and the improved approximate solution $u + E$ measured in units of TOL. Results from [29, Table 3 p. 290 and Table 4 p.291].

perturbed reference problem (1.9) including its initial condition, and it is experimentally shown to be an improvement over the approximate solution u.

The third consequence: Enright uses the approximate solution E which exactly solves (2.48) to estimate the condition of the mathematical ODE-IVP reference problem. The estimate relies on proving the inequality

$$\|\epsilon(t)\| \leq K(t)\text{TOL} \tag{2.50}$$

where K reflects the sensitivity of the exact solution s of the reference problem (1.2) with respect to perturbations. Enright appeals to the variation of constants formula to prove (2.50), and he refers the reader to the excellent discussion on the variation of

constants formula found in [45, Chapter I.14]. Certainly,

$$\kappa \stackrel{\text{def}}{=} \max_{t\in[t_0,t_{\text{end}}]} K\big(t\big) \tag{2.51}$$

can act as a condition number for the ODE-IVP reference problem. Enright computes a lower bound κ_ϵ,

$$\kappa_\epsilon \stackrel{\text{def}}{=} \max_{t\in[t_0,t_{\text{end}}]} \|\epsilon\big(t\big)\|/\text{TOL}. \tag{2.52}$$

He realized that an effective estimate of the conditioning of the ODE-IVP reference problem is

$$\kappa_E \stackrel{\text{def}}{=} \max_{t\in[t_0,t_{\text{end}}]} \|E\big(t\big)\|/\text{TOL}. \tag{2.53}$$

The effectiveness of this estimate depends upon an accurate approximate solution E to the absolute error ϵ and the sharpness of the defect estimate $\|\Delta u\| \leq \text{TOL}$.

Corless and Fillion [18, Section 12.3.2] discuss the condition of an ODE-IVP in the tangent space with the equation of first variation, and the result (2.50) is in the last two display lines on page 530. This approach assumes a convergence result. The Gröbner-Alexeev approach, including the following theorem, is considered by Corless and Fillion [18, Section 12.3.3].

Theorem 5 (Gröbner-Alexeev nonlinear variation-of-constants formula). *Assume s is the solution to the mathematical ODE-IVP reference problem (1.2) and assume z is*

the solution of

$$z'\big(t\big)= f\big(t,u\big(t\big)\big)+\gamma v\big(t,z\big(t\big)\big),\quad z\big(t_0\big)= x_0. \tag{2.54}$$

If f is continuously differentiable, then

$$z\big(t\big)-s\big(t\big)= \gamma\int_{t_0}^{t} G\big(t,\tau,z\big(\tau\big)\big)v\big(\tau,z\big(\tau\big)\big)d\tau \tag{2.55}$$

where the matrix function G is given by

$$G_{ij}\big(t,\tau,z\big(\tau\big)\big)\stackrel{\text{def}}{=}\frac{\partial s_i}{\partial x_{0,j}}\big(t,\tau,z\big(\tau\big)\big)$$

and acts as a condition number, that is, as a quantity dictating how $\gamma v\big(t,z\big(t\big)\big)$ will be magnified over the interval of integration $[t_0,t_{end}]$.

Gröbner-Alexeev applies to the perturbed reference problem (1.9) with $\Delta u\big(t_0\big)= 0$, $\gamma\stackrel{\text{def}}{=} 1/\|\Delta u\|_{[t_0,t_{\text{end}}],\infty}$, and the special defect perturbation $v\big(t,z\big(t\big)\big)\stackrel{\text{def}}{=}\Delta u\big(t\big)$ in (1.9). The mathematical form of global error and its relationship to the defect is given in (2.55). We immediately see that the global error can be very large even though the defect is very small. This phenomenon happens for chaotic problems.

The articles [15, 16, 52] and the references therein offer an introduction to the "shadowing" technique and assign meaning to the computed solution of chaotic problem. The "shadowing" technique is a forward error approach to chaotic problems. Corless and Fillion [18, p. 206] state that the defect "provides a cheap and reliable alternative to the so-called "shadowing" techniques for chaotic problems".

Chapter 3

Taylor models

Taylor models often, but not always, reduce overestimation and excess width as the degree of the Taylor polynomial increases [72, p. 137]. We provide supporting evidence for this heuristic, and we aim to intuitively understand the meaning of the statement.

A *rigorous algorithm* returns a *bound* such that the true mathematical solution is in that bound. For example, a rigorous algorithm for computing $\sqrt{2}$ may return a bound $[1.414, 1.415]$. The result is rigorous because the mathematical statement $\sqrt{2} \in [1.414, 1.415]$ is true. Synonyms for rigorous commonly used in the literature are validated, reliable, and verified.

A *Taylor model of degree k* represents a function f over $\boldsymbol{T}$ as a couple $\big(p, \boldsymbol{r}\big)$

$$\big(p, \boldsymbol{r}\big) \quad \text{means} \quad f(t) - p(t) \in \boldsymbol{r} = [\,\underline{r}, \overline{r}\,] \quad \text{for all} \quad t \in \boldsymbol{T}. \tag{3.1}$$

The Taylor model polynomomial p can be a Chebyshev or a Taylor approximation of degree k to f, and Taylor models carefully compute the bound r for the Taylor form.

Taylor models resolve the increasing degree problem. Consider the approximate

solution in Example 3,

$$u(t) = \frac{1}{2} + \frac{1}{4}t - \frac{1}{96}t^3, \quad t \in [0, h_0].$$

It has defect

$$\Delta u(t) = u'(t) - u(t) + u(t)^2 = \frac{1}{32}t^2 - \frac{1}{192}t^4 + \frac{1}{9216}t^6.$$

Notice the degree of the local, polynomial approximate solution is 3, and the degree of the defect is 6. Exact Taylor arithmetic doubles the degree of the defect because of the squared term in the vector field. The degree of the Taylor polynomial is fixed, and higher order terms arising in Taylor arithmetic are enclosed in the Taylor model bound.

The SOLLYA algorithms [12, 13] compute Taylor models and implement Taylor model arithmetic. More generally, the SOLLYA package computes a rigorous polynomial approximation (RPA) of a given real-valued function in one independent variable. Joldes [61], a SOLLYA coauthor, advanced rigorous algorithm from interval arithmetic to polynomial approximation (Chebyshev and Taylor) with RPA where the approximating polynomial is not restricted to be Taylor. Joldes justifies the SOLLYA algorithms in her thesis [61]. For functions involving more than one independent variable, Berz and Makino [8, 9, 69, 67, 68, 70, 71] simultaneously compute the Taylor model polynomial and the Taylor model bound to obtain a tight rigorous enclosure in space and time.

SOLLYA can evaluate real-valued functions at intervals and compute rigorous supremum norm. The function must involve only basic functions, and it must be sufficiently smooth. All the basic functions SOLLYA resolves are listed in Table 3.1.

+	−	·	/	∘	
sin	asin	cos	acos	tan	atan
exp	exp_m1	log	log_2	log_10	log_1p
sinh	asinh	cosh	acosh	tanh	atanh
abs	erf	erfc	sqrt	pow	

Table 3.1: Basic functions implemented in SOLLYA

The recent release of this freely available software package is timely for our project. The SOLLYA package implements RPA through a command line interface. We extend the freely available SOLLYA codes in this thesis to compute Taylor models using operator overloading in the C++ programming language without going through the command line interface.

To rigorously bound the defect of ODE initial-value problems, write a generic template for the tangent vector field. A generic template implementing the vector field function is used for automatic differentiation, Taylor model arithmetic, and floating point arithmetic. Of course, each component of the vector field must be an elementary function.

This chapter introduces interval arithmetic, Taylor forms and Taylor models, and our SOLLYA interface C++ class, class `Tmodel`. We address interval arithmetic, overestimation, excess width, and Taylor models without pursuing an axiomatic approach to the topic of interval analysis. The interested reader can find a thorough treatment of this material in the Moore, Kearfott, and Cloud introductory book [72], the foundations article by Rall [82], the interval analysis notation standard by Kearfott, Nakao, Neumaier, Rump, Shary, and Van Hentenryck [62], and the review article [80].

3.1 Interval arithmetic

An *interval* $\boldsymbol{x}$ is identified with the (nonempty) set of points between its lower bound $\underline{x} \in \mathbb{R}$ and its upper bound $\overline{x} \in \mathbb{R}$,

$$\boldsymbol{x} = \big\{ x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x} \big\}.$$

An interval is closed and nonempty. A degenerate interval $\boldsymbol{x}$ contains a single element x, and we agree, by convention, to *identify* $[x, x]$ with x.

A *box* of dimension d denoted $\boldsymbol{x} \stackrel{\text{def}}{=} [\,\underline{x}, \overline{x}\,]$ generalizes intervals to finite dimensional space, and it is identified with the (nonempty) set of points between its lower bound $\underline{x} \in \mathbb{R}^d$ and its upper bound $\overline{x} \in \mathbb{R}^d$,

$$\boldsymbol{x} = \big\{ x \in \mathbb{R}^d \mid \underline{x} \leq x \leq \overline{x} \big\}.$$

A vector $x \in \mathbb{R}^d$ is *contained* in a box $\boldsymbol{x}$ if and only if $\underline{x} \leq x \leq \overline{x}$; that is, $x \in [\,\underline{x}, \overline{x}\,]$ if and only if the statement $\underline{x} \leq x \leq \overline{x}$ holds componentwise, $x_i \in [\,\underline{x_i}, \overline{x_i}\,]$. We say that $\boldsymbol{x}$ is *degenerate* if $\underline{x} = \overline{x}$. By convention, a degenerate box $[x, x]$ is identified with the vector x. The set of all boxes of dimension d is denoted by $\mathbb{IR}^d$, and we write $\dim \boldsymbol{x} = d$ for the dimension of $\mathbb{R}^d$ where $\boldsymbol{x} \subset \mathbb{R}^d$. The *width* of a box $\boldsymbol{x}$ is $\operatorname{wid} \boldsymbol{x} = \overline{x} - \underline{x} \geq 0$, its *radius* is $\operatorname{rad} \boldsymbol{x} = \frac{1}{2} \operatorname{wid} \boldsymbol{x} = \frac{1}{2}\big(\overline{x} - \underline{x}\big)$, and its *midpoint* is $\operatorname{mid} \boldsymbol{x} = \frac{1}{2}\big(\overline{x} + \underline{x}\big)$.

Denote the set of *basic functions* in Table 3.1 by $\mathcal{B}$. Interval libraries compute tight bounds for the range of a basic function over a given interval [61]. The justification is straightforward for *monotonic* functions; that is, functions which are either increasing or decreasing as the independent variable increases. Given any monotonic increasing

function f over an interval $\boldsymbol{x}$, $f\big(\boldsymbol{x}\big)= [f\big(\underline{x}\big), f\big(\overline{x}\big)]$. For example, take the exponential function $f\big(x\big)= \exp\big(x\big)$ on $\boldsymbol{x} = [\underline{x}, \overline{x}]$. Then $f\big(\boldsymbol{x}\big)= [\exp\big(\underline{x}\big), \exp\big(\overline{x}\big)]$. A piecewise monotonic example is only slightly more complicated to justify. For example, consider the interval-valued *absolute value function* defined on intervals by

$$\mathrm{abs}\big(\boldsymbol{x}\big)= \big\{|x| \mid x \in \boldsymbol{x}\big\}.$$

We define the mignitude and absolute value of an interval to express abs as an interval. The *mignitude* of an interval $\boldsymbol{x}$ is the number

$$\langle \boldsymbol{x} \rangle = \min\big\{|x| \mid x \in \boldsymbol{x}\big\},$$

and the real-valued *absolute value* of an interval $\boldsymbol{x}$ is the number

$$|\boldsymbol{x}| = \max\big\{|x| \mid x \in \boldsymbol{x}\big\}.$$

Then the interval-valued absolute value mapping of an interval $\boldsymbol{x}$ is the interval

$$\mathrm{abs}\big(\boldsymbol{x}\big)= [\langle \boldsymbol{x} \rangle, |\boldsymbol{x}|].$$

The class of basic functions $\mathcal{B}$ is used to build up more complicated functions. Any real–valued function expressed by a finite number of arithmetic operations and compositions with basic functions and constants is called an *elementary function*.

Authors traditionally list intervals based on expressions for their endpoints for the addition, subtraction, product, and quotient interval operations, but these formulae are consistent with the treatment of any other basic function. The proof requires a

characterization: A point x is *contained* in interval $\boldsymbol{x}$ if and only if $\underline{x} \leq x \leq \overline{x}$. The *sum* of two intervals $\boldsymbol{x}$ and $\boldsymbol{y}$ is the set

$$\boldsymbol{x} + \boldsymbol{y} \stackrel{\text{def}}{=} \big\{ x + y \in \mathbb{R} \mid x \in \boldsymbol{x} \quad \text{and} \quad y \in \boldsymbol{y} \big\} = [\, \underline{x} + \underline{y}, \;\; \overline{x} + \overline{y} \,]. \tag{3.2}$$

The *difference* of two intervals $\boldsymbol{x}$ and $\boldsymbol{y}$ is the set

$$\boldsymbol{x} - \boldsymbol{y} \stackrel{\text{def}}{=} \big\{ x - y \in \mathbb{R} \mid x \in \boldsymbol{x} \quad \text{and} \quad y \in \boldsymbol{y} \big\} = [\, \underline{x} - \overline{y}, \;\; \overline{x} - \underline{y} \,]. \tag{3.3}$$

The *product* of two intervals $\boldsymbol{x}$ and $\boldsymbol{y}$ is the set

$$\boldsymbol{x} \cdot \boldsymbol{y} \stackrel{\text{def}}{=} \big\{ xy \in \mathbb{R} \mid x \in \boldsymbol{x} \quad \text{and} \quad y \in \boldsymbol{y} \big\} = [\, \min S, \quad \max S \,], \tag{3.4}$$

$S = \big\{ \underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y} \big\}$. We sometimes write $\boldsymbol{x} \cdot \boldsymbol{y}$ more compactly as $\boldsymbol{x}\boldsymbol{y}$. Finally, the *quotient* of two intervals $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $0 \notin \boldsymbol{y}$ is the set

$$\boldsymbol{x}/\boldsymbol{y} \stackrel{\text{def}}{=} \big\{ x/y \in \mathbb{R} \mid x \in \boldsymbol{x} \quad \text{and} \quad y \in \boldsymbol{y} \big\} = \boldsymbol{x} \cdot \big(1/\boldsymbol{y}\big), \tag{3.5}$$

where

$$1/\boldsymbol{y} = \big\{ 1/y \in \mathbb{R} \mid y \in \boldsymbol{y} \big\} = [\, 1/\overline{y}, 1/\underline{y} \,].$$

Since all operation definitions (3.2), (3.3), (3.4), and (3.5) have the same general form, we can summarize them,

$$\boldsymbol{x} \odot \boldsymbol{y} \stackrel{\text{def}}{=} \big\{ x \odot y \in \mathbb{R} \mid x \in \boldsymbol{x} \quad \text{and} \quad y \in \boldsymbol{y} \big\}, \tag{3.6}$$

where $\odot$ stands for any of the four binary operations introduced. We rigorously know the range of values taken by $\odot$ because we know the endpoint formulae. Hence, we know rigorous bounds for the four binary operations.

Given a real-valued function f on the box $\boldsymbol{x}$ of dimension d, we would ultimately like to know the precise range of values taken by $f(x)$ as x varies through $\boldsymbol{x}$. In other words, we would like to find the image of the set $\boldsymbol{x}$ under the mapping f:

$$f\big(\boldsymbol{x}\big) \stackrel{\text{def}}{=} \big\{ f(x) \in \mathbb{R} \mid x \in \boldsymbol{x} \subseteq \mathbb{R}^d \big\}. \tag{3.7}$$

We sometimes denote $f\big(\boldsymbol{x}\big)$ in (3.7) by $\text{range}\big(f, \boldsymbol{x}\big)$.

Suppose that $\mathbb{I}P$ and $\mathbb{I}Q$ are interval spaces (sets of intervals on the real line), and $\boldsymbol{d} \in \mathbb{I}D \subset \mathbb{I}P \mapsto \boldsymbol{f}\big(\boldsymbol{d}\big) \subset \mathbb{I}Q$ is an operator defined on a domain $\mathbb{I}D$ in $\mathbb{I}P$ which takes on values in $\mathbb{I}Q$. The result of applying $\boldsymbol{f}$ to $\boldsymbol{x} \in \mathbb{I}D$ is an interval $\boldsymbol{y} \in \mathbb{I}Q$, denoted by $\boldsymbol{y} = \boldsymbol{f}\big(\boldsymbol{x}\big)$, and $\boldsymbol{f}$ is called an *interval-valued function or interval mapping*. An interval-valued function has an *interval domain*, if $\boldsymbol{z} \in \mathbb{I}D$ implies $\boldsymbol{x} \in \mathbb{I}D$ for each subinterval $\boldsymbol{x} \subset \boldsymbol{z}$ of $\boldsymbol{z}$ [82, Rall, p. 227].

An interval mapping $\boldsymbol{f}$ is an *interval extension* of f, if for degenerate interval arguments, $\boldsymbol{f}$ agrees with f,

$$\boldsymbol{f}\big([\,x, x\,]\big) = f\big(x\big). \tag{3.8}$$

In other words, if the arguments of $\boldsymbol{f}$ are replaced by degenerate intervals, then the left-hand side of (3.8) is a degenerate interval identified with the value of f at x. The interval extension may not evaluate to a degenerate interval causing (3.8) to fail. The interval extension of a given real-valued function is never unique [72, Exercise 5.8].

Obtain the particular *natural interval extension* $\boldsymbol{f}$ of a real-valued function f by (a) replacing the real variable x with an interval variable $\boldsymbol{x}$ and (b) the real arithmetic operations with corresponding interval operations.

An interval-valued function $\boldsymbol{f}$ on the box $\boldsymbol{x}$ of dimension d is *inclusion isotonic* if

$$\boldsymbol{y} \subseteq \boldsymbol{x} \Rightarrow \boldsymbol{f}(\boldsymbol{y}) \subseteq \boldsymbol{f}(\boldsymbol{x}).$$

The symbol $\subseteq$ is evaluated componentwise. An inclusion isotonic mapping [72, Moore, Kearfott, and Cloud, Definition 5.4] is exactly the same concept as an *inclusion monotone* function [82, Rall, Definition 2.2]. If $\boldsymbol{f}$ over the domain $\mathbb{I}\boldsymbol{x}$ is an inclusion isotonic interval extension of f, then $\boldsymbol{y} \in \mathbb{I}\boldsymbol{x}$ implies $f(\boldsymbol{y}) \subseteq \boldsymbol{f}(\boldsymbol{y})$, a statement called the *fundamental theorem of interval analysis*.

Taking $n = 2$, the interval operations (3.2), (3.3), (3.4), and (3.5) satisfy

$$\boldsymbol{y}_1 \subseteq \boldsymbol{x}_1, \quad \boldsymbol{y}_2 \subseteq \boldsymbol{x}_2 \Rightarrow \boldsymbol{y}_1 \odot \boldsymbol{y}_2 \subseteq \boldsymbol{x}_1 \odot \boldsymbol{x}_2. \tag{3.9}$$

Equation (3.9) can be used to prove polynomials of degree k are inclusion isotonic. A *rational interval function* is an interval-valued function whose values are defined by a specified finite sequence of interval arithmetic operations (3.6). All rational interval functions are inclusion isotonic [72, Lemma 5.1]. In particular, the natural interval extension of a real-valued polynomial of degree k is inclusion isotonic.

An *interval enclosure* of f on the box $\boldsymbol{T}_0$ is an inclusion isotonic interval-valued function $\boldsymbol{f}$ on the box $\boldsymbol{T}_0$, with $\boldsymbol{f}(\boldsymbol{T})$ defined for all $\boldsymbol{T} \subseteq \boldsymbol{T}_0$, having the property

that

$$f(t) \in \boldsymbol{f}(t) \quad \text{for all} \quad t \in \boldsymbol{T}_0. \tag{3.10}$$

Hence, $f(\boldsymbol{T}) \subseteq \boldsymbol{f}(\boldsymbol{T})$ for all $\boldsymbol{T} \subseteq \boldsymbol{T}_0$. The interval enclosure property is weaker than the interval extension property.

Given a real-valued function f on the box $\boldsymbol{x}_0$ of dimension d, suppose $\boldsymbol{f}$ is an inclusion isotonic interval extension of f with $\boldsymbol{f}(\boldsymbol{x})$ defined for $\boldsymbol{x} \subseteq \boldsymbol{x}_0$. Moore, Kearfott, and Cloud justify the definition of excess width with the following argument [72, Definition 6.4]. If $\boldsymbol{x}$, $\boldsymbol{y}$ in $\mathbb{I}\boldsymbol{x}_0$ are intervals satisfying $\boldsymbol{x} \subseteq \boldsymbol{y}$, then there is an interval $\boldsymbol{e}$ with $\underline{e} \leq 0 \leq \overline{e}$ such that $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ and $\text{wid}(\boldsymbol{y}) = \text{wid}(\boldsymbol{x}) + \text{wid}(\boldsymbol{e})$. Because $\boldsymbol{f}$ is an inclusion isotonic interval extension of f with $\boldsymbol{f}(\boldsymbol{T})$ defined for $\boldsymbol{T} \subseteq \boldsymbol{x}_0$, we have $\boldsymbol{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{e}(\boldsymbol{x})$ for some interval-valued function $\boldsymbol{e}(\boldsymbol{x})$ with $\text{wid}(\boldsymbol{f}(\boldsymbol{x})) = \text{wid}(f(\boldsymbol{x})) + \text{wid}(\boldsymbol{e}(\boldsymbol{x}))$. We call

$$\text{wid}(\boldsymbol{e}(\boldsymbol{x})) = \text{wid}(\boldsymbol{f}(\boldsymbol{x})) - \text{wid}(f(\boldsymbol{x}))$$

the *excess width* of $\boldsymbol{f}(\boldsymbol{x})$.

As an application of excess width, let's define *tight interval bound*. Suppose we have two inclusion isotonic extensions $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ of a real-valued function f on the interval $\boldsymbol{x}$. Then $\text{range}(f, \boldsymbol{x}) \subset \boldsymbol{b}_1 = \boldsymbol{f}_1(\boldsymbol{x})$ and $\text{range}(f, \boldsymbol{x}) \subset \boldsymbol{b}_2 = \boldsymbol{f}_2(\boldsymbol{x})$. An interval bound $\boldsymbol{b}_1$ with excess width $\boldsymbol{e}_1$ is tighter than the interval bound $\boldsymbol{b}_2$ with excess width $\boldsymbol{e}_2$ whenever $\text{wid}(\boldsymbol{e}_1(\boldsymbol{x})) < \text{wid}(\boldsymbol{e}_2(\boldsymbol{x}))$. Interval techniques identify a tight interval bound $\boldsymbol{b}$ such that $\text{range}(f, \boldsymbol{x}) \subseteq \boldsymbol{b}$ without appealing to the calculus.

For reasons that will become apparent, we write two algebraically equal real-valued

polynomials of degree k in the standard power basis and in Horner form:

$$p(t) = p_0 + p_1 t + p_2 t^2 + \cdots + p_k t^k, \quad t \in \boldsymbol{T}_0 \subseteq \mathbb{R}, \tag{3.11}$$

and

$$h(t) = p_0 + t\big(p_1 + t\big(p_2 + \cdots + t\big(p_k\big)\cdots\big)\big), \quad t \in \boldsymbol{T}_0 \subseteq \mathbb{R}. \tag{3.12}$$

Consider the respective natural interval extension polynomials $\boldsymbol{p}$ corresponding to (3.11) and $\boldsymbol{h}$ corresponding to (3.12) of degree k with $P_j \stackrel{\text{def}}{=} [\, p_j, p_j \,] = p_j$:

$$\boldsymbol{p}(\boldsymbol{t}) = P_0 + P_1\boldsymbol{t} + P_2\boldsymbol{t}\cdot\boldsymbol{t} + \cdots + P_k\boldsymbol{t}\cdot\boldsymbol{t}\cdots\boldsymbol{t}, \quad \boldsymbol{t} \subseteq \boldsymbol{T}_0,$$

and

$$\boldsymbol{h}(\boldsymbol{t}) = P_0 + \boldsymbol{t}\big(P_1 + \boldsymbol{t}\big(P_2 + \cdots + \boldsymbol{t}\big(P_k\big)\cdots\big)\big), \quad \boldsymbol{t} \subseteq \boldsymbol{T}_0. \tag{3.13}$$

The Horner form usually gives tighter estimates on $\text{range}\big(p, \boldsymbol{t}\big)$ and never gives worse estimates [72, p. 48].

We close with two examples. The following example [72, p. 38] illustrates a cause of *overestimation*:

Example 1. *Consider the real-valued mapping*

$$h\big(x\big) = x^2, \quad x \in \mathbb{R}.$$

If $\boldsymbol{x} = [\,\underline{x}, \overline{x}\,]$*, it is evident that the set*

$$h\big(\boldsymbol{x}\big) = \big\{x^2 \mid x \in \boldsymbol{x}\big\}$$

can be expressed as

$$h\big(\boldsymbol{x}\big) = \begin{cases} \big[\underline{x}^2, \overline{x}^2\big], & 0 \le \underline{x} \le \overline{x}, \\ \big[\overline{x}^2, \underline{x}^2\big], & \underline{x} \le \overline{x} \le 0, \\ \big[0, \max\big\{\underline{x}^2, \overline{x}^2\big\}\big], & \underline{x} \le 0 \le \overline{x}. \end{cases}$$

Certainly $[-1,1]^2 = [0,1]$*, whereas* $[-1,1] \cdot [-1,1] = [-1,1]$ *so that* $\boldsymbol{x}^2$ *is not the same as* $\boldsymbol{x} \cdot \boldsymbol{x}$*. However* $[-1,1]$ *does contain* $[0,1]$*. The overestimation when we compute a bound on the range of* $\boldsymbol{x}^2$ *as* $\boldsymbol{x} \cdot \boldsymbol{x}$ *is due to a phenomenon called interval dependency. Indeed, if we assume* $x \in \boldsymbol{x}$ *is an unknown number, then (a) when we form the product* $x \cdot x$*, the* x *in the second factor is exactly the same as the* x *in the first factor (b) in the interval product* $\boldsymbol{x} \cdot \boldsymbol{x}$*, it is assumed that the values in the first factor and the values in the second factor vary independently.*

The next example [72, p. 43] illustrates that interval extensions for algebraically equivalent real-valued mappings may not be equal.

Example 2. *Consider the real-valued function*

$$f\big(x\big) = x\big(1 - x\big), \quad x \in [0,1].$$

f is algebraically equal to the real-valued function

$$g\big(x\big) = x - x^2, \quad x \in [0,1].$$

Validate with calculus that $f([0,1]) = g([0,1]) = [0, 1/4]$. *Form interval-valued extensions of* f *and* g*:*

$$F\big(\boldsymbol{x}\big) = \boldsymbol{x} \cdot \big([\,1,1\,] - \boldsymbol{x}\big), \quad \boldsymbol{x} = [\,\underline{x}, \overline{x}\,] \subseteq [0,1],$$

and

$$G\big(\boldsymbol{x}\big) = \boldsymbol{x} - \boldsymbol{x}^2, \quad \boldsymbol{x} = [\,\underline{x}, \overline{x}\,] \subseteq [0,1].$$

Putting $\boldsymbol{x} = [0,1]$*, we compute that* $[-1,1] = G\big(\boldsymbol{x}\big) \neq F\big(\boldsymbol{x}\big) = [0,1]$. *Notice both cases overestimate* $\mathrm{range}\big(f, [0,1]\big) = [0, 1/4]$.

3.2 Taylor models

Rational interval functions and Taylor forms offer methods for the construction of inclusion isotonic interval-valued functions. The following assumption is required for Taylor forms:

Assumption 3. *Let* f *be over* $\boldsymbol{T}_0$*,* $f \in C^{k+1}\big(\boldsymbol{T} \cap \boldsymbol{T}_0, \mathbb{R}\big)$ *for any interval* $\boldsymbol{T} \in \mathbb{I}T_0$*, and* $\boldsymbol{f}^{(k+1)}$ *an interval inclusion of* $f^{(k+1)}$ *on* $\boldsymbol{T}$.

Intervals $\boldsymbol{T}$ and $\boldsymbol{T}_0$ are convex, and the intersection $\boldsymbol{T} \cap \boldsymbol{T}_0$ is also convex. On the other hand, we assume Assumption 3 to construct a Taylor form. The process starts with the interval version of Taylor's theorem from real analysis.

Theorem 4 (Taylor theorem). *Let Assumption 3 hold. Then, for* t *and* t_0 *in* $\boldsymbol{T} \cap \boldsymbol{T}_0$*,*

we have

$$f\big(t\big)-\sum_{j=0}^{k}\frac{1}{j!}f^{(j)}\big(t_0\big)\big(t-t_0\big)^j\in\frac{1}{\big(k+1\big)!}\boldsymbol{f}^{(k+1)}\big(\boldsymbol{T}\big)\big(\boldsymbol{T}-t_0\big)^{k+1}. \tag{3.14}$$

Rall [82, p. 288] proved the interval Taylor theorem. Recall the definition of Taylor model in (3.1). Assumption 3 is not explicit in the definition of Taylor model. On the other hand, the assumption is required to compute the interval Taylor bound in the Taylor theorem.

Under Assumption 3, the *(elementary) Taylor form of degree* k is

$$\boldsymbol{f}\big(\boldsymbol{T}\big)=\sum_{j=0}^{k}\frac{1}{j!}f^{(j)}\big(t_0\big)\big(\boldsymbol{T}-t_0\big)^j+\frac{1}{\big(k+1\big)!}\boldsymbol{f}^{(k+1)}\big(\boldsymbol{T}\big)\big(\boldsymbol{T}-t_0\big)^{k+1},\quad t_0\in\boldsymbol{T}\cap\boldsymbol{T}_0 \tag{3.15}$$

It follows from the Taylor theorem, that $\boldsymbol{f}$ defined by (3.15) is an interval enclosure of f over $\boldsymbol{T}_0$. The mapping f is defined by solving (3.14) for it. Indeed, rational interval functions are inclusion isotonic. The structure of f is rational. It follows that the natural interval extension $\boldsymbol{f}$ over the domain $\mathbb{I}\boldsymbol{T}_0$ is an inclusion isotonic interval extension of f. The fundamental theorem of interval analysis says $\boldsymbol{T}\in\mathbb{I}\boldsymbol{T}_0$ implies $f\big(\boldsymbol{T}\big)\subseteq\boldsymbol{f}\big(\boldsymbol{T}\big)$. Moreover $t\in\boldsymbol{T}$ implies $f(t)\in f\big(\boldsymbol{T}\big)$. All together, we have

$$f\big(t\big)\in f\big(\boldsymbol{T}\big)\subseteq\boldsymbol{f}\big(\boldsymbol{T}\big) \tag{3.16}$$

for all $t\in\boldsymbol{T}\subseteq\boldsymbol{T}_0$. Finally, (3.16) and $\boldsymbol{f}$ inclusion isotonic prove that $\boldsymbol{f}$ is an interval enclosure of f over $\boldsymbol{T}_0$. Taylor models often, but not always, reduce excess width as the degree of the Taylor polynomial increases.

3.3 Computing the supremum norm

Under Assumption 3, we have a real-valued mapping f over $\boldsymbol{T}_0$. We compute the Taylor polynomial p denoting the polynomial of the Taylor form (3.14), and a rigorous interval bound $\boldsymbol{r} = [\,\underline{r}, \overline{r}\,]$ of the remainder term of the Taylor form. Then $f(t) - p(t) \in \boldsymbol{r} = [\,\underline{r}, \overline{r}\,]$ for all $t \in \boldsymbol{T}_0$. We seek a tight interval bound $\boldsymbol{b}$ on the supremum norm of a Taylor model,

$$\|p\|_\infty = \sup_t |p\big(t\big)| \in \boldsymbol{b}. \tag{3.17}$$

Then, on $\boldsymbol{T}_0$, we have

$$\max\big\{0, \underline{b} - \mathrm{abs}\big(\boldsymbol{r}\big)\big\} \le \|f\|_\infty \le \overline{b} + \mathrm{abs}\big(\boldsymbol{r}\big), \tag{3.18}$$

The bound we use is $\delta := \overline{b} + \mathrm{abs}\big(\boldsymbol{r}\big)$. We have two rigorous options to obtain $\boldsymbol{r}$ which use interval techniques (not calculus).

Interval evaluation uses Assumption 3. The natural interval extension of f is an inclusion isotonic interval-valued function. Set $\boldsymbol{r} = f\big(\boldsymbol{T}_0\big)$. This estimate is known to overestimate. To reduce overestimation with this approach, use the Taylor polynomial p in place of the real-valued mapping f. SOLLYA implements Horner's algorithm to help eliminate excess width. This is the first approach.

The SOLLYA package has an internal method that computes a tighter interval bound on the Taylor polynomial p than interval evaluation, the SOLLYA supremum norm method. This method is described in Joldes [61]. This is a second approach.

We also acknowledge the point evaluation approach, which is not rigorous. Sample p or f at points in $\boldsymbol{T}_0$.

3.4 Class Tmodel

SOLLYA does the heavy lifting in our C++ class `Tmodel` where we extend SOLLYA functionality using two C++ language features: generic templates and operator overloading. Use class `Tmodel` to obtain a degree k Taylor model of a $k+1$ times differential mapping $x \in \mathbb{R} \mapsto f\big(x\big) \in \mathbb{R}$.

We built a convenient C++ class `Tmodel` to access SOLLYA. Let's explore it. We illustrate our SOLLYA interface with the exponential mapping centered at $t_0 = 0$

$$\exp\big(-t\big) \in 1 - t + \frac{1}{2}t^2 - \frac{1}{6}t^3 + \frac{1}{24}t^4 - \frac{1}{120}[1, \exp\big(\frac{\pi}{2}\big)]t^5, \quad t \in [0, \frac{\pi}{2}] \tag{3.19}$$

and the sin mapping centered at $t_0 = 0$

$$\sin\big(t\big) \in t - \frac{1}{6}t^3 + \frac{1}{120}[0,1]t^5, \quad t \in [0, \frac{\pi}{2}]. \tag{3.20}$$

We consider the Taylor models returned by SOLLYA for exp, sin, exp + sin, and exp · sin over $\boldsymbol{T}_0 = [0, \frac{\pi}{2}]$ of degree 4, 5, 6, and 7. Here, the sum and product of Taylor models is an example of Taylor model arithmetic.

We must always initialize the software in `main` as follows:

```
#include <cstddef>
#include "Tmodel.hpp"
#include "SOLLYA/sollya.h"

int main()
{
  sollya_lib_init();
  /* Do something */
  sollya_lib_close();
  return 0;
}
```

The SOLLYA tool precision is measured in bits, and 64 bits is about double precision.

```
// Set tool precision for double
Tmodel tmp;
tmp.setToolPrecision(64);
```

Set the Taylor polynomial and its degree, the expansion point t_0, and the interval domain $[t_0, t_1]$.

```
int k(2);
Tmodel p(k);

// Expand around t0, and work with interval [t0,t1]
double t0 = 0.0;
double t1 = 1.0;

// Coefficient of polynomial of x
double pC[] = {1.0, 1.0};

p.setPoly(k, pC, t0, t1);
p.setBound(0.0,0.0);
```

In the final line, the snippet initializes the remainder bound for the Taylor model. Compute the bound $\boldsymbol{b}$ as defined in (3.17) with the `Tmodel` method `computeSharpBound` which implements interval evaluation. Print the SOLLYA data structure.

```
p.computeSharpBound();
printf("\nTaylor model for p\n");
p.print();
```

Compute the bound $\boldsymbol{b}$ with the `Tmodel` method supnorm.

```
double bL(0.0), mid(0.0), bR(0.0);
p.supnorm(bL,mid,bR);
bound = fmax(fabs(bL), fabs(bR));
```

We plot output for our exp and sin examples. Arithmetic operations with Taylor models result in Taylor models. Take addition for example,

$$f(t) - p_1(t) \in \boldsymbol{r}_1, \quad g(t) - p_2(t) \in \boldsymbol{r}_2$$

results in

$$\big(f(t) + g(t)\big) - \big(p_1(t) + p_2(t)\big) \in \big(\boldsymbol{r}_1 + \boldsymbol{r}_2\big).$$

We plot enclosures for our examples. The upper blue curve in each plot is the graph of the Taylor polynomial plus the bound, the lower curve is in each plot is the graph of the Taylor polynomial minus the bound, the red curve is the graph of the Taylor polynomial. The green curve is the graph of the true mapping. The upper-left plot always corresponds to the mapping $t \in [0, \pi/2] \mapsto \sin\big(t\big)$. The upper-right plot always corresponds to the mapping $t \in [0, \pi/2] \mapsto \exp\big(-t\big)$. The lower-left plot always corresponds to the mapping $t \in [0, \pi/2] \mapsto \sin\big(t\big) + \exp\big(-t\big)$. The lower-right plot always corresponds to the mapping $t \in [0, \pi/2] \mapsto \sin\big(t\big) \cdot \exp\big(-t\big)$. Figure 3.1 renders four Taylor model plots of degree 4. Figure 3.2 renders four Taylor model plots of degree 5. Figure 3.3 renders four Taylor model plots of degree 6. Figure 3.4 renders four Taylor model plots of degree 7. Notice the graphs in each plot converge as degree increases. This was the desired illustration.

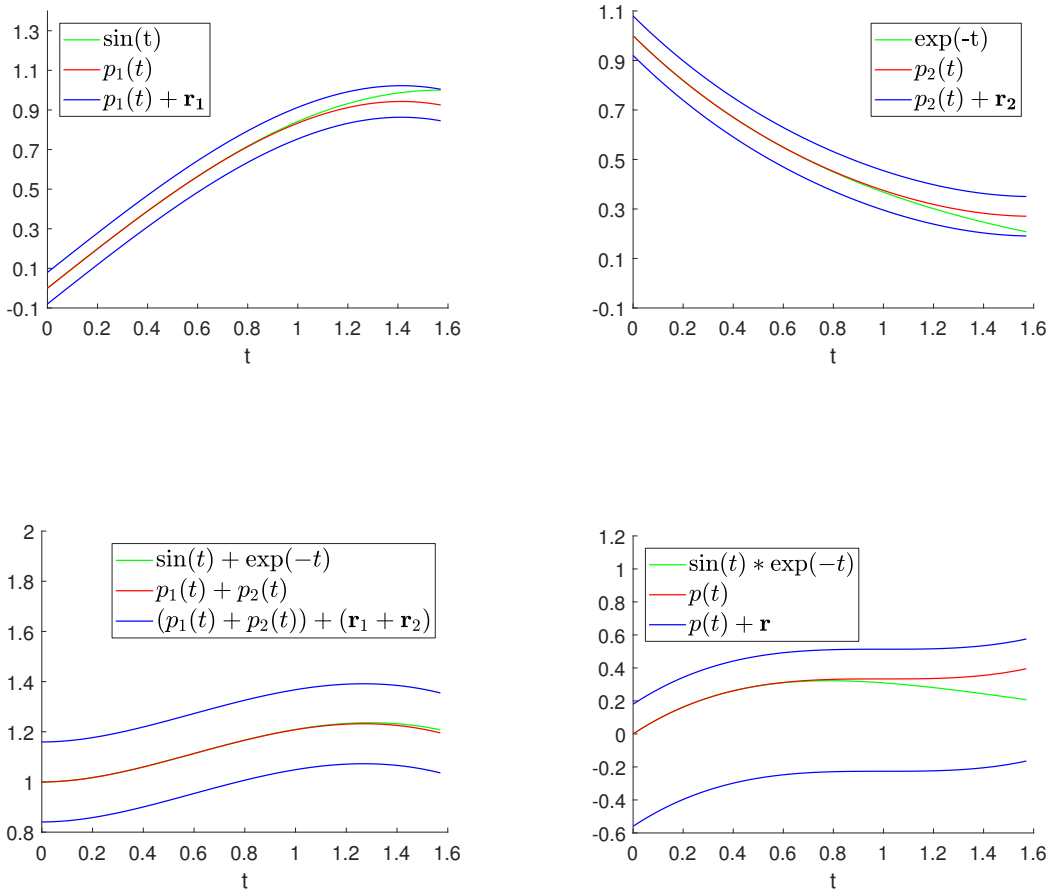


Figure 3.1: Taylor model plots of degree 4.

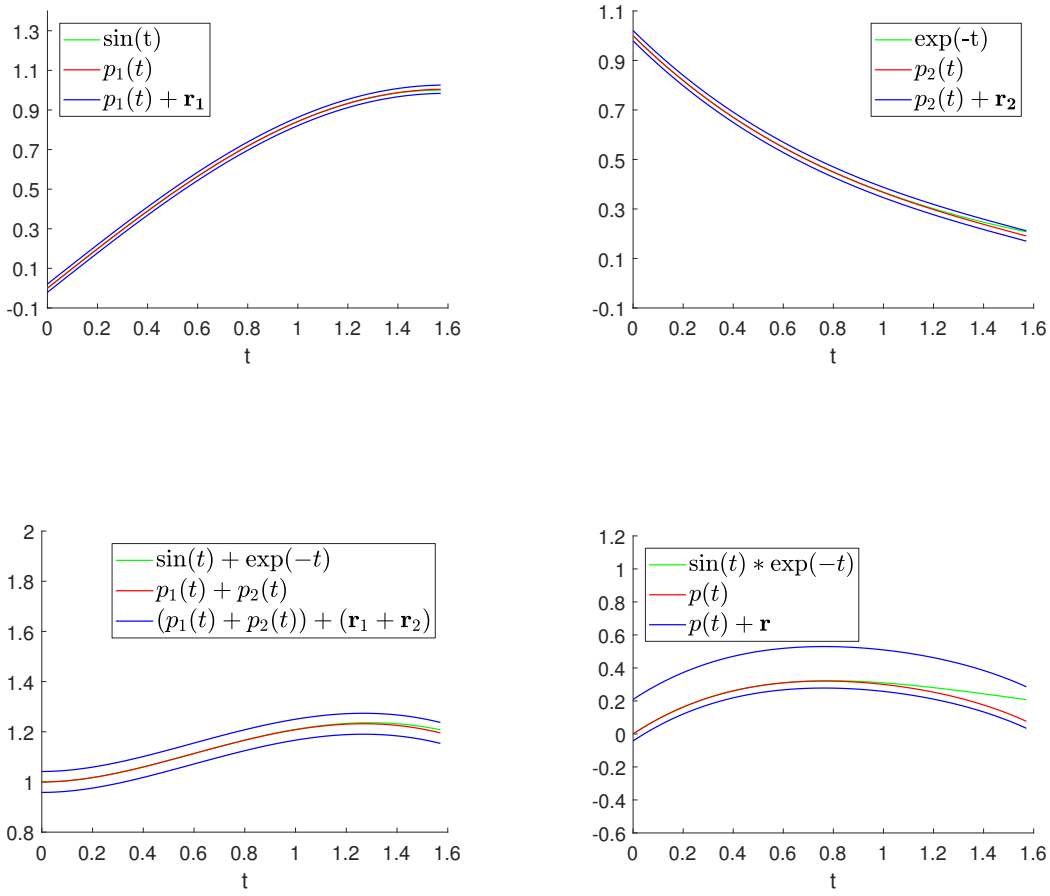


Figure 3.2: Taylor model plots of degree 5.

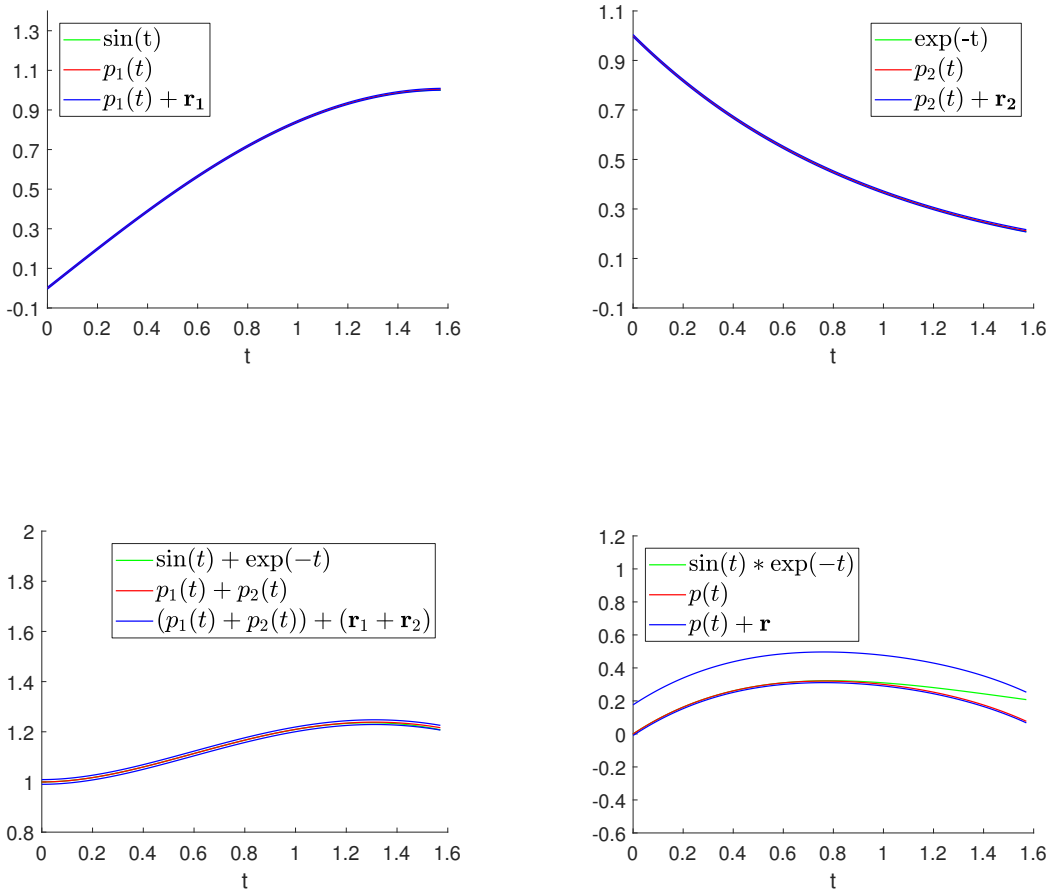


Figure 3.3: Taylor model plots of degree 6.

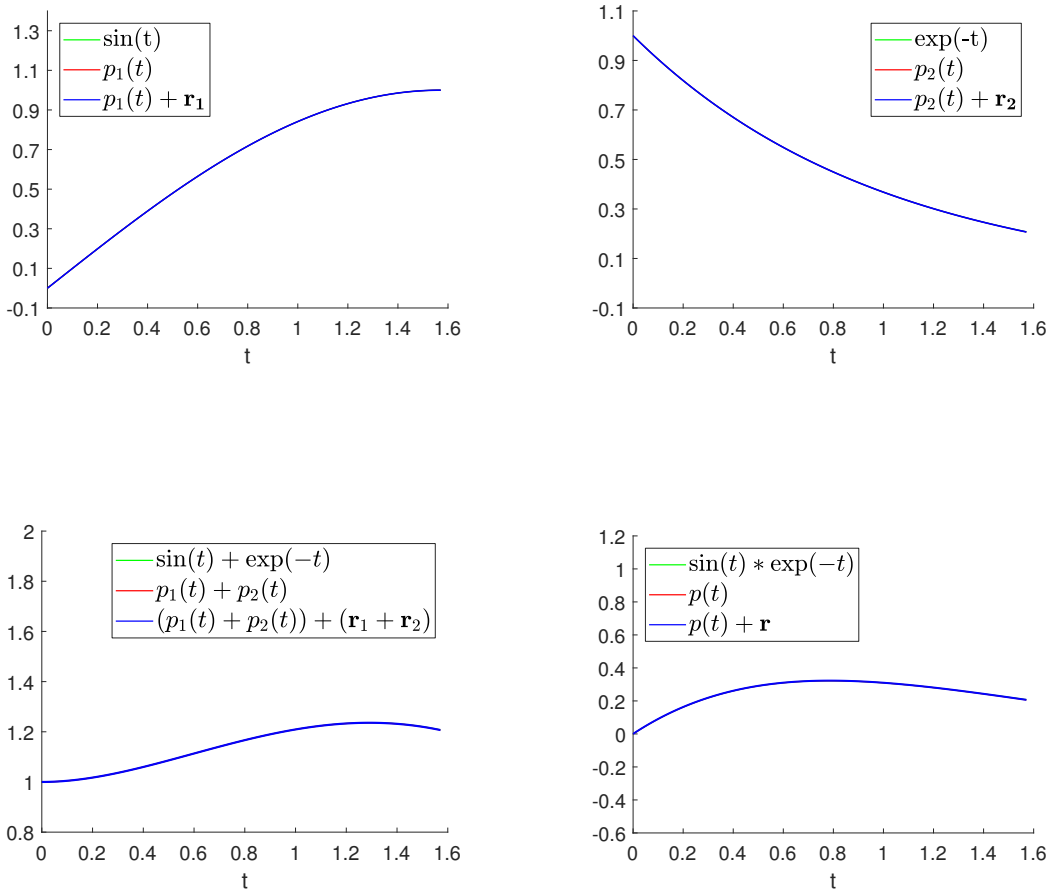


Figure 3.4: Taylor model plots of degree 7.

Chapter 4

Automating rigorous defect control

Automating rigorous defect control means implementing the three phases introduced in Chapter 1 for computing a global approximate solution. We discuss our software design choices. Phase II relies on Taylor models implemented in the SOLLYA package [12, 13] through our interface class `Tmodel`, we discussed class `Tmodel` in Chapter 3. Our software ODETS combines our C++ classes into a numerical ODE initial-value problem solver which automates rigorous defect control.

Standard ODE-IVP solvers control local forward error on each step, but this error control can be deceived when the error estimate is smaller than the true forward error. Validated ODE solutions use interval methods to compute rigorous bounds on the numerical solution, but it is challenging to keep the rigorous bound tight. The software package VNODE implements this approach [73, 75, 79].

4.1 Input and driver

We list a fully functional driver to ODETS in Figure 4.1 for solution of the Predator-Prey example. The ODETS Predator-Prey driver requires setup, let's explain the input. Call ODETS with the interface method `solve` which requires several input parameters. Of course, the initial time `t`, the dimension of the problem `d`, the initial point `x`, and the final time `tout` are required. The `nprt` parameter requests output at this number of points per integration step. The `tprt` parameter is ignored unless `nprt` is zero. Otherwise `tprt` is a constant interval width between printed output. The first two parameters are the two different types of the vector field: a TADIFF type and a `Tmodel` type.

The SOLLYA verbosity parameter is defined in the SOLLYA documentation [12, 13], and it controls SOLLYA output. Set SOLLYA verbosity to zero to eliminate all SOLLYA chatter. The SOLLYA precision parameter is set to 64 bits for double precision.

4.2 Phase I: Compute an approximate solution

Given the initial condition $x_{n-1} \in \mathbb{R}^d$ at $t_{n-1} \in \mathbb{R}$, stepsize h_n, and order q, take a step to $t_n = t_{n-1} + h_n$ with the Taylor series method as the engineered problem; that is, compute a Taylor series approximate solution v of degree $k \stackrel{\text{def}}{=} q + 2$

$$v\big(t\big) = x_{n-1} + (x_{n-1})_1(t - t_{n-1}) + \cdots + (x_{n-1})_k(t - t_{n-1})^k \tag{4.1}$$

to the local ODE-IVP reference problem,

$$x'\big(t\big) = f\big(t, x\big(t\big)\big), \quad x\big(t_{n-1}\big) = x_{n-1}, \quad t \in [t_{n-1}, t_n].$$

```
#include <cstddef>
#include "sollya.h"
#include "odets.hpp"

#define DIM 2

template <typename T>
void rhs(T t, const T * x, T * xp)
{
  xp[0] = x[0] -  0.1*x[0]*x[1] +  0.02*t;
  xp[1] =-x[1] + 0.02*x[0]*x[1] + 0.008*t;
}

int main(int argc, char* argv[])
{
  // Set output style and precision
  std::cout.precision(17);
  std::cout << std::scientific;

  sollya_lib_init();
  sollya_obj_t zer = sollya_lib_constant_from_int(0);
  sollya_lib_set_verbosity(zer);
  sollya_obj_t prercision = sollya_lib_constant_from_int(64);
  sollya_lib_set_prec(precision);

  double tol = atof(argv[1]);
  double order = atof(argv[2]);
  int d(DIM);
  double * x = new double[DIM];

  int nprt(50);
  double t(0.0), tout(40.0), tprt(0.001);

  x[0] = 30.0;
  x[1] = 20.0;

  solve(rhs, rhs, t, d, x, tout, order, tol, nprt, tprt);

  delete [] x;
  sollya_lib_close();

  return 0;
}
```

Figure 4.1: Predator-Prey driver to ODETS

The vector field f need not be analytic to compute a Taylor series approximate solution. It does need to be $k+1$ times continuously differentiable to compute the coefficients $(x_{n-1})_r$ in (4.1) and rigorously bound the remainder term. The consistency (or accuracy) order q has the traditional meaning [2, p. 40].

We rely on SOLLYA to evaluate the Taylor and the Hermite approximate solution. Internally SOLLYA rigorously evaluates polynomials.

Taylor series methods are based on Taylor arithmetic [11], and they are thought to be computationally expensive. This is not the case, but they are still much more expensive than Runge–Kutta. With a judicious implementation, Taylor arithmetic efficiently computes the Taylor coefficients $(x_{n-1})_j$ at t_{n-1}.

Bendtsen and Stauning wrote automatic differentiation packages FADBAD++ and TADIFF [5, 6, 92] in the C++ programming language which efficiently compute Taylor coefficients. The flexibility of FADBAD++ and TADIFF derives from operator overloading and generic programming. Automatic differentiation relies on the computational graph of the vector field f. This computational graph is computed only once. Once computed, the computational graph is called repeatedly to automatically generate the Taylor series coefficients at different expansion points.

A tutorial explanation of automatic differentiation and Taylor series methods is found in the introductory book Moore, Kearfott, and Cloud [72, Section 9.3] as well as the tutorial article Chang and Corliss [11]. Baydin, Pearlmutter, Radul, Siskind [4] offer a recent and advanced article on this topic, it details the forward and the reverse modes.

A Taylor series approximate solution requires only floating-point arithmetic, and automatic differentiation does not faithfully reproduced the true Taylor coefficients.

The Taylor coefficients are possibly tainted with rounding errors [3, 14]. In defect control, even an approximate solution based on tainted data is the exact solution of the perturbed reference problem.

We wrote class `TaylorExpansion`, a C++ interface to TADIFF and a data management class. It generates the Taylor coefficients for our Taylor series method. Our class handles vector valued ODE-IVP.

We assume a generic template implementing the ODE function. In Section 4.1, we listed our driver for the Predator-Prey application where the vector field is codified as follows:

```
template <typename T>
void fcn( T t, const T * x, T * xp )
{
  xp[0] =  x[0] -  0.1*x[0]*x[1] +  0.02*t;
  xp[1] = -x[1] + 0.02*x[0]*x[1] + 0.008*t;
}
```

The vector field need not be polynomial, but this vector field is polynomial. With `tlast` $= t_{n-1}$ and `xlast` $= x_{n-1}$, a typical application will require the following method calls from class `TaylorExpansion`:

```
  TaylorExpansion * tvf = new TaylorExpansion(d, k);
  tvf->setCodeList(fcn);
  tvf->getAllTaylorCoefficients(tlast, xlast, d, k, coeff);

  // Get the last coefficient
  tvf->getTaylorCoeff(k-1, d, k, coeff, last);
```

The $d \times k$ dimensional array `coeff` stores all of the Taylor series coefficients, and the d-dimensional array `last` stores the kth degree Taylor series coefficients.

We make a design choice: The defect at the Taylor series approximate solution (4.1) will have degree k, just as the Taylor series approximate solution has degree k. Our class `ApproximateSolution` manages the d-dimensional, vector valued approximate solution as well as its first derivative and the defect.

The defect at the Taylor series approximate solution v is not continuous, and, technically, it is a deviation. The Hermite approximate solution u with $x_n = v(t_n)$ interpolates $\big(t_{n-1}, x_{n-1}\big)$, $f\big(t_{n-1}, x_{n-1}\big)$, $\big(t_n, x_n\big)$, and $f\big(t_n, x_n\big)$. Compute the Hermite approximate solution,

$$u\big(t\big) = v\big(t\big) + \frac{\Delta v\big(t_n\big)}{h_n^k}\big(t - t_{n-1}\big)^{k+1} - \frac{\Delta v\big(t_n\big)}{h_n^{k+1}}\big(t - t_{n-1}\big)^{k+2}. \tag{4.2}$$

This ensures $\Delta u\big(t_{n-1}\big) = \Delta u\big(t_n\big) = 0$. Hence, the defect at the Hermite approximate solution is continuous.

4.3 Phase II: Bound the defect

We bound the defect. To do so, we implement the tight interval bound on the supremum norm of a Taylor model from Section 3.3.

The approximate solution can be any smooth path. We are interested in the Taylor series approximate solution or the Hermite approximate solution introduced in the previous section. An approximate solution u will always be the exact solution of the perturbed reference problem, even an approximate solution tainted with floating–point errors. It follows that the error bound of the approximate solution is zero, and the Taylor model of the approximate solution is known to mathematical certainty. We use the SOLLYA package to evaluate the code list of $x' - f(t, x)$ with $\big(u, [0, 0]\big)$ in Taylor

model arithmetic. The ith component of the result is the Taylor model $\big(p_i, \boldsymbol{r}_i\big)$ so that, by definition, we have

$$\Delta u_i\big(t\big) - p_i\big(t\big) \in \boldsymbol{r}_i \qquad \text{for all } t \in [\, t_{n-1}, t_n \,]$$

Compute (3.17), a rigorous enclosure $\boldsymbol{b}_i = [\, \underline{b_i}, \overline{b_i} \,]$ of the supremum norm:

$$\|p_i\|_\infty = \sup_{t \in [\, t_{n-1}, t_n \,]} |p_i\big(t\big)| \in \boldsymbol{b}_i$$

We have by (3.18)

$$\max\big\{0, \underline{b_i} - \mathrm{abs}\big(\boldsymbol{r}_i\big)\big\} \le \|\Delta u_i\|_\infty \le \overline{b_i} + \mathrm{abs}\big(\boldsymbol{r}_i\big).$$

The bound we use is

$$\delta_i \stackrel{\text{def}}{=} \overline{b_i} + \mathrm{abs}\big(\boldsymbol{r}_i\big). \tag{4.3}$$

Call the method `getSupNorm` in class `Tmodel` to compute the upper bound (4.3) for the defect. We apply the built-in SOLLYA supremum norm to compute a rigorous $\boldsymbol{b}_i$. The rigorous bound $\boldsymbol{r}_i$ is computed as the Taylor model for the defect is computed by SOLLYA. We can obtain $\boldsymbol{r}_i$ with interval evaluation or SOLLYA through interval techniques (not calculus) as we discussed in Section 3.3. ODETS uses SOLLYA to compute (4.3) through its supremum norm method.

It may seem curious that we computed the Taylor model for $x' - f(t, x)$ only to bound the result, but the purpose of computing this Taylor model is indeed to compute a rigorous tight bound (4.3). Recall that direct interval evaluation of the

defect $x' - f(t,x)$ will in general lead to overestimation.

In closing of this section, consider the logistic model problem.

Example 1. *Consider the logistic model*

$$x' = f(x) = x - x^2, \quad x(0) = 0.2, \quad t_1 = 0.4.$$

Compute the Taylor series,

$$v(t) = 0.2 + 0.16t + 0.048t^2 + 1.0667{\times}10^{-3}t^3.$$

The first three Taylor coefficients are exact, but the last coefficient is rounded to 4 digits. Interpolating $f\big(v(t_1)\big)$, rounded here to 4 digits, find

$$u(t) = v(t) + 1.5795{\times}10^{-2}t^4 - 3.9486{\times}10^{-2}t^5.$$

Evaluating $x' - (x - x^2)$ with $(u, [0,0])$, compute $\Delta u(t) - p(t) \in \boldsymbol{r}$,

$$\begin{aligned} p(t) &= 1.3878{\times}10^{-17}t + 1.0000 \times 10^{-10}t^2 \\ &\quad +7.7898{\times}10^{-2}t^3 - 2.0426{\times}10^{-1}t^4 + 2.8849{\times}10^{-2}t^5, \\ \boldsymbol{r} &= [-5.1923{\times}10^{-5}, 1.8090{\times}10^{-17}]. \end{aligned}$$

We plot the enclosures in Figure 4.2. The lower enclosure is $\Delta\underline{u}(t) = p(t) + \underline{r}$, and the upper enclosure is $\Delta\overline{u}(t) = p(t) + \overline{r}$. Similar identifications hold for Δv. The shape of the deviation at v and the defect at u is typical for each local step. Here δ shows the tight bound we use for $\|\Delta u\|_\infty$. We mark this bound for the Hermite approximate

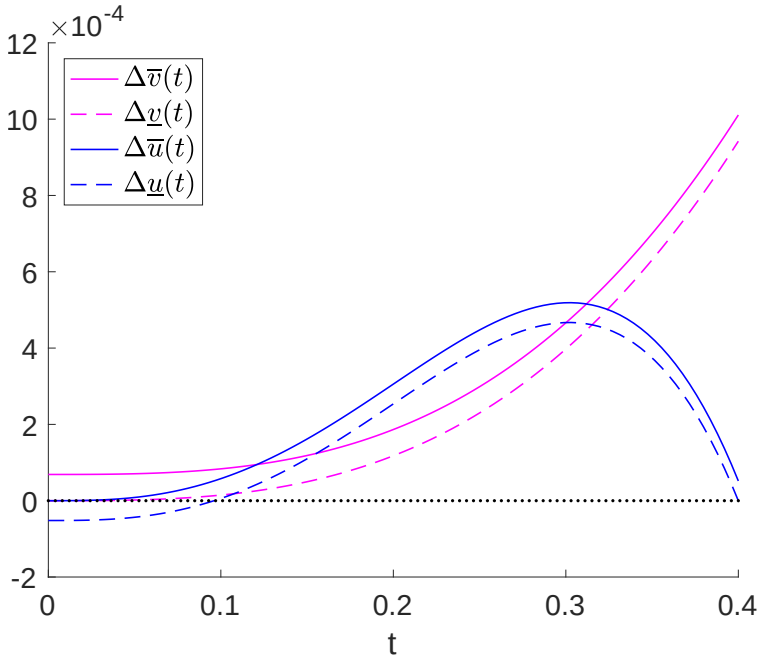


Figure 4.2: Enclosures of $\Delta v(t)$ and $\Delta u(t)$

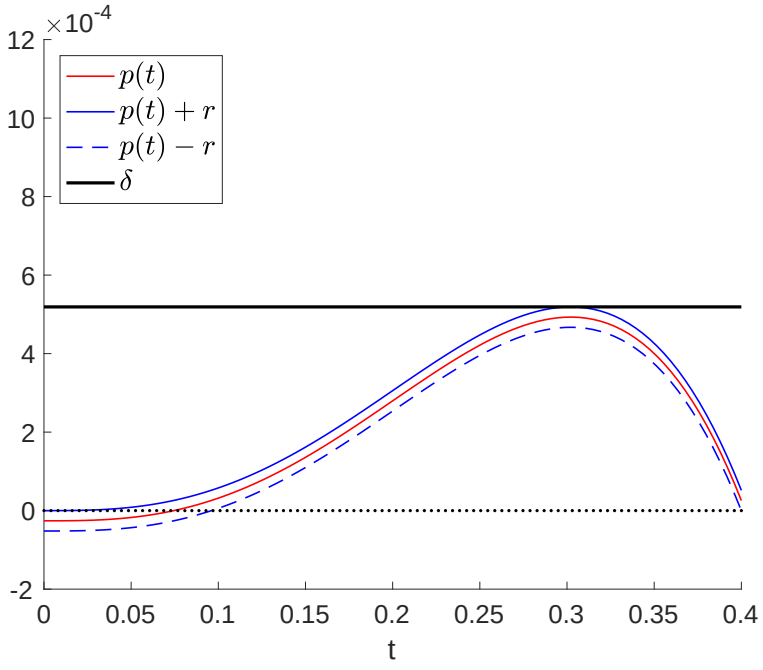


Figure 4.3: Enclosures of $\Delta u(t)$ with bound

solution in Figure 4.3.

Let's consider interval arithmetic evaluation of the vector field. We expect overestimation of the rigorous bound. Evaluating $\Delta u = u' - (u - u^2)$ *in interval arithmetic (3.9) gives*

$$\begin{aligned} u([0,0.4]) \in \boldsymbol{u} &= [0.2000, 0.2722] \\ u'([0,0.4]) \in \boldsymbol{u'} &= [0.1599, 0.2030] \\ \Delta u \in \boldsymbol{u'} - \left(\boldsymbol{u} - \boldsymbol{u}^2\right) &= [-0.0722, 0.0771] \end{aligned}$$

While interval arithmetic evaluation is inexpensive to compute, the width can grow unbounded for complicated f*'s. Find* $\operatorname{wid}\left([-0.0722, 0.0771]\right)$ *is* 1.4917×10^{-1}*, but* $\operatorname{wid}\left(\boldsymbol{r}\right)$ *is* 5.1923×10^{-5}*. In this example, observe that Taylor models keep bounds tight.*

4.4 Phase III: Accept/reject step

A step h_n is accepted or rejected based on the decision logic of an adaptive time-stepping algorithm. The elementary controller was considered in Section 2.7, and we implement this adaptive time-stepping algorithm in ODETS.

Given a user specified tolerance TOL, we assume that $\delta_{\max} \leq \text{TOL}$, and we assume that the Taylor remainder bound $\boldsymbol{r}_i$ is in the interval $[0, \text{TOL}]$ for all components. In this case, the step is accepted, and we predict the next step with the elementary controller

$$h_{n+1} \leftarrow 0.9 h_n \left(\frac{0.5\text{TOL}}{\delta_{\max}} \right)^{1/q}$$

else, we reject the step. If a step is rejected, we recompute $\delta_{\max}$ with stepsize

$$h_n \leftarrow h_n \left(\frac{0.25\text{TOL}}{\delta_{\max}} \right)^{1/q}$$

that is, we repeat from Phase I. This involves computing a new Hermite approximate solution based on the old Taylor approximate solution, and reevaluating $x' - f(t, x)$ in Taylor model arithmetic. The Hermite approximate solution depends on the stepsize, but the Taylor coefficients need not be recomputed.

This phase of the algorithm validates the supremum norm of the defect is less than TOL, and it validates the width of the Taylor model bound is less than TOL. The q is the consistency order of the Taylor series method.

4.5 Initial stepsize

Watts [96] provides a history on the topic of initial stepsize selection, and his selection algorithm is based on a second order Taylor series. Gladwell, Shampine, and Brankin [39] also developed an automatic initial stepsize selection, and they demonstrate that their initial stepsize selection leads to a robust and efficient integration. The Taylor formula of degree k is

$$x\big(t_0 + h\big) \approx x\big(t_0\big) + x'\big(t_0\big)h + x''\big(t_0\big)\frac{h^2}{2!} + \cdots + x^{(k)}\big(t_0\big)\frac{h^k}{k!}. \tag{4.4}$$

The local error of formula (4.4) is asymptotically

$$E_k = \|x^{(k+1)}\big(t_0\big)\frac{h^{k+1}}{(k+1)!}\|_\infty. \tag{4.5}$$

The authors [39, p. 179] postulate that the norm of the local error of any one-step method of order $p \geq k$ can be approximated by $E_k^{(p+1)/(k+1)}$. Applying this postulate, the largest step size H which leads to a local error (4.5) no larger than the given tolerance TOL is given by

$$|H| = \left((k+1)!/\|x^{(k+1)}(t_0)\|_\infty\right)^{1/(k+1)} \mathrm{TOL}^{1/(p+1)}. \tag{4.6}$$

Stepsize (4.6) is attributed in [39] to private discussions with H. J. Stetter. On the other hand, Gladwell, Shampine, and Brankin [39, p. 175] point out the virtues and pitfalls in this selector.

4.6 ODETS software

Our C++ software ODETS integrates unit tested modules implementing Phase I to Phase III. ODETS implements the Corless and Corliss proposed rigorous defect control algorithm, Algorithm 1. The user provides the input and driver as explained in Section 4.1. We consider the design of the interface for ODETS.

Before we dive into our software implementation, let's demonstrate that it works. Consider again the defect controlled logistic equation $x' = x - x^2$ with final time $t_{\mathrm{end}} = 1000$. The result is in Figure 4.4. The defect is rigorously controlled less than TOL, and the stepsizes illustrate that they are controlled for numerical stability.

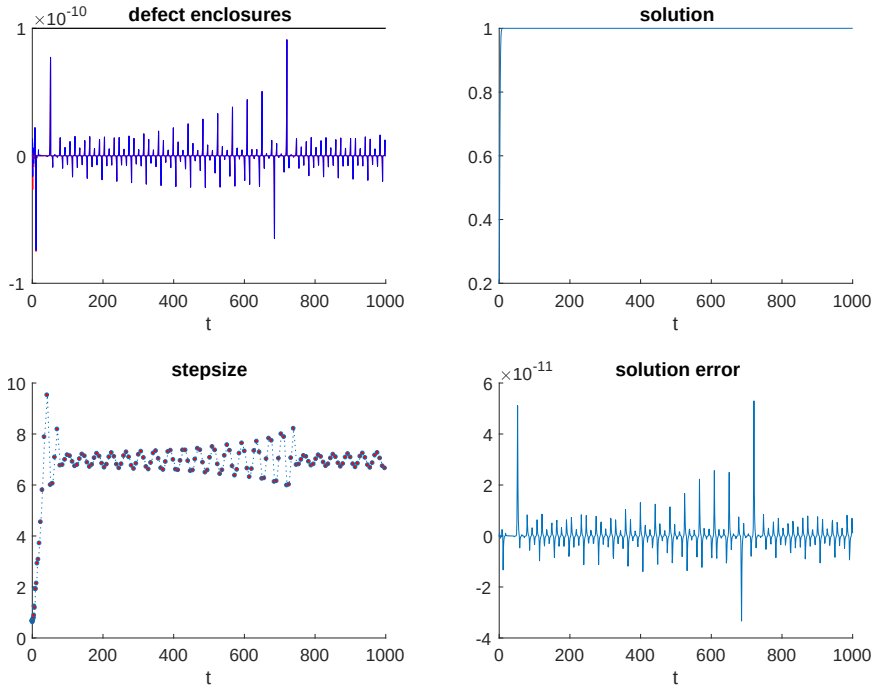


Figure 4.4: $x' = x - x^2$, $x(0) = 0.2$, order 15, TOL $= 10^{-10}$

4.6.1 Build requirements

We build ODETS under Ubuntu Linux. This requires the gcc compiler, google-test for unit testing, CMake for automated validation, and several Ubuntu packages required by SOLLYA. An ODETS build depends on the software:

- SOLLYA [13] for computing Taylor models and supremum norm;
- FADBAD++/TADIFF [92] for computing Taylor coefficients.

4.6.2 Requirements and Specification by Example

This project followed Test-Driven Development (TDD) and unit testing practices, Langr's book [64] offers a helpful tutorial guide to software craftsmanship through TDD and design patterns. The author reached out to Langr [65] for further insight

into requirements, and his reply assisted the following discussion.

Advising sessions translate into test scenarios. The author wrote unit tests with Google Test [41]. Following Specification by Example practices [1], the author was enabled to elevate unit tests to specify system behavior by grouping unit tests to reflect test scenarios. The *nix utility CMake implemented automated validation without changing specification. In effect, we get executable specifications. If the tests all pass, then the product owner accepts the product.

These executable specifications are the central negotiating point with the product owner, here my thesis committee. They describe end-to-end capabilities in the system, and they describe all the core intended goals that people have for the system. They are also the artifact describing best the capabilities of the system requested by the customer. In one sense, they are living documentation. As long as all these tests pass, we know the system does what the tests describe.

Our requirement aim is simple enough to state. We aim to deliver a rigorous defect control ODE-IVP solver based on the Taylor series method that integrates seamlessly into problem solving environments (MATLAB).

4.6.3 Algorithm overview

ODETS is a collection of C++ methods implementing rigorous defect control.

The stepping algorithm: The rationale for rigorous defect control proposed by Corless and Corliss is implemented in the ODETS method `solve`. It tightly follows Algorithm 1. We list an abbreviated illustration of `solve` in Figure 4.5 with actual C++ snippets.

Input: Integration starts at the last known continuation point `tlast`, `xlast`, `d` with `xlast` in $\mathbb{R}^d$, final time `tend`, `order`, tolerance `tol`, and output specified by `nprt` and `tprt` as in Section 4.1. The first two parameters are for the template function `f`, the first `f` is type TADIFF and the second `f` is type `Tmodel`.

Initial stepsize: The initial stepsize H with $p = k$ from (4.6) is used in ODETS. The stepsize may be restricted to limit integration to the interval $[t_0, t_{\text{end}}]$. The main advantage of this initial stepsize is that all the required information to compute this initial stepsize is available from other necessary computations.

Minimal stepsize: We take a simple approach to the minimum stepsize. The adaptive time-stepping algorithm is evaluated in double precision: The minimal stepsize for any integration step is $t\big(1 + 4\varepsilon\big)$ where ε is the machine epsilon in the current precision.

Output: The SOLLYA print facility can be called on component i of the solution:

```
sol->s[i].print(); sol->p[i].print(); sol->d[i].print();
```

We designed `solout` listed in Figure 4.6 for ODETS output to support data collection. This listing reveals the structure of the output. The variable `err` holds the supremum norm of the defect of the Hermite approximate solution, and `posneg` is the sign of the defect on the current subinterval $[t_{n-1}, t_n]$.

Stepsize control permits efficient integration, and discourages closely spaced output points. The input parameters `nprt` and `tprt` defined and described in Section 4.1 enables ODETS to evaluate the Hermite and Taylor approximate solution at equispaced

```
int num_taylor_coeff  = order + 2;
int num_hermite_coeff = num_taylor_coeff + 2;
// Data structure to hold Taylor and Hermite coefficients
ApproximateSolution * taylor =
  new ApproximateSolution(d, num_taylor_coeff);
ApproximateSolution * hermite=
  new ApproximateSolution(d, num_hermite_coeff);

TaylorExpansion * tvf = new TaylorExpansion(d, num_hermite_coeff);
tvf->setCodeList(tadiff_fcn);
tvf->getAllTaylorCoefficients(tlast, xlast, d, num_hermite_coeff,
   coeff);
// Get the last coefficient
tvf->getTaylorCoeff(num_taylor_coeff-1, d, num_hermite_coeff, coeff,
                    last_taylor_coeff);
// Estimate first step, it can be no larger than htrial = tend-last
while( tlast < tend )
{
  // Step and error computation
  bool accepted = false;
  while( !accepted )
  {
    // Set model data and domain and time
    taylor->setSCoeff(d, num_taylor_coeff, coeff, tlast, ttrial);
    toHermite(taylor, hermite, tvf, coeff, tlast, ttrial, htrial);
    hermite->evalP(tmodel_fcn);
    hermite->evalD();

    // Rigorous defect error model
    err = getSupNormN(d, hermite->d, sup_norm_wk, 0);
    if(err <= tol)
    {
      // predict accepted stepsize
      accepted = true;
      if( ttrial > tend ) ttrial = tend;
      // Local ODE output for times in [t_{i-1}, t_i]
      // Get new skelatal point (tlast, xlast): Advance step!
    }
    else
    {
      // Compute rejected stepsize and repeat step.
    }
  }
}
```

Figure 4.5: Defect-controlled algorithm

points or at a fixed number of points per integration step. Defect control is rigorous, and the defect remains less than TOL at each time in the integration step. No additional error check is necessary.

4.6.4 The class structure of ODETS

ODETS is implemented as a collection of C++ classes. We describe the functionality of the classes and the relationship between them.

In Figure 4.7, we draw UML class diagrams illustrating the class dependency structure in ODETS. The arrow points to the class that instantiates it. We achieved a clean expressive interface.

Class `ApproximateSolution`: An instance of class `ApproximateSolution` has a solution s, its first derivative p, and its defect d at the solution. This class manages these objects, and a knowledgeable programmer is responsible for their actual data. Each component of each of these objects is an instance of class `Tmodel`, and we can access them in that way.

`ODETS`: ODETS is a collection of C++ methods and functions implementing rigorous defect control. The `solve` method implements the main integration algorithm. Output method `solout` implements printing continuous output used for data collection in this project. We discuss `solout`, the first step function `getFirstStep`, and the minimal step function `getHmin`.

Previous sections discussed building the Hermite approximate solution (Phase I) and adaptive time-stepping with the elementary controller (Phase III). ODETS implements accepted steps with the function `getAcceptedStep` and rejected steps with the function `getRejectedStep`. The predicted stepsize H with $p = k$ from (4.6)

```
#include "odets.hpp"
#include <math.h>

void solout(double tout,
            ApproximateSolution * hermite,
            ApproximateSolution * taylor,
            double err, double posneg, double tol)
{
  double val(0.0);
  int n = hermite->getN();

  std::cout << "t =" << tout  << ", ";
  for(int i=0;i<n;i++)
  {
    hermite->getS(i, tout, val);
    std::cout << val << ", ";
  }
  for(int i=0;i<n;i++)
  {
    hermite->getP(i, tout, val);
    std::cout << val << ", ";
  }
  for(int i=0;i<n;i++)
  {
    hermite->getD(i, tout, val);
    std::cout << val << ", ";
  }
  for(int i=0;i<n;i++)
  {
    taylor->getS(i, tout, val);
    std::cout << val << ", ";
  }
  for(int i=0;i<n;i++)
  {
    taylor->getP(i, tout, val);
    std::cout << val << ", ";
  }
  for(int i=0;i<n;i++)
  {
    taylor->getD(i, tout, val);
    std::cout << val << ", ";
  }
  std::cout << posneg*err << ", ";
  std::cout << tol << ", ";
}
```

Figure 4.6: Printing via SOLOUT from ODETS

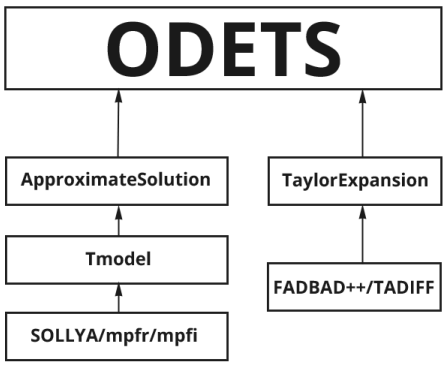


Figure 4.7: Arrow points to the class that instantiates it.

is implemented in ODETS with the function `getPredictedStep`, but it is not called in `solve`.

Class `TaylorExpansion`: An instance of class `TaylorExpansion` has considerable functionality, we will cover the portion relevant to `solve`. This class generates the code list for a type TADIFF function with the class method `setCodeList`, generates all Taylor coefficients with `getAllTaylorCoefficients`, and retrieves a specific Taylor coefficient with `getTaylorCoefficient`. The data array `coeff` instantiated outside class `TaylorExpansion` and class `ApproximateSolution` helps to maintain a clean interface.

Class `Tmodel`: The class `Tmodel` interfaces with SOLLYA. It can be applied to compute a Taylor model in any application, including ODETS, with considerable functionality and flexibility. All rigorous computations pass through class `Tmodel`.

The SOLLYA tool implements Taylor models in one independent variable, and it implements a rigorous supremum norm. We described it's user interface in Chapter 3.

Chapter 5

Numerical results and discussions

Our ODE-IVP solver ODETS fully automates rigorous defect control. We demonstrate through computational experiments that rigorous defect control is possible, and defect control is an effective error control strategy. We demonstrate that ODETS can solve autonomous and non-autonomous ODE-IVP known to give adaptive time-stepping strategies difficulty.

The next section defines our test cases. Additional model problems are discussed and numerically solved in the final section of this chapter.

5.1 Test cases

We are interested in integration tests to distinguish adaptive time-stepping strategies and evaluate performance. We choose the ill-conditioned Lorenz attractor and the well-conditioned predator-prey model problems as our test cases.

Lorenz: We test with the Lorenz attractor;

$$\begin{aligned} x_1' &= x_2\ x_3 - (8x_1)/3, \quad & x_1(0) &= 36, \\ x_2' &= 10\ (x_3 - x_2), & x_2(0) &= 15, \quad t \in [0, 15]. \\ x_3' &= x_2\ (28 - x_1) - x_3, & x_3(0) &= 15, \end{aligned} \tag{5.1}$$

A constant linear transformation of variables applied to the Lorenz attractor in [29] leads to the Lorenz attractor stated in [18, p. 514], where the initial condition is $x_1(0) = 27$, $x_2(0) = -8$, $x_3(0) = 8$, and the integration interval is $[t_0, t_f] = [0, 20]$. We are testing with the vector field defined in [18, p. 514]. We use the integration interval and transformed initial condition defined in [29]. As a result, a plot of our numerical solution is visually comparable to the plot in [18, p. 514], and we can compare the numerical performance between ODETS and the results in [29].

Predator-Prey: We test with the Predator-Prey model problem stated in [29];

$$\begin{aligned} x_1' &= x_1 - 0.1\ x_1\ x_2 + 0.02t, & x_1(0) &= 30, \\ x_2' &= -x_2 + 0.02\ x_1\ x_2 + 0.008t, & x_2(0) &= 20, \end{aligned} \quad t \in [0, 40]. \tag{5.2}$$

The final time t_{end} in the paper is 4, but plots in that article indicate that t_{end} should be 40.

5.2 Adaptive time-stepping study

An adaptive time-stepping strategy is stepsize control, as discussed in Section 2.7. We consider adaptive time-stepping strategies based on two assumptions: Error is proportional to a power of the stepsize, and the constant of proportionality slowly

varies as the process under control moves from step to step. In forward error control, the power is related to the consistency order. There is no counterpart to consistency order in rigorous defect control, but Taylor models enable us to overcome this obstacle.

We seek to rigorously control the supremum norm of the defect at the Hermite approximate solution to be less than a user specified tolerance. As shown with the logistic model in Section 4.3, interval evaluation of the tangent vector field at the approximate solution is known to overestimate the supremum norm of the defect, and we know Taylor models reduce overestimation. We introduce Taylor models of the defect to rigorously and tightly bound the defect. The Taylor polynomial of the defect naturally provides a counterpart in defect control to consistency in forward error control. The degree of the Taylor model of the defect naturally provides a counterpart in defect control to consistency order in forward error control. We assure in our algorithm that the width of the defect Taylor model bound is less than the user specified tolerance TOL and the computed upper bound δ defined in (3.18) is also less than TOL.

The adaptive time-stepping study identifies an acceptable adaptive time-stepping strategy, and it summarizes our reasoning through computer experiments aimed at choosing a stepsize control strategy.

Compare time-stepping algorithm choices: In Table 5.1, we recall the cost per step of the explicit strict defect control continuous Runge–Kutta (SDC CRK) formulas of Enright [29, Table 1, p. 284]: Subsection 2.5.2 introduced the classical p^{th}-order, s-stage, discrete Runge–Kutta formula, and Subsection 2.5.3 introduced $\widetilde{s}$ additional stages to obtain a differentiable solution and continuous defect. We expect SDC CRK methods to take three times the computational time of a classical Runge–Kutta

Formula	p	s	$\widetilde{s}$	$(s+\widetilde{s})/s$
SDC5	5	6	12	3.0
SDC6	6	7	15	3.1
SDC8	8	13	27	3.1

Table 5.1: Cost per step of strict defect control [29, Table 1, p. 284].

method based on the final column in Table 5.1. Enright [29] did not report a timing performance metric, but he did report total number of steps. The number of SDC CRK steps are recorded in Table 5.2 and Table 5.3.

We compare Enright's asymptotic defect control steps with the steps required of respected Runge–Kutta software. We record the accepted and rejected steps reported from the Runge–Kutta software; order 5 DOPRI5 developed by Hairer, Norsett, and Wanner [45], order 6 DVERK developed by Hull, Enright, and Jackson [60], and order 8 DOP853 developed by Hairer, Norsett, and Wanner [45]. The number of Runge–Kutta steps are recorded in Table 5.2 and Table 5.3.

For the purpose of comparison, we compute Taylor series steps with the software DAETS developed by Nedialkov and Pryce [76, 77, 78]. Gladwell, Shampine, and Brankin [39] mentioned Stetter's stepsize controller (4.4), (4.5), and (4.6). Nedialkov independently rediscovered this error control, and he implemented it with $p = k$ in his successful DAE initial value problem software DAETS [74]. We call Stetter's error control *predictive control* because the error predicts a stepsize, based on the elementary controller, that meets tolerance. The number of Taylor series steps are recorded in Table 5.2 and Table 5.3.

We developed six variations of ODETS to study adaptive time-stepping, designed so that the non-rigorous predictive forward error control, the non-rigorous elementary

ORDER	TOL	SDC CRK	DAETS	Runge–Kutta	CPO ODETS
5	10^{-2}	356	172/0	129/33	195/0
	10^{-4}	751	393/0	283/48	426/0
	10^{-6}	1738	818/0	685/32	916/0
	10^{-8}	4304	1780/0	1703/0	1975/0
6	10^{-2}	316	154/0	177/0	169/0
	10^{-4}	642	299/0	364/0	334/0
	10^{-6}	1339	573/0	744/0	638/0
	10^{-8}	2865	1112/0	1573/0	1233/0
8	10^{-2}	145	131/0	**84/36**	142/0
	10^{-4}	228	211/0	**136/44**	234/0
	10^{-6}	371	357/0	**223/76**	394/0
	10^{-8}	634	593/0	**392/118**	656/0

Table 5.2: Lorenz model problem Equation (5.1). A bold table entry indicates that the percentage of rejected steps exceeds 20 percent [36].

ORDER	TOL	SDC CRK	DAETS	Runge–Kutta	CPO ODETS
5	10^{-2}	70	48/0	35/7	54/0
	10^{-4}	148	111/0	**64/17**	123/0
	10^{-6}	315	238/0	154/12	264/0
	10^{-8}	705	513/0	382/6	569/0
6	10^{-2}	65	48/0	48/0	50/0
	10^{-4}	134	87/0	92/0	96/0
	10^{-6}	277	168/0	188/0	186/0
	10^{-8}	585	325/0	392/0	359/0
8	10^{-2}	34	38/0	**24/15**	41/0
	10^{-4}	53	63/0	**37/10**	69/0
	10^{-6}	83	103/0	60/15	114/0
	10^{-8}	127	173/0	**104/29**	190/0

Table 5.3: Predator-Prey model problem Equation (5.2). A bold table entry indicates that the percentage of rejected steps exceeds 20 percent [36].

forward error control, and rigorous defect control can be compared.

The *coefficient predicted observed* ODETS solver (CPO ODETS) implements predicted forward error control. The computed upper bound δ of the defect defined in (3.18) is simply observed. CPO ODETS is not rigorous defect control. The number of steps required from CPO ODETS and DAETS runs are expected to be "identical". The number of CPO ODETS steps are recorded in Table 5.2 and Table 5.3.

The *coefficient predicted validated* ODETS solver (CPV ODETS) implements predicted forward error control for accepted steps and defect control with the elementary controller for rejected steps. One can also compute rejected steps with the elementary controller based on forward error, but this situation did not result in an acceptable stepsize controller. The step is accepted whenever the width of the defect Taylor model bound is less than TOL and the computed upper bound δ defined in (3.18) is also less than TOL. CPV ODETS is rigorous defect control. The number of CPV ODETS steps are recorded in Table 5.4 and Table 5.5.

The *coefficient elementary observed* ODETS solver (CEO ODETS) implements the elementary stepsize controller (accepted and rejected steps) based on traditional forward error control. The computed upper bound δ of the defect defined in (3.18) is simply observed. CEO ODETS is not rigorous defect control. The number of CEO ODETS steps are recorded in Table 5.4 and Table 5.5.

The *coefficient elementary validated* ODETS solver (CEV ODETS) implements the elementary stepsize controller with accepted steps based on traditional error control with additional requirements. The step is accepted whenever the width of the defect Taylor model bound is less than TOL and the computed upper bound δ defined in (3.18) is also less than TOL. Rejected steps are based on defect control. Once again,

ORDER	TOL	CPO	CPV	CEO	CEV	DEF	DEV
5	10^{-2}	195/0	**451/446**	121/30	**432/432**	475/17	529/13
	10^{-4}	426/0	**982/958**	241/20	**909/909**	1033/1	1184/1
	10^{-6}	916/0	**2103/2019**	465/1	**1907/1907**	2228/1	2547/1
	10^{-8}	1975/0	**4522/4319**	897/0	**4033/4033**	4798/1	5480/1
6	10^{-2}	169/0	**364/364**	121/36	**369/369**	367/32	418/29
	10^{-4}	334/0	**692/679**	209/31	**659/659**	718/1	810/1
	10^{-6}	638/0	**1327/1278**	357/11	**1243/1243**	1388/1	1563/1
	10^{-8}	1233/0	**2555/2449**	655/1	**2365/2365**	2679/1	3015/1
8	10^{-2}	142/0	**267/267**	**116/40**	**273/273**	278/55	304/49
	10^{-4}	234/0	**433/432**	176/42	**424/424**	445/24	488/19
	10^{-6}	394/0	**715/705**	273/33	**691/691**	733/1	808/1
	10^{-8}	656/0	**1188/1162**	423/9	**1140/1140**	1223/1	1348/1

Table 5.4: Lorenz model problem Equation (5.1). A bold table entry indicates that the percentage of rejected steps exceeds 20 percent [36].

ORDER	TOL	CPO	CPV	CEO	CEV	DEF	DEV
5	10^{-2}	54/0	112/14	**53/14**	**104/60**	114/14	129/11
	10^{-4}	123/0	241/26	99/10	**215/155**	235/10	270/9
	10^{-6}	264/0	516/48	189/17	**461/408**	501/9	575/5
	10^{-8}	569/0	1113/108	358/13	**990/967**	1077/0	1234/0
6	10^{-2}	50/0	90/9	49/11	**87/46**	94/21	103/14
	10^{-4}	96/0	172/16	83/12	**157/90**	171/18	192/16
	10^{-6}	186/0	333/36	146/15	**300/227**	321/12	364/10
	10^{-8}	359/0	640/64	256/14	**574/482**	617/7	697/4
8	10^{-2}	41/0	66/12	**43/14**	**69/32**	**74/23**	**79/21**
	10^{-4}	69/0	110/17	66/16	**107/59**	115/21	126/17
	10^{-6}	114/0	180/11	101/15	**172/110**	185/23	204/16
	10^{-8}	190/0	301/30	158/20	**283/207**	304/21	335/13

Table 5.5: Predator-Prey model problem Equation (5.2). A bold table entry indicates that the percentage of rejected steps exceeds 20 percent [36].

one can compute rejected steps with the elementary controller based on forward error, but this situation did not result in an acceptable stepsize controller. CEV ODETS is rigorous defect control. The number of CEV ODETS steps are recorded in Table 5.4 and Table 5.5.

The *defect elementary validated* ODETS solver (DEF) predicts accepted and rejected stepsize with the elementary stepsize controller evaluated at the supremum norm of the defect at the Hermite approximate solution. This controller accepts the step whenever the width of the defect Taylor model bound is less than the user specified tolerance TOL and the computed upper bound δ defined in (3.18) is also less than TOL. DEF is rigorous defect control. The number of DEF steps are recorded in Table 5.4 and Table 5.5.

The *deviation elementary validated* ODETS solver (DEV) is designed similar to DEF except it computes the supremum norm of the deviation at the Taylor approximate solution. DEV is rigorous deviation control.

We numerically solve the test cases at TOL $= 10^{-8}$ and order 14. Nedialkov [74] recommended these parameters based on experience with Taylor series methods under forward error control. We did not know the dependence of rigorous defect control performance on order ahead of time. We graphically illustrate the piecewise polynomial Hermite approximate solution, the generated local stepsizes, and the supremum norm of the defect on each step in a side-by-side comparison. We also compare our ODETS solution with the solution generated by the MATLAB method ODE113, a variable-step, variable-order Adams-Bashforth-Moulton PECE method of orders 1 to 13. We chose a relative and absolute tolerance of 10^{-12} for ODE113, and, at this tolerance, this Adams-Bashforth code renders a smoother error plot when comparing the ODE113

numerical solution to the DEF numerical solution than the go-to MATLAB method ODE45 rendered. The full predator-prey model is illustrated in Figure 5.1, and the first steps are illustrated in Figure 5.2. The full Lorenz attractor is illustrated in Figure 5.3, and the first steps are illustrated in Figure 5.4.

Discussion on time-stepping algorithm choices: We compare asymptotic defect control with rigorous defect control. Admittedly asymptotic defect control at orders 5, 6, and 8 have an advantage, because Taylor series methods perform best at orders in the range 15-20 [74]. Asymptotic defect control results were compared to accepted/rejected step counts collected from well known Runge–Kutta methods at orders 5, 6, and 8. We compare the methods on the test cases.

We found that the defect controlled SDC CRK strategy require roughly twice as many steps as the forward error controlled Runge–Kutta, DAETS, or CPO ODETS strategy in both test cases. SDC CRK methods are defect controlled, and the Runge–Kutta methods are forward error controlled.

The DAETS and CPO ODETS strategies roughly required the same number of steps which indicates that our software is implemented as expected. We found that DAETS and CPO ODETS exhibit zero step rejections.

The number of CPO ODETS, CPV ODETS, CEO ODETS, and CEV ODETS steps are recorded in Table 5.4 and Table 5.5. Coefficient predictive control (CPO ODETS and CPV ODETS) does not perform well when validated, nearly each step in CPV ODETS is rejected in Lorenz. Similarly, coefficient elementary control (CEO ODETS and CEV ODETS) does not perform well when validated, over 20 percent of the steps in CEV ODETS are rejected in Lorenz and Predator-Prey. Moreover, the validated software

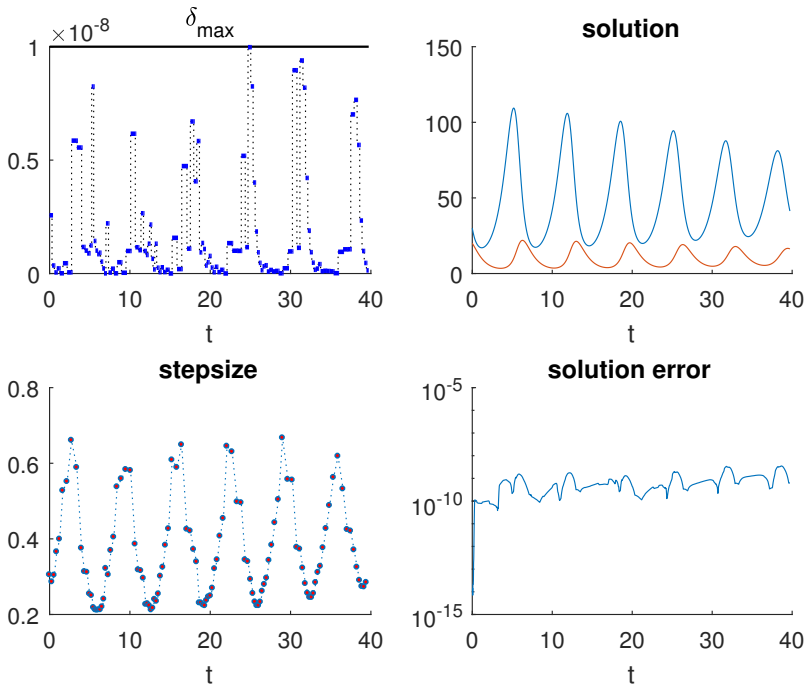


Figure 5.1: Defect controlled predator-prey at order 14 and TOL $= 10^{-8}$

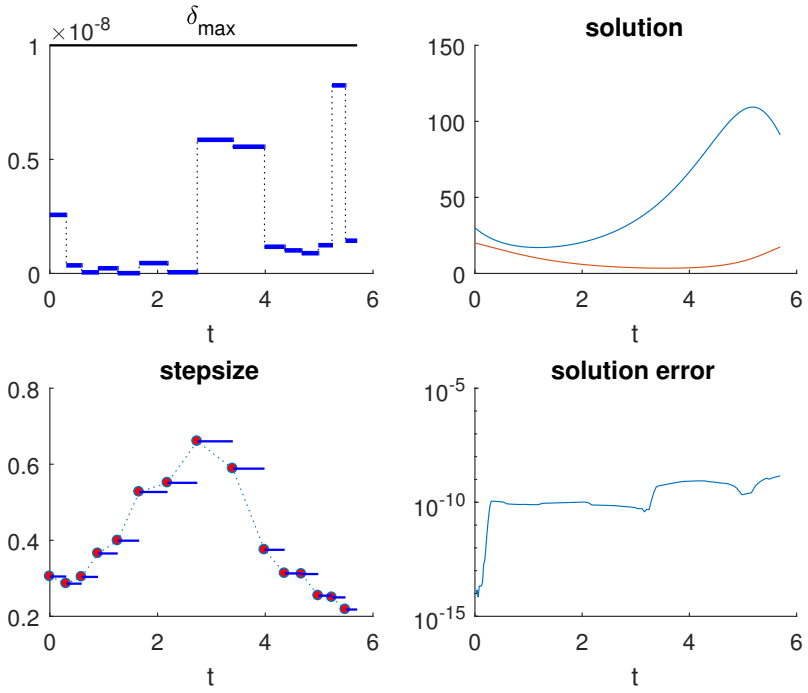


Figure 5.2: Defect controlled predator-prey (zoom in) at order 14 and TOL $= 10^{-8}$

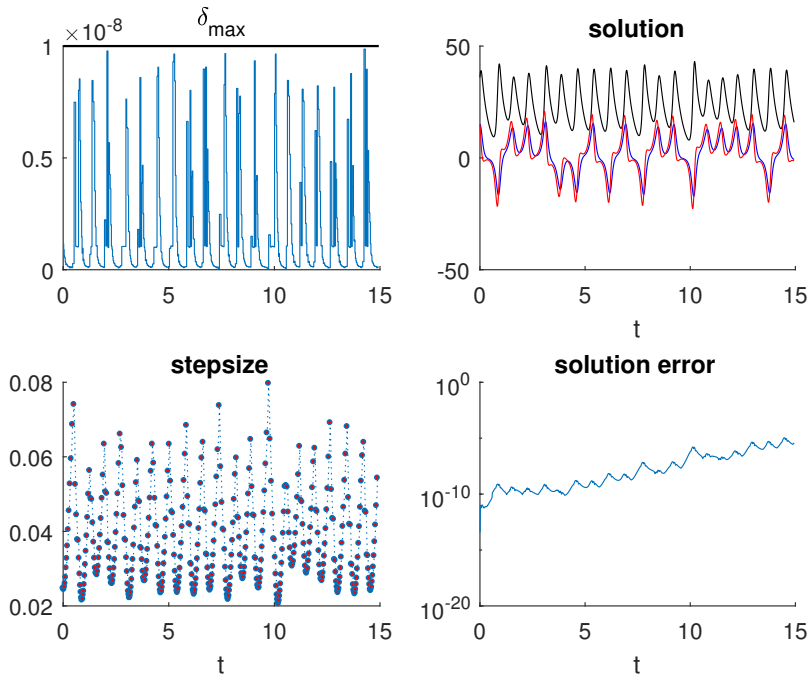


Figure 5.3: Defect controlled Lorenz at order 14 and TOL $= 10^{-8}$

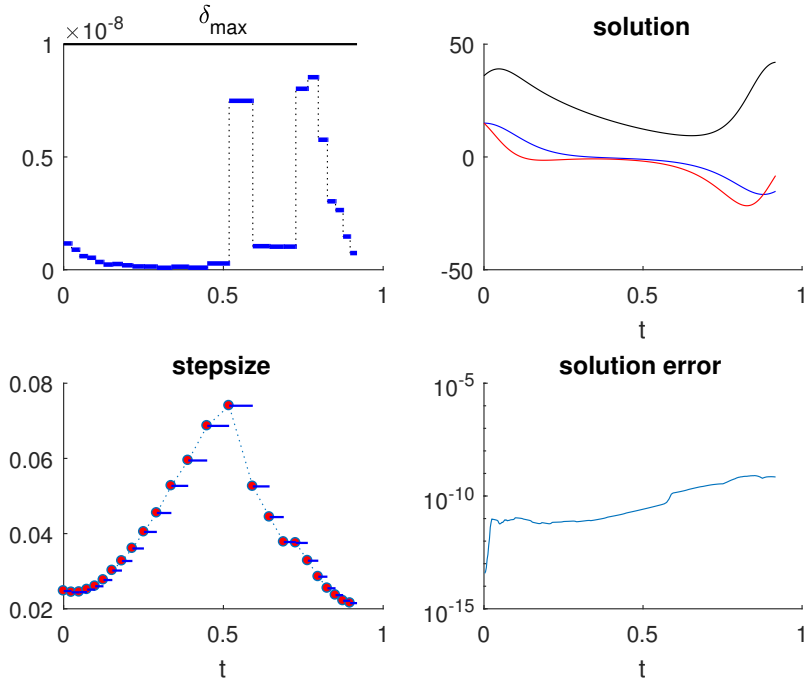


Figure 5.4: Defect controlled Lorenz (zoom in) at order 14 and TOL $= 10^{-8}$

CPV ODETS and CEV ODETS performed prohibitively poorly[1] when rejected steps used the forward error to compute the new stepsize. We conclude that the forward error recommends stepsizes which are too large, and step rejections make a predictive control strategy prohibitively inefficient.

We investigate defect elementary validated and deviation elementary validated control. Observe that the number of defect elementary validated control steps or the number of deviation elementary validated control steps is roughly the number of strict defect control steps SDC CRK reported by Enright. Figure 4.2 supports the observation that the supremum norm of the defect is less than the supremum norm of the deviation on each step. Thus the stepsize recommended by a deviation will be smaller than the stepsize recommended by a defect. This explains the larger number of steps required for deviation elementary validated control over defect elementary validated control.

We find that rigorous defect control is possible as illustrated in Figure 5.1, Figure 5.2, Figure 5.3, and Figure 5.4.

5.3 Performance study

The performance study extends the adaptive time-stepping study reported earlier in this chapter to higher order for defect elementary validated control (DEF).

Performance study accepted and rejected step counts: The performance study counts the number of accepted and rejected steps required to numerically solve the test cases with defect elementary validated ODETS. We collected data from defect

[1] The number steps produced enough output to fill 120 Gb of disk space.

order	tol	Lorenz $t_{\text{end}} = 15$ acc/rej	pred. prey $t_{\text{end}} = 40$ acc/rej
15	10^{-6}	**285/66**	**81/23**
	10^{-8}	371 / 56	105 / 21
	10^{-10}	483 / 22	136 / 15
	10^{-12}	640 / 2	181 / 18
20	10^{-6}	**215/61**	**64/21**
	10^{-8}	**261/68**	**77/23**
	10^{-10}	**319/68**	**94/25**
	10^{-12}	399 / 48	**116/24**

Table 5.6: Lorenz model problem Equation (5.1) and Predator-Prey model problem Equation (5.2). A bold table entry indicates that the percentage of rejected steps exceeds 20 percent [36].

elementary validated control at orders 15 and 20 over a wide range of tolerances. The results of the performance study are recorded in Table 5.6.

Discussion on performance study: Table 5.4 and Table 5.5 record accepted/rejected steps for defect elementary validated control at TOL $= 10^{-6}$ and order 8; Lorenz is 733/1 and Predator-Prey is 185/23. The same tables record accepted/rejected steps for defect elementary validated control at a tighter tolerance TOL $= 10^{-8}$ and the same order 8; Lorenz is 1223/1 and Predator-Prey is 304/21. These performance metrics are well within specification that rejected/total steps is less than 0.2. We say this to point out that the results of this study in Table 5.6.

Table 5.6 illustrates that the accepted steps decrease as order increases. However, the number of rejected steps increases so that rejected/total steps is greater than 0.2 in most entries of the table, which concerns us.

We investigate the cause for rejected steps in the defect width study.

5.4 Defect width study

The ordered sequence of Taylor model plots of increasing degree, Figure 3.1, Figure 3.2, Figure 3.3, and Figure 3.4 illustrate that the width of the Taylor bound may decreases as the degree of the Taylor model increases. The defect width study increases the order. Increasing the order increases the degree of the Taylor model of the defect. The defect width study aims to illustrate that increasing consistency order decreases the width of the Taylor model bound for the defect at the Hermite approximate solution.

The defect width study also illustrates the effect of order on accepted/rejected steps holding tolerance constant. It graphically illustrates the maximal width of the Taylor model bound on every step, and it illustrates the upper bound of the defect used in defect elementary validated control on every step. These two bounds are projected, and it is easy to visualize these projected bounds as a box plot. Each box indicates the median center mark and indicates the 25th/75th percentile bottom/upper edges. The outliers are plotted using the '+' symbol.

Defect width study results: Table 5.7 records the effect of order on accepted and rejected steps holding tolerance constant at $\text{TOL} = 10^{-10}$ for defect elementary validated control as the orders vary $10, \ldots, 20$.

The left box-plots in Figure 5.5 and Figure 5.6 illustrate the infinity norm of the remainder bound for the defect's Taylor model. The right box-plots in these figures illustrate the dependence of the upper bound δ defined in (3.18) for the supremum norm of the defect at the Hermite approximate solution on order.

Discussion of the defect width study: We observe in Table 5.7 for the Lorenz and Predator-Prey test cases that the number of accepted steps decreases as the order

	Lorenz $t_{\text{end}} = 15$		pred. prey $t_{\text{end}} = 40$	
order	acc / rej	runtime (sec)	acc / rej	runtime (sec)
10	1123 / 1	9.3	292 / 18	1.2
11	895 / 1	9.3	239 / 21	1.2
12	740 / 1	8.8	201 / 20	1.1
13	627 / 1	8.3	174 / 17	1.2
14	545 / 7	7.1	153 / 20	1.0
15	483 / 22	6.8	136 / 15	1.0
16	436 / 40	7.4	124 / 21	1.1
17	398 / 51	6.9	114 / 20	1.1
18	366 / 57	7.0	**106/25**	1.1
19	340 / 62	7.1	**100/26**	1.1
20	**319/68**	7.2	**94/25**	1.1

Table 5.7: Lorenz model problem Equation (5.1) and Predator-Prey model problem Equation (5.2). A bold table entry indicates that the percentage of rejected steps exceeds 20 percent [36].

increases, and the number of rejected steps increase as the order increases.

The left box-plots in Figure 5.5 and Figure 5.6 illustrate for Predator-Prey and Lorenz that the infinity norm of the remainder bound for the defect's Taylor model is independent of the order; the median width and the 25/75 percentiles of the defect's Taylor remainder bound remain constant. The right box-plots in these figures illustrate the dependence of the upper bound δ defined in (3.18) for the supremum norm of the defect at the Hermite approximate solution on order; the median width decays exponentially, but the decrease is no longer significant after order 14. On the other hand, we observe a large number of outliers in all the box-plots, and the outliers for the upper bound of the supremum norm of the defect at the Hermite approximate solution may be in the whole tolerance interval.

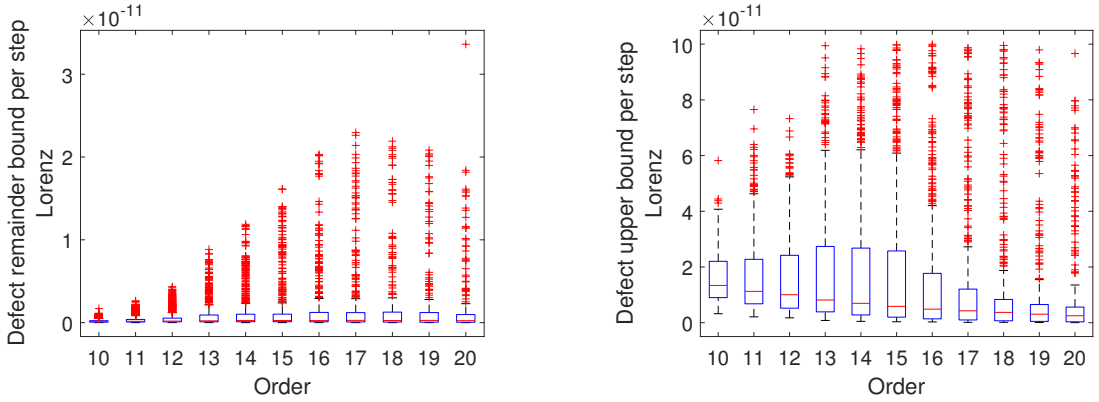


Figure 5.5: (Left) Infinity norm of the defect's Taylor remainder bound at each order. (Right) Upper bound for the supremum norm of the defect for Lorenz at each order.

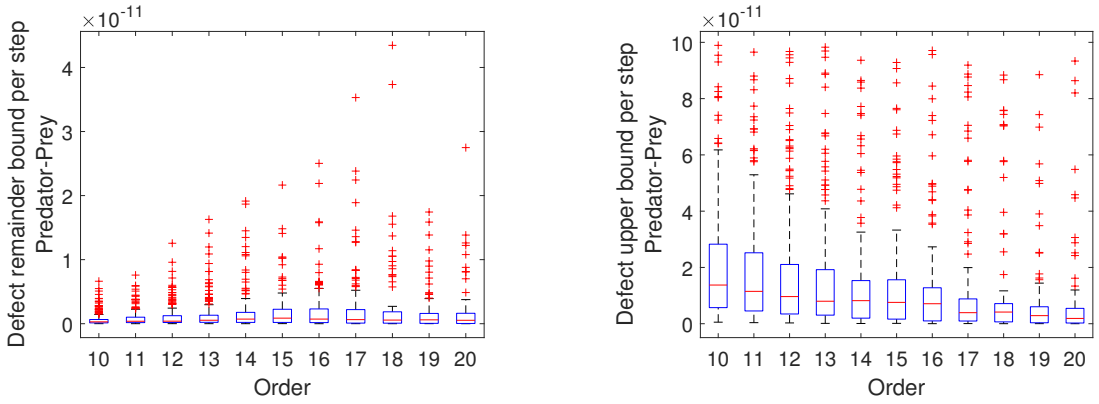


Figure 5.6: (Left) Infinity norm of the defect's Taylor remainder bound at each order. (Right) Upper bound for the supremum norm of the defect for Predator-Prey at each order.

5.5 Initial value problems solved

We demonstrate that defect elementary validated control ODETS can solve autonomous and non-autonomous ODE-IVP. The order is always set to 14, and the tolerance is always TOL $= 10^{-10}$.

5.5.1 Three DETEST examples

We choose three examples from Enright and Pryce [37]: non-stiff C5 (five outer planets and the sun), non-stiff D1 (orbit equations), and stiff D3 (Nonlinear with real eigenvalues).

Five outer planets and the sun: The five outer planets and the sun example is fully defined in [37, non-stiff C5]. Runtime statistics: 2.227 recommended initial stepsize, 2.227 accepted initial stepsize, 403 accepted steps, and 80 rejected steps.

Enright and Pryce orbit: The Enright and Pryce orbit example is fully defined in [37, non-stiff D1]. Runtime statistics: 3.613×10^{-1} recommended initial stepsize, 3.613×10^{-1} accepted initial stepsize, 48 accepted steps, and 3 rejected steps.

Nonlinear example with real eigenvalues: This nonlinear example with real eigenvalues example is fully defined in [37, Stiff D3]. Runtime statistics: 9.232×10^{-6} recommended initial stepsize, 4.397×10^{-6} accepted initial stepsize, 898 accepted steps, and 44 rejected steps.

5.5.2 Restricted two-body model

Ascher and Petzold [2, Problem 4.12] exhibit this classical restricted two-body model from astronomy to motivate stepsize control. It is known to give stepsize controllers difficulty.

Consider two bodies of masses $\mu_1 = 0.012277471$ and $\mu_2 = 1 - \mu_1$ in planar motion, and a third body of negligible mass moving in the same plane. The classical physics

model for this situation is the second order system

$$
\begin{aligned}
& x_1'' = x_1 + 2\ x_2' - \mu_2\ \left(\tfrac{x_1+\mu_1}{d_1(x_1,x_2)}\right) - \mu_1\ \left(\tfrac{x_1-\mu_2}{d_2(x_1,x_2)}\right), \\
& x_2'' = x_2 - 2\ x_1' - \mu_2\ \left(\tfrac{x_2}{d_1(x_1,x_2)}\right) - \mu_1\ \left(\tfrac{x_2}{d_2(x_1,x_2)}\right), \\
& (u,v) \in \mathbb{R}^2 \mapsto d_1(u,v) \stackrel{\text{def}}{=} \left(\left(u+\mu_1\right)^2 + \left(v\right)^2\right)^{3/2}, \\
& (u,v) \in \mathbb{R}^2 \mapsto d_2(u,v) \stackrel{\text{def}}{=} \left(\left(u-\mu_2\right)^2 + \left(v\right)^2\right)^{3/2}, \\
& x_1(0) = 0.994, \quad x_1'(0) = 0, \\
& x_2(0) = 0, \quad x_2'(0) = -2.00158510637908252240537862224, \\
& t \in [0, 17.1].
\end{aligned}
\tag{5.3}
$$

The mapping d_1 is mathematically zero at $(-\mu_1, 0)$, and d_2 is mathematically zero at $(\mu_2, 0)$ which can result in numerical difficulties as the trajectory passes near these singular points.

Runtime statistics: 7.420×10^{-4} recommended initial stepsize, 4.538×10^{-4} accepted initial stepsize, 331 accepted steps, and 23 rejected steps.

5.5.3 Forced Brusselator

Chapter 4 of Bendtsen and Stauning [6] discuss the application of TADIFF to ODE initial value problems, and their example is the forced Brusselator. The framework to construct a Taylor approximate solution is presented, and it would be a minor achievement to implement an ODE solver with constant stepsize. However adaptive and rigorous stepsize control which is the topic of this thesis is not discussed.

Define the mapping

$$
(u,v) \in \mathbb{R}^2 \mapsto f(u,v) = u\ \left(\frac{6}{5} - u\ v\right) \in \mathbb{R}.
$$

The forced Brusselator is the ODE-IVP

$$\begin{aligned} x_1' &= \tfrac{2}{5} - x_1 - f(x_1, x_2) + 0.03\ \cos\big(\tfrac{\pi}{4}t\big), \quad x_1(0) = 0.38, \\ x_2' &= f(x_1, x_2), \qquad\qquad\qquad\qquad\quad x_2(0) = 3.3, \\ t &\in [0, 100]. \end{aligned} \tag{5.4}$$

Runtime statistics: 9.648×10^{-1} recommended initial stepsize, 9.648×10^{-1} accepted initial stepsize, 136 accepted steps, and 17 rejected steps.

5.5.4 Discussion

For each example, we set ODETS order parameter to 14 and tolerance TOL $= 10^{-10}$. The three DETEST examples and the restricted two-body model are autonomous. The Forced Brusselator is non-autonomous.

We applied our rigorous defect control solver – defect elementary validated control ODETS – to the five examples in this section. In each example, ODETS computed a continuously differentiable, global numerical approximate solution u which satisfies exactly the perturbed reference problem

$$u'\big(t\big) = f\big(t, u\big(t\big)\big) + \Delta u\big(t\big), \quad u\big(t_0\big) = x_0 + \Delta u\big(t_0\big), \quad \|\Delta u\|_{[t_0, t_{\text{end}}],\infty} \leq \text{TOL}.$$

We confirmed the statement $\|\Delta u\|_{[t_0, t_{\text{end}}],\infty} \leq$ TOL rigorously held, the continuous differentiability of u, and the continuity of the defect in our output.

Stetter's initial stepsize recommendation (4.6) was accepted in 3 of the 5 problems. It failed to produce an initial stepsize satisfying $\|\Delta u\big(t_0\big)\|_\infty \leq$ TOL for the stiff D3 and restricted two-body model examples, but the first follow up stepsize from the

elementary controller succeeded. While the more complicated initial stepsize selector of Gladwell, Shampine, and Brankin [39] is available, it was not required for our test examples.

Chapter 6

Conclusions and future work

In conclusion, this thesis automated rigorous defect control for explicit, first-order, nonlinear ODE-IVP with a bounded, Lipschitz continuous tangent vector field. Indeed, Lipschitz continuity holds automatically for the Taylor series method, because it follows from the mean-value theorem, whenever the vector field is continuously differentiable. The vector field must be at least continuously differentiable for the Taylor series method to be applicable. Rigorous defect control is a local three phase process. By assumption, the local solution can be continued to the interval specified by the reference problem.

6.1 Conclusions

We described a complete software implementation of the Corless and Corliss algorithm based on the Taylor series method and extensively tested our software. This thesis project, proposed as an unsolved problem in the PhD thesis [73], concerned the implementation of this algorithm. Our choice to represent the defect as a Taylor

model enabled us to repurpose basic time-stepping software to defect control.

This thesis connected defect control to residual-based backward error analysis and distinguished residual-based backward error from backward error. Residual-based backward error analysis is a remarkably straightforward procedure and a powerful point of view. In residual-based backward error analysis, the defining equation is altered by the known residual only, and the backward error is a residual. Existence and uniqueness of solution to ODE-IVP enables us to conclude that the backward error is the defect. This background material from Chapter 2 is the theoretical foundation of this thesis.

Defect control controls the quality of the initial value problem itself. In geometric terms, defect control controls the tangency of the approximate solution to the tangent vector field at the approximate solution. The norm of the defect is controlled to be within a user specified tolerance. The user must validate that the perturbed reference problem is a nearby problem.

Chapter 2 discussed two approaches to defect control, asymptotic defect control in Section 2.5 and guaranteed defect control in Section 2.6. The discussion provides sufficient detail on asymptotic defect control for the reader to understand two justifications for the method which lead Enright to two implementations of his method. Both implementations by Enright are not rigorous in our sense. The mathematical foundation for Enright's approach tightly integrates the approach with the particulars of the underlying engineered problem, in this case the Runge–Kutta method. The asymptotic nature derives from approximation theory and requires the Lipschitz condition from the existence and uniqueness theorem.

We summarize guaranteed defect control. Given a user supplied tolerance TOL,

an ODE-IVP engineered problem computes a local approximate solution of the local reference problem, and a global numerical approximate solution u is constructed which satisfies exactly the perturbed reference problem

$$u'(t) = f(t, u(t)) + \Delta u(t), \quad u(t_0) = x_0 + \Delta u(t_0), \quad \|\Delta u\|_{[t_0,t_{\text{end}}],\infty} \leq \text{TOL}.$$

We know to mathematical certainty that u is the exact solution to the perturbed reference problem. This implies that the defect subsumes all errors. In rigorous defect control, we bound the defect so that the mathematical statement

$$\|\Delta u\|_{[t_0,t_{\text{end}}],\infty} \leq \text{TOL}$$

is not an approximation, it is true to mathematical certainty.

The tolerance parameter controls the magnitude of permissible defects in the model. We control the infinity norm of the defect by controlling the stepsize. Rigorous defect control is automated backward error analysis and a new stepsize control for ODE-IVP. Rigorous defect control computes an outer enclosure of the defect, and it cannot be fooled.

The SOLLYA authors developed their package for validated computing applications. Our software brings the SOLLYA package to software programs through a C++ API. The SOLLYA authors intended SOLLYA to be accessible from their command line interface.

SOLLYA applies interval arithmetic and Taylor models to compute a tight rigorous bound on the supremum norm of the defect. In particular, SOLLYA does interval arithmetic and computes Taylor models. Taylor models complicate the software

solution, but they are known to compute tighter bounds than interval evaluation without them. We tested interval evaluation with Taylor models and the supremum norm method in SOLLYA for bounding the supremum norm of the defect. Both approaches require a Taylor model representation of the defect. The SOLLYA supremum norm reduces overestimation. As a result, larger stepsizes are possible.

This thesis project developed an API through the C++ class `Tmodel` to conveniently access SOLLYA functionality. Write a C++ generic template for any $k+1$ times differential elementary function $t \in \mathbb{R} \mapsto f\big(t\big) \in \mathbb{R}$. Class `Tmodel` will output a degree k Taylor model representation of f. We hope this class will be useful to the computational community.

We introduced several stepsize controllers in Section 2.7. These controllers are designed for an asymptotic error model, or they are designed to estimate the radius of convergence of a series. Not only do Taylor models enable tight bounds in rigorous defect control, they enable us to apply existing time-stepping software. The degree of the Taylor model replaces the order parameter in basic time-stepping software.

The complexity of order-k accurate solutions computed with Taylor series using automatic differentiation is $O\big(k^2\big)$. Thus the cost of obtaining the Taylor series at one step is ck^2 [19]. Similar reasoning implies the cost of obtaining the Taylor models on each step is ck^2. The constant c depends on the number and type of operations in the vector field. Rigorous defect control is roughly twice the cost of the Taylor series method itself.

We demonstrate through computational experiments that rigorous defect control is possible, and defect control is an effective error control strategy. We demonstrate that ODETS can solve autonomous and non-autonomous ODE-IVP known to give adaptive

time-stepping strategies difficulty. While we hope the time-stepping software will be improved in the future, the elementary controller performs satisfactorily.

6.2 Future work

We select future research directions related to defect control in this section. Topics are motivated by open problems discovered in the literature during the preparation of this thesis, by challenges faced in computational experiments, and by comments from members of the thesis committee for this thesis.

Improve time-stepping performance: In the right box-plots of Figure 5.5 and Figure 5.6, we observe a large number of outliers from the 25/75 percentiles in defect upper bound per step data, and the defect median upper bound per step is roughly TOL/10. Loosely speaking, we ideally want the median upper bound per step to be near TOL, the 25/75 percentiles close to the median upper bound, and the outliers to cluster close to the median upper bound per step.

Our ODETS software implements the elementary stepsize controller [91] to predict and control the defect upper bound $\delta_{\max}$ to be less than TOL. A beneficial contribution to ODETS rigorous defect control will develop a time-stepping algorithm which meets the ideal in the previous paragraph and reduces the number of rejected steps.

Software module: Rigorous defect control would benefit from a software module that easily inserts into applications. Given an approximate solution on a local step, this software module evaluates outer and inner enclosures for the supremum norm of the defect at the approximate solution.

An immediate application is to validate the error estimates used for asymptotic defect control in existing continuous output methods. This software module will help in the validation phase of developing new continuous output methods as described in Section 2.5. A validation tool is also useful in grid selection for ODE boundary value problems [35, 88].

Maintaining invariants: We considered three orbit systems described by ODE-IVP in this thesis: The five outer planets and the sun example, the Enright and Pryce orbit example, and the Ascher-Petzold orbit example. These systems and other such Hamiltonian systems conserve physical quantities through a system of equalities or inequalities called invariants. For example, drift off the Poincare invariant is a concern during the numerical integration of the solar system. Typical engineered problems will not conserve invariants, and this can lead to solutions which are not even qualitatively correct [86].

If one is interested in qualitative behaviour of a dynamical system, as opposed to precise prediction in the future, is it more important to control the defect or to ensure the invariants of the true system are preserved, as in symplectic integration algorithms for Hamiltonian systems? For example, if one is integrating the solar system, then drift of the Poincare invariants is a concern. Would preserving invariants help from the point of view of controlling the global defect rigorously?

Software improvements: Let's point out a few software improvements for our implementation of ODETS. We mention that computing the defect (Phase II) in ODETS would support massive parallelization.

ODETS currently analyzes the absolute defect. Some examples can be automatically scaled and, in these cases, using the relative defect (2.14) instead of the absolute defect will help. ODETS would benefit from a user option for absolute/relative defect.

SOLLYA requires the user to call a bookkeeping method for initialization. This should be hidden from users. ODETS can be improved by hiding the initialization and destruction method calls required for SOLLYA.

MATLAB implements event location for its ODE suite. ODETS should accomodate event location. Rootfinding in event location requires ODETS to support a sequence of t_{end}'s that are not monitone. If the t_{end} sequence is in the current integration step, then the Hermite approximate solution can be queried to find the event. Otherwise, the integration must change direction. Under the current implementation of ODETS, the Taylor series at t_{n-1} is re-used, and a stepsize is computed to t_n in the reverse direction. Steps are taken in this manner until the event is located.

MATLAB implements output by storing the approximating polynomial solution. The Hermite or Taylor approximate solution can be used to implement continuous output, and ODETS should replace `solout` with functionality to support this feature.

Bibliography

[1] Adsic, G.: Specification by example: How successful teams deliver the right software. Manning Publications Co (2011)

[2] Ascher, U.M., Petzold, L.R.: Computer methods for ordinary differential equations and differential-algebraic equations, vol. 61. SIAM (1998)

[3] Bauer, F.L.: Computational graphs and rounding error. SIAM Journal on Numerical Analysis **11**(1), 87–96 (1974)

[4] Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: A survey. arXiv preprint arXiv:1502.05767 (2015)

[5] Bendsten, C., Stauning, O.: FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods. Tech. Rep. 1996-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark (1996)

[6] Bendsten, C., Stauning, O.: TADIFF, a flexible C++ package for automatic differentiation using Taylor series. Tech. Rep. 1997-x5-94, Department of Mathematical Modelling, Technical University of Denmark, DK-2800, Lyngby, Denmark

(1997)

[7] Bergsma, M.C.W.: Application of Taylor Series integration to reentry problems. MSc, Delft University of Technology, Delft, Netherlands (2015)

[8] Berz, M., Makino, K.: Verified integration of ODEs and flows using differential-algebraic methods on high-order Taylor models. Reliable Computing **4**(4), 361–369 (1998)

[9] Berz, M., Makino, K.: Suppression of the wrapping effect by Taylor model-based verified integrators: Long-term stabilization by shrink wrapping. International Journal of Differential Equations and Applications pp. 1–29 (2006)

[10] Chang, Y., Corliss, G.: Ratio-like and recurrence relation tests for convergence of series. IMA Journal of Applied Mathematics **25**(4), 349–359 (1980)

[11] Chang, Y., Corliss, G.: ATOMFT: Solving ODEs and DAEs using Taylor series. Computers & Mathematics with Applications **28**(10), 209–233 (1994)

[12] Chevillard, S., Joldes, M., Lauter, C.: Sollya: An environment for the development of numerical codes. In: International Congress on Mathematical Software, pp. 28–31. Springer (2010)

[13] Chevillard, S., Lauter, C., Joldes, M.: Users manual for the Sollya tool. http://www.imm.dtu.dk/fadbad.html (2008).
Accessed: 2016-08-01

[14] Christianson, B.: Reverse accumulation and accurate rounding error estimates for Taylor series coefficient. Optimization Methods and Software **1**(1), 81–94 (1992)

[15] Corless, R.M.: Defect-controlled numerical methods and shadowing for chaotic differential equations. Physica D: Nonlinear Phenomena **60**(1), 323–334 (1992)

[16] Corless, R.M.: What good are numerical simulations of chaotic dynamical systems? Computers & Mathematics with Applications **28**(10-12), 107–121 (1994)

[17] Corless, R.M., Corliss, G.F.: Rationale for guaranteed ODE defect control. In: L. Atanassova, J. Herzberger (eds.) Computer Arithmetic and Enclosure Methods, pp. 3–12. North-Holland, Amsterdam (1992)

[18] Corless, R.M., Fillion, N.: A graduate introduction to numerical methods from the viewpoint of backward error analysis. Springer-Verlag, New York (2013)

[19] Corless, R.M., Ilie, S.: Polynomial cost for solving IVP for high-index DAE. BIT Numerical Mathematics **48**(1), 29–49 (2008)

[20] Corliss, G.F.: Guaranteed error bounds for ordinary differential equations. Theory and Numerics of Ordinary and Partial Differential Equations. Clarendon Press, Oxford (1995)

[21] Corliss, G.F.: Where is validated ODE solving going? Mathematical Research **89**, 48–57 (1996)

[22] Deuflhard, P.: A stepsize control for continuation methods and its special application to multiple shooting techniques. Numerische Mathematik **33**(2), 115–146 (1979)

[23] Enright, W.H.: Analysis of error control strategies for continuous Runge-Kutta methods. SIAM Journal on Numerical Analysis **26**(3), 588–599 (1989)

[24] Enright, W.H.: A new error-control for initial value solvers. Applied Mathematics and Computation **31**, 288–301 (1989)

[25] Enright, W.H.: The relative efficiency of alternative defect control schemes for high-order continuous Runge-Kutta formulas. SIAM Journal on Numerical Analysis **30**(5), 1419–1445 (1993)

[26] Enright, W.H.: Continuous numerical methods for ODEs with defect control. Journal of Computational and Applied Mathematics **125**(1), 159–170 (2000)

[27] Enright, W.H.: Software for ordinary and delay differential equations: Accurate discrete approximate solutions are not enough. Applied Numerical Mathematics **56**(3), 459–471 (2006)

[28] Enright, W.H.: Verifying approximate solutions to differential equations. Journal of Computational and Applied Mathematics **185**(2), 203–211 (2006)

[29] Enright, W.H.: Reducing the uncertainty when approximating the solution of ODEs. In: Uncertainty Quantification in Scientific Computing, pp. 280–293. Springer (2012)

[30] Enright, W.H., Hayashi, H.: A delay differential equation solver based on a continuous Runge–Kutta method with defect control. Numerical Algorithms **16**(3-4), 349–364 (1997)

[31] Enright, W.H., Hayashi, H.: The evaluation of numerical software for delay differential equations. In: Quality of Numerical Software, pp. 179–193. Springer (1997)

[32] Enright, W.H., Hayes, W.B.: Robust and reliable defect control for Runge-Kutta methods. ACM Transactions on Mathematical Software (TOMS) **33**(1), 1 (2007)

[33] Enright, W.H., Higham, D.J.: Parallel defect control. BIT Numerical Mathematics **31**(4), 647–663 (1991)

[34] Enright, W.H., Jackson, K., Norsett, S.P., Thomsen, P.G.: Interpolants for Runge-Kutta formulas. ACM Transactions on Mathematical Software (TOMS) **12**(3), 193–218 (1986)

[35] Enright, W.H., Muir, P.: Runge–Kutta software with defect control for boundary value ODEs. SIAM Journal on Scientific Computing **17**(2), 479–497 (1996)

[36] Enright, W.H., Nedialkov, N.S.: Private communication (2017)

[37] Enright, W.H., Pryce, J.D.: Two FORTRAN packages for assessing initial value methods. ACM Transactions on Mathematical Software **13**(1), 1–27 (1987)

[38] Enright, W.H., Yan, L.: The reliability/cost trade-off for a class of ODE solvers. Numerical Algorithms **53**(2-3), 239–260 (2010)

[39] Gladwell, I., Shampine, L., Brankin, R.: Automatic selection of the initial step size for an ODE solver. Journal of Computational and Applied Mathematics **18**(2), 175–192 (1987)

[40] Gladwell, I., Shampine, L.F., Baca, L., Brankin, R.: Practical aspects of interpolation in Runge-Kutta codes. SIAM Journal on Scientific and Statistical Computing **8**(3), 322–341 (1987)

[41] Google: Google Test.
https://github.com/google/googletest.
Accessed: 2015-11-01

[42] Gustafsson, K.: Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods. ACM Transactions on Mathematical Software (TOMS) **17**(4), 533–554 (1991)

[43] Gustafsson, K.: Control-theoretic techniques for stepsize selection in implicit Runge-Kutta methods. ACM Transactions on Mathematical Software (TOMS) **20**(4), 496–517 (1994)

[44] Gustafsson, K., Lundh, M., Söderlind, G.: A PI stepsize control for the numerical solution of ordinary differential equations. BIT Numerical Mathematics **28**(2), 270–287 (1988)

[45] Hairer, E., Norsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I. Nonstiff Problems, *Springer Series in Computational Mathematics*, vol. 8. Springer-Verlag, New York (1987)

[46] Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems, *Springer Series in Computational Mathematics*, vol. 14. Springer-Verlag, Berlin (1991)

[47] Hall, G.: Equilibrium states of Runge Kutta schemes. ACM Transactions on Mathematical Software (TOMS) **11**(3), 289–301 (1985)

[48] Hall, G.: Equilibrium states of Runge-Kutta schemes: Part II. ACM Transactions on Mathematical Software (TOMS) **12**(3), 183–192 (1986)

[49] Hall, G., Higham, D.J.: Analysis of stepsize selection schemes for Runge-Kutta codes. IMA Journal of Numerical Analysis **8**(3), 305–310 (1988)

[50] Hanson, P., Enright, W.H.: Controlling the defect in existing variable-order Adams codes for initial-value problems. ACM Transactions on Mathematical Software (TOMS) **9**(1), 71–97 (1983)

[51] Hayashi, H.: Numerical solution of retarded and neutral delay differential equations using continuous Runge-Kutta methods. PhD, University of Toronto, Toronto, Ontario (1996)

[52] Hayes, W.B., Jackson, K.R.: A fast shadowing algorithm for high-dimensional ODE systems. SIAM Journal on Scientific Computing **29**(4), 1738–1758 (2007)

[53] Higham, D.J.: Defect estimation in Adams PECE codes. SIAM Journal on Scientific and Statistical Computing **10**(5), 964–976 (1989)

[54] Higham, D.J.: Robust defect control with Runge-Kutta schemes. SIAM Journal on Numerical Analysis **26**(5), 1175–1183 (1989)

[55] Higham, D.J.: Runge–Kutta defect control using Hermite–Birkhoff interpolation. SIAM Journal on Scientific and Statistical Computing **12**(5), 991–999 (1991)

[56] Higham, D.J., Hall, G.: Embedded Runge-Kutta formulae with stable equilibrium states. Journal of Computational and Applied Mathematics **29**(1), 25–33 (1990)

[57] Higham, N.J.: Accuracy and stability of numerical algorithms, 2nd edn. Society for Industrial and Applied Mathematics (2002)

[58] Hoefkens, J.: Rigorous numerical analysis with high-order Taylor models. Ph.D. thesis, Department of Mathematics and Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824 (2001)

[59] Hull, T.E.: The effectiveness of numerical methods for ordinary differential equations. In: J.N. Ortega, W.C. Rheinboldt (eds.) Studies in Numerical Analysis 2, pp. 114–121. Society for Industrial and Applied Mathematics (1970)

[60] Hull, T.E., Enright, W., Jackson, K.: User's guide for DVERK: A subroutine for solving non-stiff ODE's. University of Toronto, Department of Computer Science (1976)

[61] Joldes, M.M.: Rigorous polynomial approximations and applications. Ph.D. thesis, Ecole normale supérieure de lyon-ENS LYON (2011)

[62] Kearfott, R.B., Nakao, M.T., Neumaier, A., Rump, S.M., Shary, S.P., Van Hentenryck, P.: Standardized notation in interval analysis. In: Proc. XIII Baikal International School-seminar Optimization Methods and their Applications, vol. 4, pp. 106–113 (2005).
Accessed: 2016-08-01

[63] Kierzenka, J., Shampine, L.F.: A BVP solver based on residual control and the Matlab PSE. ACM Transactions on Mathematical Software **27**(3), 299–316 (2001)

[64] Langr, J.: Modern C++ programming with test-driven development. The Pragmatic Programmers, LLC (2013)

[65] Langr, J.: Private communication via direct message to @jlangr on twitter (2017)

[66] MacDonald, C.: A new approach for DAEs. PhD, University of Toronto, Toronto, Ontario (1999)

[67] Makino, K.: Rigorous analysis of nonlinear motion in particle accelerators. Ph.D. thesis, Michigan State University, East Lansing, MI, USA (1998)

[68] Makino, K., Berz, M.: Taylor models and other validated functional inclusion methods. International Journal of Pure and Applied Mathematics **4**(4), 379–456 (2003)

[69] Makino, K., Berz, M.: Suppression of the wrapping effect by Taylor model-based validated integrators. Tech. Rep. MSU HEP 40910, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA (2004)

[70] Makino, K., Berz, M.: Suppression of the wrapping effect by Taylor model-based verified integrators: Long-term stabilization by preconditioning. International Journal of Differential Equations and Applications **10**(4) (2005)

[71] Makino, K., Berz, M.: Suppression of the wrapping effect by Taylor model-based verified integrators: The single step. International Journal of Pure and Applied Mathematics **36**(2), 175–196 (2007)

[72] Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to interval analysis. SIAM (2009)

[73] Nedialkov, N.S.: Computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation. Ph.D. thesis, Department of Computer Science, University of Toronto, Toronto, Canada (1999)

[74] Nedialkov, N.S.: Private communication (2017)

[75] Nedialkov, N.S., Kreinovich, V., Starks, S.A.: Interval arithmetic, affine arithmetic, Taylor series methods: Why, what next? Numerical Algorithms **37**(1-4), 325–336 (2004)

[76] Nedialkov, N.S., Pryce, J.D.: Solving differential-algebraic equations by Taylor series (I): Computing Taylor coefficients. BIT Numerical Mathematics **45**(3), 561–591 (2005)

[77] Nedialkov, N.S., Pryce, J.D.: Solving differential-algebraic equations by Taylor series (II): Computing the System Jacobian. BIT Numerical Mathematics **47**(1), 121–135 (2007)

[78] Nedialkov, N.S., Pryce, J.D.: Solving differential algebraic equations by Taylor series (III): The DAETS code. Journal of Numerical Analysis, Industrial and Applied Mathematics **3**(1–2), 61–80 (2008)

[79] Neher, M., Jackson, K.R., Nedialkov, N.S.: On Taylor model based integration of ODEs. SIAM Journal on Numerical Analysis **45**(1), 236–262 (2007)

[80] Neumaier, A.: Taylor forms use and limits. Reliable Computing **9**(1), 43–79 (2003)

[81] Nguyen, H.: Interpolation and error control schemes for algebraic differential equations using continuous implicit Runge-Kutta methods. PhD, University of Toronto, Toronto, Ontario (1995)

[82] Rall, L.B.: Mean value and Taylor forms in interval analysis. SIAM Journal on Mathematical Analysis **14**(2), 223–238 (1983)

[83] Rump, S.M.: INTLAB - INTerval LABoratory. In: T. Csendes (ed.) Developments in Reliable Computing, pp. 77–104. Kluwer Academic Publishers, Dordrecht (1999)

[84] Shakourifar, M., Enright, W.H.: Reliable approximate solution of systems of Volterra integro-differential equations with time-dependent delays. SIAM Journal on Scientific Computing **33**(3), 1134–1158 (2011)

[85] Shampine, L.F.: Interpolation for Runge-Kutta methods. SIAM Journal on Numerical Analysis **22**(5), 1014–1027 (1985)

[86] Shampine, L.F.: Conservation laws and the numerical solution of ODEs. Computers & Mathematics with Applications **12**(5-6), 1287–1296 (1986)

[87] Shampine, L.F.: Solving ODEs and DDEs with residual control. Applied Numerical Mathematics **52**(1), 113–127 (2005)

[88] Shampine, L.F., Muir, P.H.: Estimating conditioning of BVPs for ODEs. Mathematical and Computer Modelling **40**(11-12), 1309–1321 (2004)

[89] Söderlind, G.: Automatic control and adaptive time-stepping. Numerical Algorithms **31**(1-4), 281–310 (2002)

[90] Söderlind, G.: Digital filters in adaptive time-stepping. ACM Transactions on Mathematical Software (TOMS) **29**(1), 1–26 (2003)

[91] Söderlind, G., Wang, L.: Adaptive time-stepping and computational stability. Journal of Computational and Applied Mathematics **185**(2), 225–243 (2006)

[92] Stauning, O., Bendtsen, C.: FADBAD++.
http://www.imm.dtu.dk/fadbad.html (2003).
Accessed: 2016-08-01

[93] Stetter, H.J.: Considerations concerning a theory for ODE-solvers. In: R. Bulirsch, R. Grigorieff, J. Schroder (eds.) Numerical treatment of differential equations, *Lecture Notes in Mathematics*, vol. 631, pp. 188–200. Springer (1978)

[94] Stetter, H.J.: Interpolation and error estimation in Adams PC-codes. SIAM Journal on Numerical Analysis **16**(2), 311–323 (1979)

[95] de Swart, J.J., Söderlind, G.: On the construction of error estimators for implicit Runge-Kutta methods. Journal of Computational and Applied Mathematics **86**(2), 347–358 (1997)

[96] Watts, H.A.: Starting step size for an ODE solver. Journal of Computational and Applied Mathematics **9**(2), 177–191 (1983)

[97] Zivari-Piran, H., Enright, W.H.: An efficient unified approach for the numerical solution of delay differential equations. Numerical Algorithms **53**(2), 397–417 (2010)