12-15-2017 2:00 PM

Feasible Computation in Symbolic and Numeric Integration

Robert H.C. Moir, *The University of Western Ontario*

Supervisor: Corless, Robert M., *The University of Western Ontario*
Joint Supervisor: Moreno Maza, Marc, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Applied Mathematics

Abstract

Two central concerns in scientific computing are the reliability and efficiency of algorithms. We introduce the term *feasible computation* to describe algorithms that are reliable and efficient given the contextual constraints imposed in practice. The main focus of this dissertation then, is to bring greater clarity to the forms of error introduced in computation and modeling, and in the limited context of symbolic and numeric integration, to contribute to integration algorithms that better account for error while providing results efficiently.

Chapter 2 considers the problem of spurious discontinuities in the symbolic integration problem, proposing a new method to restore continuity based on a pair of unwinding numbers. Computable conditions for the unwinding numbers are specified, allowing the computation of a variety of continuous integrals. Chapter 3 introduces two structure-preserving algorithms for the symbolic-numeric integration of rational functions on exact input. A structured backward and forward error analysis for the algorithms shows that they are *a posteriori* backward and forward stable, with both forms of error exhibiting tolerance proportionality.

Chapter 4 identifies the basic logical structure of feasible inference by presenting a logical model of stable approximate inference, illustrated by examples of modeling and numerical integration. In terms of this model it is seen that a necessary condition for the feasibility of methods of abstraction in modeling and complexity reduction in computational mathematics is the preservation of inferential structure, in a sense that is made precise. Chapter 5 identifies a robust pattern in mathematical sciences of transforming problems to make solutions feasible. It is showed that computational complexity reduction methods in computational science involve chains of such transformations. It is argued that the structured and approximate nature of such strategies indicates the need for a "higher-order" model of computation and a new definition of computational complexity.

Keywords: symbolic integration, symbolic-numeric integration, unwinding numbers, rational functions, effective validity, effective logic, feasible computation

Co-Authorship Statement

Chapter 2 was co-authored with Robert Corless and David Jeffrey and will be submitted for publication. Rob Corless supervised the research for this chapter. Chapter 3 was co-authored with Robert Corless, Marc Moreno Maza and Ning Xie and will be submitted for publication. Marc Moreno Maza and Robert Corless supervised the research for this chapter, and Marc Moreno Maza supervised the extensive implementation work for this chapter.

Chapters 4 and 5 are both single-authored. Chapter 5 has been accepted for publication and chapter 4 will be submitted for publication.

Robert Corless reviewed earlier drafts of chapters 4 and 5. Marc Moreno Maza reviewed and revised portions of an earlier draft of chapter 5.

To my wife and best friend, Lori

Acknowledgements

A great many people have contributed to different aspects of the work leading to this dissertation. Principal among them are my supervisors Rob Corless and Marc Moreno Maza. I would not even have entertained this degree had I not wandered into Rob's office one afternoon, looking for things to do after finishing my PhD in philosophy, leading to him suggesting doing a PhD with him in one year. It ended up taking longer, but the journey led to many excellent opportunities to expand my professional and scholarly horizons. This included becoming the local organizer (with Rob Corless, Chris Smeenk and Nic Fillion) of the first multi-disciplinary conference on Algorithms and Complexity in Mathematics, Epistemology and Science, and included many opportunities to develop the ideas from my philosophy PhD by integrating prior work with my growing knowledge of applied mathematics and computational science, while thinking about new approaches to error handling in symbolic integration. This would not have been possible without the enormously flexible and supportive supervision style of Rob Corless.

Taking the graduate course in Distributed and Parallel Systems with Marc Moreno Maza contributed to a rekindling of my love of programming and led to the idea of implementing symbolic integration algorithms in C++ using the basic polynomial algebra subprograms (BPAS) package developed by Marc Moreno Maza, Ning Xie and others. The project for this course led to my being the first "customer" of BPAS, eventually to my joining the project as a developer as my research became intertwined with the project, and now to my continuing the development of BPAS as a postdoctoral researcher. The pleasure of working with Marc has allowed me to implement efficient symbolic computation algorithms in the challenging programming context of the CilkPlus branch of C++, and contributed significantly to my noticing the general pattern of inferentially stable transformations that reduce inferential complexity in symbolic computation and pure mathematics.

Other teachers of mine that have contributed to making this work possible are my supervisors for my philosophy PhD, Bob Batterman and John Bell, and my medical qigong sifu Bob Youngs. My attempt to harmonize Bob Batterman's philosophy of applied mathematics with John Bell's philosophy of pure mathematics has led to the understanding of feasible computation and inference articulated here. Their contributions and insights, as well as those of Rob Corless and Marc Moreno Maza, permeate this dissertation. By teaching me the Daoist principles of good health and longevity, as well as so much more, Bob Youngs has enabled me to find the discipline to maintain good health and to cultivate the energy needed to complete the rigours of this PhD.

This work has also benefited from the support of many colleagues, family and friends. Nic Fillion has always supported and stimulated my research as long as I have known him, through friendship and countless excellent conversations over the years. Emerson Doyle has also been an ardent friend and supporter of my philosophical and mathematical ideas for as long as I have known him, providing a challenging interlocutor and excellent feedback on drafts of many of my papers. Conversations with Sona Ghosh, Veromi Arsiradam and Andrew Fenton have also been valuable in more and less subtle ways. As always, I thank my parents for their steady, loving support from the very beginning of my venture into graduate work, up to its completion now.

I would also like to thank my examiners for their thoughtful, challenging and penetrating comments and questions. In particular: I thank David Jeffrey for his challenges to harmonize results with their careful mathematical presentation, and to meet the combined interest of the reader and implementer of mathematical algorithms; I thank David Stoutemyer for his support of the project and his very careful reading of the entire dissertation, along with numerous thoughtful comments about how to improve the presentation; and I thank Chris Smeenk for his challenges to justify my view of effective logic as a generalization of traditional deductive logic and to articulate ways in which it stands to have normative force.

Most of all I would like to thank my wife Lori, to whom this work is dedicated. She was the first to support the idea of doing a second PhD in applied mathematics, seeing its potential to allow me to more fully develop and integrate my ideas. She has been a consistent source of love and support throughout the process, as well as providing some of the strongest challenges to carefully articulate and justify my thinking, with all of the conceptual benefits that entails.

Contents

Abstract i

Co-Authorship Statement ii

Dedication iii

Acknowledgements iv

List of Figures viii

List of Tables ix

List of Abbreviations, Symbols, and Nomenclature x

**1 Introduction:
Continuity and Complexity Reduction in Scientific Computing** 1

1.1 Motivation . . . 2
1.2 Outline . . . 4
1.3 Bibliography . . . 7

2 An Unwinding Number Pair for Continuous Integrals 9

2.1 Introduction . . . 10
2.2 Unwinding Numbers for Paths on the Riemann Sphere . . . 12
2.3 Restoring Continuity for the Logarithm and Arctangent Functions . . . 17
2.4 Continuous Real & Complex Integration of Rational Functions . . . 26
2.5 Conclusion . . . 36
2.6 Bibliography . . . 37

3 Symbolic-Numeric Integration of Rational Functions 38

3.1 Introduction . . . 39
3.1.1 Symbolic-Numeric integration of Rational Functions . . . 39
3.2 Methods for Exact Integration of Rational Functions . . . 44
3.3 The Algorithms . . . 46
3.4 Analysis of the Algorithm . . . 49
3.5 Implementation . . . 59
3.6 Experimentation . . . 61
3.7 Conclusion . . . 66

3.8 Bibliography . . . 66

4 Effective Validity: A Logical Model for Stable Inference **68**

4.1 Introduction . . . 69
4.2 Stability Under Syntactic and Semantic Variation . . . 74
4.3 Interpreted Languages and Synto-Semantic Stability . . . 85
4.4 Effective Logic and Scientific Problem-Solving . . . 95
4.5 Modeling, Complexity Reduction and Inferential Structure-Preservation . . 105
4.6 Conclusion . . . 108
4.7 Bibliography . . . 109

5 Feasible Computation: Methodological Contributions from Computational Science **111**

5.1 Introduction . . . 112
5.2 Approximate Problem Solving in Physics . . . 114
5.3 Feasible Computation: Algorithmic Problem Solving . . . 123
5.4 Consequences for Computability Theory . . . 132
5.5 Bibliography . . . 134

6 Conclusion: Feasible Computation in the Mathematical Sciences **136**

6.1 Concluding Remarks . . . 137
6.2 Future Work . . . 141
6.3 Bibliography . . . 145

Curriculum Vitae **147**

List of Figures

2.3 The Riemann sphere with its equator as the image of the unit circle. 14

2.7 Discontinuities of the one- and two-argument arctangent functions over the complex plane. 18

2.13 Discontinuities of the complex arctangent function over the complex plane. 22

2.17 Discontinuities of $\frac{1}{2}\left(\arctan\left(\frac{x}{1+y}\right)+\arctan\left(\frac{x}{1-y}\right)\right)+\frac{i}{4}\log\left(\frac{(1+y)^2+x^2}{(1-y)^2+x^2}\right)$ over the complex plane. 24

2.34 Correcting the contour integral of $f(z)=\frac{3z^2+1}{z(z^2+1)}$ for continuity. 35

3.1 MAPLE output for the integral $\int\frac{x^2-1}{x^4+5x^2+7}dx$. 40

3.28 Sample output of `snIntLRT` and `snIntPFD`. 60

3.29 Runtime comparison of `snIntPFD` and `snIntLRT` for integrals of $\frac{1}{x^n-2}$ and $\frac{1}{x^n+x-2}$ 63

4.2 An elliptic curve, the real algebraic variety of $2y^2-x^3+2x-1$. 86

4.3 The simple pendulum model. 89

4.6 The simple harmonic oscillator compared to the simple pendulum for a moderate initial angle. 92

4.7 The simple harmonic oscillator compared to the simple pendulum for a small initial angle over the time interval $[0,6]$. 93

4.8 The double pendulum model. 96

5.7 Solution of the van der Pol equation using MATLAB's `ode45` routine. 128

List of Tables

3.1 Tolerance proportionality of the global forward and backward error bounds for `snIntLRT` and `snIntPFD` on $\int \frac{dx}{x^{128}+2}$. 64

3.2 Tolerance proportionality of the singularity boundary widths for `snIntLRT` and `snIntPFD` on $\int [8,8]_{e^x/x}(x)dx$ for the singularity at $x \doteq 10.949$. 64

List of Abbreviations, Symbols, and Nomenclature

Abbreviations

ApaTools	approximate polynomial algebra tools	43
BPAS	basic polynomial algebra subprograms	59
CAS	computer algebra system	10
HoTT	homotopy type theory	80
LRT	Lazard-Rioboo-Trager algorithm	42
MPSolve	multiprecision polynomial solver	59
NIC	near-identity contisomorphism	82
PFD	partial fraction decomposition	42
QFT	quantum field theory	143
RSR	Rioboo singularity removal algorithm	28
RT	Rothstein-Trager algorithm	42
`snIntLRT`	BPAS implementation of LRT-based symbolic-numeric integration algorithm	59
`snIntPFD`	BPAS implementation of PFD-based symbolic-numeric integration algorithm	59

Symbols

$\lVert f(x)\rVert_\infty$	maximum absolute value of f on an interval	48
$[m,n]_f(x)$	the Padé approximant of order $[m/n]$ of f	61
$\Gamma \vdash_S p$	p is a valid syntactic consequence of Γ with imposed sentences S	88
$\Gamma \models_R p$	p is a valid semantic consequence of Γ with imposed relations R	88
$\Gamma \overset{\sim}{\vdash}_S p$	p is an effectively valid syntactic consequence of Γ with imposed sentences S	88
$\Gamma \overset{\sim}{\models}_R p$	p is an effectively valid semantic consequence of Γ with imposed relations R	88
$\Gamma \Vvdash^{\sim}_{L} p$	p is an effectively valid consequence of Γ in interpreted language L	95
$\Gamma \Vvdash^{\sim}_{L\to L'} p$	p is an effectively valid consequence of Γ when L is interpreted in L' according to transformation $\to$	99
$\mathrm{dc}_c(f)$	dominant coefficient of the Laurent expansion of f at c	19
C	$\mathbb{C}\cup\{\infty\}$, the one-point compactification of $\mathbb{C}$	17
$\mathcal{C}$	contour in the complex plane or Riemann sphere	26
ε	input tolerance	47
$\mathrm{invexp}(z)$	continuous inverse exponential manifold	25

$\mathrm{invtan}(x)$	continuous inverse tangent manifold (real version)	15
$\mathrm{invtan}(z)$	continuous inverse tangent manifold (complex version)	23
lc	leading coefficient of a polynomial	29
$\mathbb{K}$	(computable) real closed field	26
$\mathcal{K}_r(x)$	radial unwinding number	15
$\mathcal{K}_\theta(z)$	angular unwinding number	15
$\mathrm{ord}_c(f)$	the order of f at c	19
R	$\mathbb{R} \cup \{\infty\}$, the one-point compactification of $\mathbb{R}$	14

Nomenclature

angular unwinding number 15
backward inferentially stable transformation 103
clew branch 16
continomorphsism 81
contisomorphsism 81
dominant coefficient 19
effective validity 72
effective logic 71
feasible algorithm 113
feasible problem 113
forward inferentially stable transformation 103
inferential complexity 107
inferentially stable transformation 103
inferential structure 102
invexponential manifold 25
invtangent manifold (complex version) 23
invtangent manifold (real version) 15
pejorative manifold 43
radial unwinding number 15
real circle 14
Rothstein-Trager resultant 46
semantically stable language 87
semidefinite integral 10
syntactically stable language 87
synto-semantically stable language 87

CHAPTER 1

Introduction: Continuity and Complexity Reduction in Scientific Computing

Introduction: Continuity and Complexity Reduction in Scientific Computing

1.1 Motivation

Computational modeling is now commonplace throughout most areas of our society, including science, technology, business and industry. As a consequence, a great deal of belief-formation and decision-making processes rely on the results of computation. The reliability of these beliefs and decisions, then, depends significantly on the reliability of computational tools and methods, and the validity or truth of the results they produce. Reliable belief-formation and decision-making therefore requires reliable computing.

A major challenge to ensuring reliability in computing is that algorithm and mathematical software design typically involve finding appropriate trade-offs between three key desiderata: accuracy, efficiency and generality. Generally speaking, only two of these desiderata can be maintained simultaneously: ensuring accuracy and efficiency typically sacrifices generality, which, for example, is one reason why there are so many different kinds of numerical methods for solving differential equations; ensuring accuracy and generality often sacrifices efficiency, which is why many symbolic algorithms that make weak or no assumptions about a problem are so expensive; ensuring efficiency and generality often sacrifices accuracy, for example because general methods require sacrificing exact computation to be efficient or as a result of the fact that abstraction processes efficiently yield compact general descriptions by eliminating details.

Given that accuracy and efficiency are of paramount concern in algorithm and mathematical software design as well as computational modeling in general, methods, models and their products will all have limitations on their generality. Though this much is obvious, what is not so obvious is *where* the boundaries to accurate and efficient results actually lie.

In numerical computing, this is often traced to the use of asymptotic analysis to justify numerical stability results. For numerical integration of differential equations, such results imply that for a sufficiently fine computational mesh the error (according to a contextually appropriate measure) goes to zero. Whether the needed mesh size is computable in practice is highly dependent on the problem in question.

In symbolic computing, the assumption is often made that because the computation is exact there is no need to be concerned with error or boundaries of validity. Exact computation can hide more subtle forms of error, however. A more obvious example is the fact that exact problems are often the result of a mathematical modeling process that makes approximations. But a more subtle example is where a problem in mathematical analysis is converted into a problem in algebra such that the translation, though exact in some sense, does not respect the continuity of the analytic context. We will consider such a case in detail below.

Computational modeling more generally involves not only the combined reliability concerns of numerical and symbolic computing, but also the reliability of assumptions made about the phenomenon being modeled. Since the phenomenon in question typically escapes independent, precise mathematical description, the reliability of such assumptions generally escapes mathematical analysis. Nevertheless, there is a well-developed area of verification and validation of models that attempts to address such concerns.

Though boundaries of validity of computational methods and models are difficult to specify in general, it is often the case that information about where such boundaries are, or are likely to lie, is accessible computationally given the specification of a particular problem. A central claim of this dissertation is that, where possible, algorithms and implementations should take into account the locations, or likely locations, of boundaries of validity, correcting for them if possible, and providing useful information about boundaries of validity to the user.

A related, and often overlooked, issue in scientific computing is a general lack of awareness among users of mathematical software of the existence of boundaries of validity for the solutions presented. Though users should certainly do their own error analysis, many do not and assume that the results presented are always valid. The overall thesis of this dissertation is that, more generally, there is a need for an increased awareness of the ubiquity of error in computational modeling and how the introduction of error is and can be mitigated. This issue is handled deftly by the developers of efficient algorithms and high quality mathematical software, but an increased clarity concerning the sources of error in computing in general is needed to mitigate the risk of unreliable belief-formation and decision-making based on the results of computational modeling.

We approach this very general and challenging problem in a number of complementary ways. At one level, we consider the development of algorithms for symbolic integration that either avoid encountering boundaries of validity (chapter 2) or provide detailed information about where the results of a computation become unstable (chapter 3). At another level, we address the issue of increasing clarity concerning the sources of error in computing, as well as how reliability can be maintained despite the need for accurate and efficient algorithms. Our contribution in this direction involves the development of a model of the general structure of stable inferences in the presence of error (chapter 4), and a highlighting of the general structure of complexity reduction in scientific computing in particular, and scientific inference more generally (chapter 5). We now consider the content of these four chapters in more detail.

1.2 Outline

In the context of numerical computing, aside from truncation and other forms of discretization error, the reliance on floating point arithmetic in implementation makes it evident that most computation is subject to (rounding) error, so that methods must be demonstrably stable to be reliable. In the context of symbolic computing, however, the assumption is typically made that because computation is exact there is no need to be concerned about error. Corless and Jeffrey[1] have pointed out, however, that an analogue of numerical stability in symbolic computing is *continuity*.

The failure to adequately handle continuity in symbolic computation leads to issues such as algebraic identities for complex functions no longer being generally valid and solutions to symbolic problems that are only valid for certain values of their independent variables or parameters. Thus genuine error can result from inattention to continuity. There are also efficiency concerns, since special cases typically result from limiting instances of generically valid results. Thus, computing the special cases separately can be more expensive than computing the limiting case of an already computed generic result, and therefore efficiency gains can be made by the computing of degenerate cases using limits [1, 2, 3].

Chapter 2 addresses the problem of discontinuous expressions in symbolic integration by extending the unwinding number approach of Corless and Jeffrey [6] for restoring continuity at branch cuts to a method of restoring continuity at branch points. This is accomplished by considering the paths of integrals to traverse the Riemann sphere (or equivalently the one-point compactification of the complex plane) rather than the complex plane. It is shown how discontinuities owing to passes through branch points of the logarithm and arctangent functions can be corrected with a second kind of unwinding number. It is shown how the jump conditions for the unwinding numbers can be com-

[1]Private communication, but see [6, 7] for examples of the error that results from the failure to adequately handle continuity in symbolic computation.

puted for real and complex integrals of rational functions, including rational functions of a transformed variable.

To make this approach work, we consider the codomains of meromorphic functions f, away from poles, to form a smooth two-dimensional complex manifold (or a pair of two-dimensional real manifolds forming the real and imaginary parts), so that paths on the Riemann sphere map to continuous curves in the image manifold. These image manifolds are defined for the logarithm and arctangent, respectively, in terms of data-structures called the *invexponential manifold* and *invtangent manifold*. The unwinding numbers are employed to make these manifolds computable for continuous paths on the Riemann sphere by stitching together branches. Using these manifolds, we can define an antiderivative for meromorphic functions, and can compute contour integrals in terms of the algebraic antiderivative of f evaluated along a parameterization of the contour, rather than by integrating the parameterization.

Chapter 3 considers the hybrid symbolic-numeric integration of rational functions on the assumption of exact input. We consider two algorithms that combine symbolic integration with the numerical evaluation of algebraic numbers using multiprecision numerical rootfinding. One approach, based on a structure-preserving partial fraction decomposition (PFD), approximates the roots of the denominator, computes the full PFD, and then integrates; the other, based on the efficient Lazard-Rioboo-Trager (LRT) symbolic integration algorithm, integrates exactly and then approximates the algebraic numbers appearing in the integral. The hybrid approach allows simpler expressions, for ease of analysis and intelligibility, while producing output in a format that is more easily integrated with other forms of scientific computing. The LRT approach also allows the exact integral to be retained for further symbolic computation. Both methods provide exceptional structured numerical stability, even near singularities of the integrand. We provide an analysis of the (structured) linear forward and backward error of both algorithms, and design the error analysis so that error estimates can be computed concurrently during the execution of the main integration algorithm. Both algorithms are implemented in BPAS [5], which is built using MPSolve [4]. In the implementations, based on a user-supplied tolerance, global error estimates are computed for non-singular problems; for singular problems, bounds are computed around the singularities where the error exceeds the tolerance.

Beginning with chapter 4 we shift to a consideration of the epistemology of mathematics and computational science. Although this is traditionally an area that is consigned to philosophers, the aim of chapter 4 is to contribute to a mathematical approach to epistemology of mathematics. Rather than following the more idealized approaches of formal epistemology common in philosophy, such as [8], the strategy is to build mathematical tools that can elucidate the epistemological structure of mathematics in practice, by treating mathematical models and methods as a complex phenomenon to be investigated using specially designed mathematical modeling tools. Thus, the aim is to elucidate the conceptual and methodological structure of mathematics in much the same way we use models to elucidate the constitutive and behavioural structure of natural phenomena.

Chapter 4 is motivated by the observation that though logic is fundamental to our understanding of scientific inference and computation, much of which involves reasoning reliably in the presence of error, the standard concept of logical validity takes no account of error in inference. The standard concept of valid inference, based in strict deducibility or truth-preservation, is seen to be suited to the strict deductive reasoning found in pure mathematics, rather than the error-aware reasoning ubiquitous in applied mathematics. We show how a generalization of the concept of validity to near-preservation of syntactic and/or semantic structure gives rise to a concept of stable valid inference, which we call *effective validity*. This concept makes an explicit connection between stability of inferences and a generalized form of continuous map. We then consider the basic features of this concept as applied to the forms of interpreted language commonly encountered in mathematical practice. It is shown how the notion of effective validity can be extended to stable transformations between interpreted languages, which is illustrated through an example of the problem transformations involved in the numerical integration of ordinary differential equations. Since such transformations underlie strategies of complexity reduction in computational mathematics, the concept of effective validity clarifies that a basic requirement of such methods is near-preservation of inferential structure, in a sense that is made precise. It is also pointed out that the same basic kind of process is involved in the use of models and theories to describe, predict and control natural phenomena.

As an historical and philosophical coda, chapter 5 examines the strategy in computational science of transforming problems to reduce computational complexity and its origins in the history of science. The concept of *feasible computation*, as a notion of reliable, efficient computation given the constraints encountered in scientific practice, is introduced as the central focus of computational science. It is pointed out that the ubiquitous strategy of transforming problems to reduce computational complexity has an algorithmic structure, being a recursive sequence of reduction steps until a solution becomes feasibly computable, followed by a sequence of back-interpretation steps to produce a solution to the original problem. There are exact and approximate versions of this recursive strategy. The origins of the approximate version of the strategy in the history of physics are considered, through the development of numerical methods, asymptotic expansions and perturbation theory. The ubiquity of the recursive pattern in contemporary computational science is then illustrated through a number of examples in numerical and symbolic computing. It is argued that the robustness of the higher-order structural pattern in computational science should be accounted for by computability theory. Moreover, it is argued that the ability of approximations to move between traditional complexity classes lead to a new way of viewing the computational complexity of a problem.

Overall, this dissertation is a mathematical and epistemological contribution to the clarification of the concepts of error and efficiency in computational science. This is accomplished by contributing new algorithms for error handling in the context of exact and approximate symbolic integration, by developing a general logical model for stable, efficient inference in computational science and scientific inference more broadly, and by identifying a robust algorithmic structure to methods in the mathematical sciences that leads to new ways of thinking about computational complexity.

1.3 Bibliography

[1] Parisa Alvandi, Changbo Chen, and Marc Moreno Maza. Computing the limit points of the quasi-component of a regular chain in dimension one. In *International Workshop on Computer Algebra in Scientific Computing*, pages 30–45. Springer, 2013.

[2] Parisa Alvandi, Mahsa Kazemi, and Marc Moreno Maza. Computing limits of real multivariate rational functions. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, pages 39–46. ACM, 2016.

[3] Parisa Alvandi, Marc Moreno Maza, Éric Schost, and Paul Vrbik. A standard basis free algorithm for computing the tangent cones of a space curve. In *International Workshop on Computer Algebra in Scientific Computing*, pages 45–60. Springer, 2015.

[4] Dario A Bini and Leonardo Robol. Solving secular and polynomial equations: A multiprecision algorithm. *Journal of Computational and Applied Mathematics*, 272:276–292, 2014.

[5] Changbo Chen, Svyatoslav Covanov, Farnam Mansouri, Marc Moreno Maza, Ning Xie, and Yuzhen Xie. Basic polynomial algebra subprograms. *ACM Communications in Computer Algebra*, 48(3/4):197–201, 2015.

[6] Robert M Corless and David J Jeffrey. The unwinding number. *ACM SIGSAM Bulletin*, 30(2):28–35, 1996.

[7] David J Jeffrey. Integration to obtain expressions valid on domains of maximum extent. In *Proceedings of the 1993 international symposium on Symbolic and algebraic computation*, pages 34–41. ACM, 1993.

[8] Kevin T Kelly. *The logic of reliable inquiry.* Oxford University Press, 1996.

CHAPTER 2

An Unwinding Number Pair for Continuous Integrals

An Unwinding Number Pair for Continuous Integrals[1]

2.1 Introduction

This paper contributes to the solution of a problem raised by Jeffrey [3] concerning the expressions of integrals generated by computer algebra systems (CAS). Such expressions often have spurious discontinuities that result from a branch of a multivalued function exceeding its domain of definition. Since such discontinuities are not inherent to the integral, the integration algorithm ought to correct for them and return a result that is continuous on the maximum domain over which the integral is continuous. This has therefore been called the problem of obtaining integrals on domains of maximum extent.

After nearly 25 years since Jeffrey's paper this issue does not have an adequate resolution. If we now ask Maple or Mathematica to integrate $2/(1+3\sin^2 t)$ we are presented with the result

$$\int \frac{2\,dt}{1+3\sin^2 t} = \arctan\left(2\tan(t)\right), \tag{2.1}$$

which is a valid *algebraic* antiderivative but is discontinuous at the singularities of the tangent function appearing in the argument of the arctangent function, which occur whenever $t = \left(n+\frac{1}{2}\right)\pi$, $n \in \mathbb{Z}$. Since the arctangent function is defined as the principal branch of $\arctan z$, rather than increasing without bound as $t \to \infty$, the integral cycles through values in the bounded interval $\left(-\frac{\pi}{2}, -\frac{\pi}{2}\right)$.

We trace the root of the problem to the widely held assumption that for symbolic integration CAS and the Risch algorithm and its variants, based in differential algebra, deal with antiderivatives. If the principal constraint on a valid solution to a symbolic integration problem is that its derivative recovers the input function, then there is no need to be concerned about domains of definition or branch choices and cut locations of multivalued functions. One must be aware of these issues, however, to obtain expressions continuous on domains of maximum extent. Partly for this reason, we take the *definite* integral to be fundamental, understood in the following sense. Symbolic integration algorithms should return a *semidefinite integral*, meaning

$$F(x) = \int_a^x f(t)dt + c(a),$$

[1]A version of this paper will be submitted for publication in an appropriate refereed journal.

but where a can take any valid value in the domain of f and such that $F(x) - F(a)$ evaluated at x (with x in the same connected component of the domain of f as a) must yield the correct value of the definite integral. Since such an expression is unambiguously an antiderivative of $f(x)$, we still solve the algebraic problem, but algorithms must also be aware of correct domains of definition and continuity. All we are really requiring here is that the antiderivatives returned by integration algorithms vary continuously as the independent variable is varied when the expression of the integral is continuous.

Taking the (semi)definite integral as fundamental in symbolic integration (semidefinite integral is returned, evaluates correctly to the definite integral) simultaneously addresses the continuity of the antiderivative and the problem of continuity in parameters on account of the following

Theorem 2.2 *If $f(x,\xi)$ is separately continuous in x and ξ on a compact, connected region $\mathcal{D}$ of $\mathbb{R}^2$, then*

$$F(x;\xi) = \int_a^x f(t,\xi)dt,$$

is continuous in x for each ξ and continuous in ξ for each x.

Proof. For fixed ξ, the fundamental theorem of calculus implies $F(x;\xi)$ is differentiable hence continuous. Fix x and let $\{\xi_n\}$ be any sequence of parameters such that $(x,\xi_n) \in \mathcal{D}$ and $\xi_n \to \xi$. Then $F(x;\xi_n)$ is continuous in x for each n by the previous result. Then, $\sup_{t\in[a,x]}|F(t;\xi_n) - F(t;\xi)| \to 0$ as $n \to \infty$, so that $F(x;\xi_n)$ converges uniformly to $F(x;\xi)$. The continuity of $F(x;\xi)$ then follows from the continuity of each $F(x;\xi_n)$. □

The method for correcting discontinuous integrals we present uses an extension of the unwinding number concept introduced by Corless and Jeffrey [2]. We accomplish this by approaching the problem of multivaluedness in complex analysis from a different perspective that takes path continuity (or path analyticity) as fundamental. The outcome of our approach is that the codomain of a complex function $f(x+iy) = u(x,y) + iv(x,y)$ is treated as a pair of 2 dimensional real manifolds (corresponding to $u(x,y)$ and $v(x,y)$) generated by correcting the spurious discontinuities that arise from branch crossings. These manifolds form continuous surfaces for continuous complex functions, though they may be multisheeted, which introduces path-dependence. Then for a real parameter t such that $z(t)$ traces out a path on the Riemann sphere, and finite $f(z(t))$, the image of $z(t)$ under f is a pair of continuous space curves, that are the real and imaginary parts of $u(x(t),y(t)) + iv(x(t),y(t))$. As such, the approach can be seen as a kind of harmonization of branching and Riemann surfaces. To accomplish this we must extend the usual unwinding number method of handling branch cut crossings by also correcting spurious discontinuities that arise from the crossing of branch points.

2.2 Unwinding Numbers for Paths on the Riemann Sphere

One of the reasons why the introduction of spurious discontinuities in expressions of integrals is a generic issue for the integration problem is (the strong version of) Liouville's theorem, which states that for f in a differential field F of characteristic 0 with constant field K, then if $g = \int f$ can be expressed in finite terms such that g lies in an elementary extension field G of F, then we can obtain an expression of the integral as

$$\int f = \varphi_0 + \sum_{k=1}^{n} c_k \log(\varphi_k),$$

where log is the natural logarithm, $\varphi_0, \varphi_k \in F$ and $c_k \in K$ for all $k = 1, \ldots, n$. It therefore follows that if any of the functions φ_k have a complex value, $\varphi_j = u + i\,v$, then the expression of the integral contains a term the can be expressed in the form

$$c_j \log(\varphi_j) = c_j \log(u + i\,v) = c_j \log\left(\sqrt{u^2 + v^2}\right) + c_j\, i \arctan\left(\frac{v}{u}\right),$$

when the term is expressed in terms of its real and imaginary parts. Notice that if v/u has an odd order pole lying in the path of integration, the integral of f will have a spurious discontinuity of a similar form to the first example in the previous section. In addition, where φ_j crosses the branch cut of the logarithm we will also have a spurious discontinuity arising from the choice of branch of the logarithm function. Thus, two kinds of spurious discontinuities (from logarithmic branch cut crossing and arctangent argument singularities) are quite generic in the integration of functions in finite terms on account of Liouville's theorem.

The unwinding number approach introduced by Corless and Jeffrey [2] provides a method of restoring continuity for expressions of integrals that implicitly cross branch cuts of complex functions. This approach does not, however, address the issue of discontinuities due to diverging arguments of functions like the arctangent function, which tolerate singular arguments without themselves diverging. This latter variety of discontinuity is treated by Jeffrey [3] and Jeffrey and Rich [4] by introducing Heaviside step functions at the points of discontinuity to ensure a continuous result. The approach that we propose here handles both kinds of spurious discontinuity by combining these approaches into a uniform framework, tying them both to the geometry of complex functions.

The original unwinding number approach deals with the case where the rotation of the input to a direct function in the complex plane leads to multiple coverings of the complex plane, allowing one to unwind these rotations to seamlessly join the branches of the mulitvalued inverse function. In this way, the unwinding number can be regarded as unwinding rotations about the origin. By restricting attention to paths and shifting to viewing rotations on the Riemann sphere rather than the Argand plane, we can interpret the discontinuities due to singular arguments as the result of windings of paths around the Riemann sphere through the point at infinity. These paths can then be unwound in terms of a second kind of unwinding number. Provided that a continuous finite path crosses two (or more) odd order poles of a direct function, more than one branch is needed to define the inverse function, which ought to be continuous since the direct function follows a continuous path through the poles. For a smooth curve through such a pole, the path on the Riemann sphere passes through the point at infinity and emerges safely from the other side. This makes the inverse functions multivalued with branch points at ∞. This kind of motion through poles can occur (implicitly or explicitly) for real, line and contour integrals. Therefore, a winding number that tracks the motions through infinity can be used to unwind these motions and yield correct continuous expressions, where the integrals contain inverses of multiply covering direct functions.

Before introducing the two unwinding numbers, some further reflection on the geometry of the two kinds of discontinuities is warranted. Branch cuts for complex functions occur along curves in the complex plane, which can be finite or semi-infinite depending on the function. Regarded on the Riemann sphere, therefore, branch cuts are always curves joining two points on the Riemann sphere. Crossing of these curves is therefore the kind of discontinuity for which the traditional unwinding number was devised. The kind of discontinuities resulting from passes through infinity are thus seen to be the result of passing through a hidden branch point at infinity. Discontinuities can also arise from the other end of the branch cut, however, and can be treated in precisely the same way. Indeed, as we will see, to render some inverse functions continuous we will also need to unwind passes through the origin when there is a branch point there. In this case, it is only passes *through* zero that lead to discontinuities, just as for the point at infinity. Since only odd multiplicity zeros pass through zero, only they generate discontinuities. Just as for even order poles, even multiplicity zeros "bounce" off of the branch point

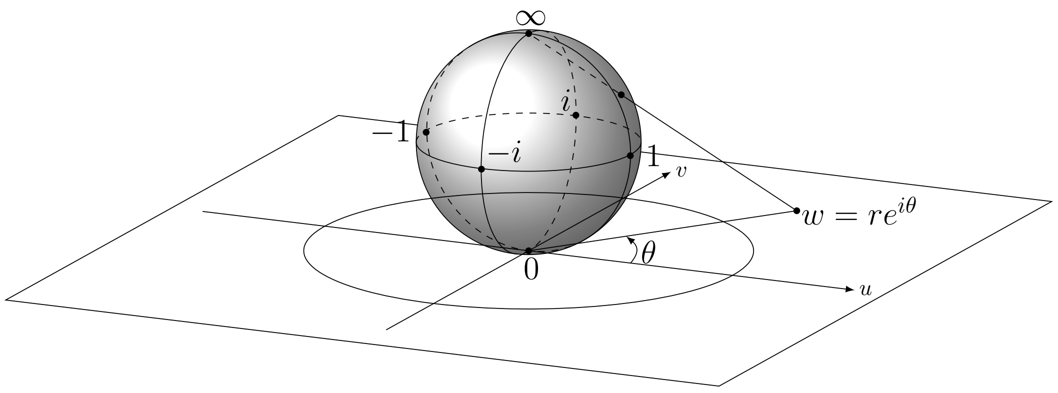


Figure 2.3: The Riemann sphere with its equator as the image of the unit circle.

without passing through it.

To solve the continuity problem for integrals, then, we must solve the continuity problem for inverse and implicitly defined functions. Given the centrality of the logarithm and arctangent functions for the integration problem given Liouville's theorem, we restrict our attention here to the restoration of continuity for these functions. For these functions it is possible to restore of continuity using only the unwinding numbers for the real one-argument arctangent function, as we will see.

To illustrate the issue and its resolution, consider the real tangent function and its inverse. Notice that the image of the real line on the Riemann sphere is a great circle if we include the point at infinity (see figure 2.3). We will refer to the image of $\mathsf{R} = \mathbb{R} \cup \{\infty\}$ on the Riemann sphere as the *real circle*. We will analyze the image of $\tan x$ in terms of the real circle.[2] In this case, if we increase x from 0 to π, then $\tan x$ completes a full counterclockwise rotation around the real circle, starting from zero, reaching the point at infinity for $x = \frac{\pi}{2}$ by approaching $+\infty$ on the real line, passing through the point at infinity and reconnecting with the real line as it approaches $-\infty$ and then returning to 0 along the negative real axis. If we analyze the domain of $\arctan x$ in terms of the real circle, then the standard definition of $\arctan x$ takes values $(-\pi/2, \pi/2)$ on $\mathbb{R}$, and we may take $\arctan(\infty) = \pi/2$. Starting at 0, then, we can only make a half counterclockwise rotation around the real circle before encountering the discontinuity from the branch point at ∞. Thus, we need multiple branches of the real arctangent, which we may denote $\arctan_k x$, that take values in the interval $\left(\frac{(2k-1)\pi}{2}, \frac{(2k+1)\pi}{2}\right]$, so that $\arctan x = \arctan_0 x$.

To keep track of the branch of the real arctangent then, we can introduce a new

[2]The set R is the one-point compactification of the real line. It is more convenient to work with R as a set since it provides a space topologically equivalent to the real circle on the Riemann sphere but is identical to $\mathbb{R}$ for finite values. Hence using R we achieve the same ability to consider paths through ∞ while avoiding the need to deal with stereographic projection onto the real circle.

unwinding number $\mathcal{K}_r(x)$, defined by[3]

$$\arctan(\tan x) = x - \pi\mathcal{K}_r(x), \tag{2.4}$$

analogous to the unwinding number for branch cut crossings defined by

$$\ln(e^z) = z - 2\pi\, i\, \mathcal{K}_\theta(z). \tag{2.5}$$

Based on the polar representation of a complex number as $z = re^{i\theta}$, since $\mathcal{K}_\theta(z)$ unwinds rotations around the origin in the complex plane, we call it the *angular unwinding number*, hence the choice of subscript. Since $\mathcal{K}_r(x)$ corrects discontinuities due to extreme values of r (∞ in this case), we call it the *radial unwinding number*. Substituting $\arctan_k(x)$ for x in (2.4), we then have

$$\arctan_k(x) = \arctan(x) + \pi\mathcal{K}_r(\arctan_k(x)) = \arctan(x) + \pi k.$$

Now, for the integration problem, to obtain continuous expressions for integrals we require functions that remain continuous when the path of integration crosses a branch cut or branch point. Such functions must be able to pass smoothly through branch points or cuts without encountering any discontinuity. For the inverse tangent, this can be accomplished by defining a continuous codomain for the inverse tangent, set up so that it can be used to compute continuous images for a path on the real circle. We call the continuous codomain the *real invtangent manifold*, for reasons we will make clear shortly. Using the notation $\operatorname{invtan}(x)$, with the property that for $x \in \mathbb{R}$, we let

$$\operatorname{invtan}(x) = \arctan(x), \tag{2.6}$$

but for any continuous path $\varphi(t)$ in R (with $x \cdot \infty = \infty$ for $x \in \mathbb{R}\backslash\{0\}$), *i.e.*, the one point compactification of the reals,

$$\operatorname{invtan}(\varphi(t)) = \arctan(\varphi(t)) + \pi\mathcal{K}_r \operatorname{invtan}(\varphi(t))$$

[3] We note that this is, strictly speaking, not a new unwinding number because it can be recovered from the definition of the standard unwinding number ($\mathcal{K}_r(x) = \mathcal{K}_\theta(2x\, i)$), as is clear from their definitions in terms of the ceiling function (see below). We introduce the radial unwinding number as a distinct notion, however, because is conceptually distinct on account of its being defined as a function of a real variable and used to unwind paths on the Riemann sphere, rather than being defined as a function of a complex variable and used to unwind multiple coverings of the entire complex plane.

is continuous. Thus, $\operatorname{invtan}(x)$ is capable of changing branches continuously, and the continuity is ensured by computing the radial unwinding number for the branch $\operatorname{invtan}(x)$ is currently on. As such, $\operatorname{invtan}(x)$ is truly a data structure instead of a mathematically defined function, since it amalgamates the different branches of $\arctan x$ such that branches can be crossed smoothly, but can only be evaluated as a function for x within the domain of the default branch of $\operatorname{invtan}(x)$, given by (2.6), which we call the *clew branch*.[4] For paths $\varphi(t)$ that cross branch points, the radial unwinding number can correct the discontinuity so that the function defined by $t \mapsto \operatorname{invtan}(\varphi(t))$ is a continuous function on the entire domain of the path.[5] Because the invtangent stitches together the different branches of the arctangent function, the image curve forms a smooth one-dimensional manifold.

Note that there is freedom in terms of how we define the clew branch of $\operatorname{invtan}(x)$. We have chosen the principal branch of the arctangent function according to (2.6). Insofar as the invtangent is employed for continuous integrals, the choice of clew will not affect the value of the integral, since it only changes the constant of integration. For predictable behaviour for the user, however, the choice (2.6) is the natural one.

It is useful to note that the radial unwinding number can be defined alternatively as[6]

$$\mathcal{K}_r(x) = \left\lceil \frac{x - \frac{\pi}{2}}{\pi} \right\rceil,$$

analogously to the alternative definition of $\mathcal{K}_\theta$ as

$$\mathcal{K}_\theta(z) = \left\lceil \frac{\mathsf{Im}(z) - \pi}{2\pi} \right\rceil.$$

In terms of the real invtangent we find that

$$\mathcal{K}_r(\operatorname{invtan}(x)) = \left\lceil \frac{\operatorname{invtan}(x) - \frac{\pi}{2}}{\pi} \right\rceil,$$

[4]Given the labyrinthine nature of branches of (real and complex) functions and their conceptual navigation, we name the starting, default, branch of a function for the mythical ball of thread given by Ariadne to Theseus to allow him to navigate the labyrinth. The name seems appropriate given that we make branch crossings navigable by unwinding paths on the Riemann sphere, just as Theseus navigated the labyrinth by unwinding the spherical clew held by Ariadne.

[5]As a result, we could define the invtangent alternatively as a function of two arguments, a point $x \in \mathbb{R}$ and a path φ on the real circle. The proper consideration of this approach is a subject for future work.

[6]Note that this imposes closure on the right of each interval on which the unwinding number is constant. We could impose closure on the left by defining instead $\mathcal{K}_r(\operatorname{invtan}(x)) = \left\lfloor \frac{x+\frac{\pi}{2}}{\pi} \right\rfloor$.

which makes it evident that the value of the radial unwinding number for the invtangent changes when the invtangent crosses an odd multiple of $\frac{\pi}{2}$. This makes the connection to odd poles, since it is odd poles that cause the invtangent to change branches, hence to cross an odd multiple of $\frac{\pi}{2}$.

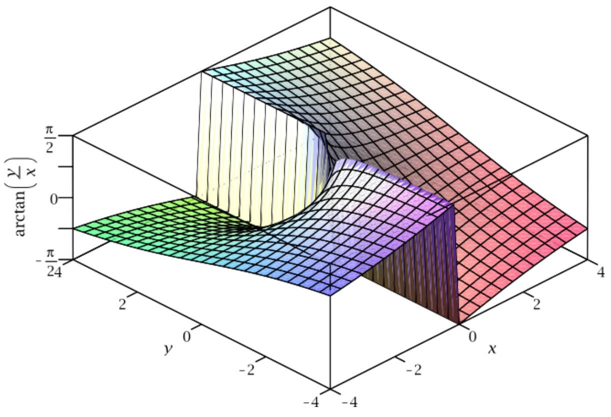
The above introduction of the notation $\operatorname{invtan}(x)$ is important and requires some comment. Part of the difficulty of handling complex functions for CAS is that the typical user does not understand the subtleties of branching and multivaluedness. The power of this definition is that it hides the unwinding numbers that the CAS can use to compute a continuous inverse tangent, presenting only to the user an expression that is known to be path-continuous (indeed analytic) for continuous changes of x, even when x is replaced by a function with odd order poles. This has advantages for both developers and users of CAS.

For developers, they are free to choose whichever system of unwinding numbers they desire to compute the continuous expressions because the manifold defining the continuous inverse itself is unique (up to a constant that is irrelevant for definite integration problems). On the other hand, for users of CAS, they only need to deal with the continuous function, which avoids any need to be concerned with the subtleties of multivaluedness and allows them to evaluate integrals simply by using the fundamental theorem of calculus. As we will see in section 2.4, this can be accomplished for complex as well as real functions that are continuous on $\mathsf{C} = \mathbb{C} \cup \{\infty\}$, the one-point compactification of the complex plane. Thus, the user is not confronted with the complexities of unwinding numbers, and only needs to understand that $\operatorname{invtan}(x)$ (or $\operatorname{invtan}(z)$) specifies a continuous multisheeted codomain such that a path over R (or C) determines a continuous function.

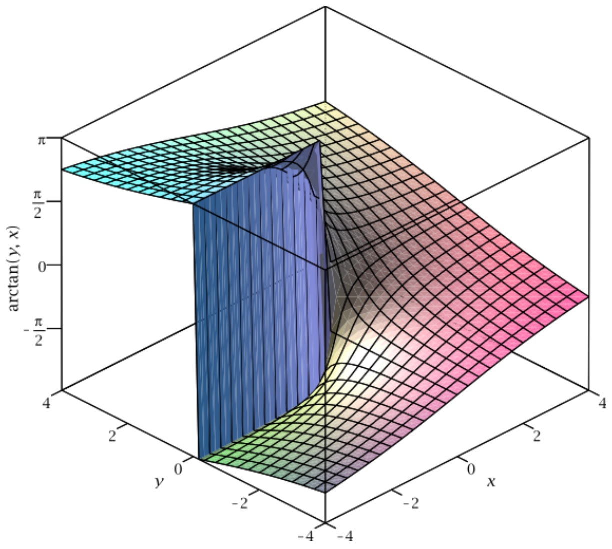
2.3 Restoring Continuity for the Logarithm and Arctangent Functions

Since the real arctangent functions are central in continuous integration as a result of Liouville's theorem, as pointed out above, we begin with a consideration of how to restore continuity using unwinding numbers for $\arctan x$, $x \in \mathbb{R}$, and $\arctan(y, x)$, $x, y \in \mathbb{R}$. We will then proceed to a method for restoring continuity of the complex arctangent and logarithm functions in terms of the real arctangent functions.

For analyzing the discontinuous behaviour of $\arctan x$, we must consider the cases in which poles of the argument can generate discontinuities. To this end, we consider the function $\arctan(y/x)$ and its behaviour at zeros of x. Letting $z = x + i\,y = re^{i\theta}$, $-\pi < \theta \leq \pi$,



(a) The one-argument arctangent function $\arctan(y/x)$ has discontinuities of magnitude π at zeros of x.



(b) The two-argument arctangent function $\arctan(y, x)$ has discontinuities of magnitude 2π at logarithmic branch cut crossings, *i.e.*, where x is negative and y changes sign.

Figure 2.7: Discontinuities over the complex plane, with $z = x + iy$, for the (a) the one-argument and (b) two-argument arctangent functions. Notice that in (a) the branch point crossing of $\arctan x$ at ∞ is expanded into two branch cuts over $\mathbb{C}$, one for $x \in (-\infty, 0)$ and one for $x \in (0, \infty)$ and that in (b) the discontinuity is precisely the branch cut of the complex logarithm function.

it follows that

$$\arctan(y/x) = \begin{cases} \theta & x \geq 0 \\ \theta - \pi & x < 0, y \geq 0 \\ \theta + \pi & x < 0, y < 0 \end{cases} .$$

This function is discontinuous along the entire y axis, where the function drops by π if we cross $x = 0$ for positive y and increases by π if we cross $x = 0$ for negative y (see figure 2.7a). There is no discontinuity along the negative real axis because for $x < 0$, $y = 0$, $\arctan(y/x) = 0$ and for $x < 0$, $y < 0$, $\lim_{y \to 0} \arctan(y/x) = 0$.

The source of the discontinuity here are points where $y \neq 0$ and x changes sign, which correspond to poles of y/x of order one. Here, the argument of the arctangent goes through the point at infinity, so that the value of $\arctan(y/x)$ jumps by $\pm\pi$, the direction of the jump depending on whether y/x starts off approaching $-\infty$ (increase by π) or approaching $+\infty$ (decrease by π). In terms of y and x, this depends on the sign of y and the direction x crosses zero (increase for $y < 0$ and x goes from negative to positive or $y > 0$ and x goes from positive to negative, and decrease under the contrary conditions). Such discontinuities are not the only possibilities, however, since allowing x and y to be functions of some parameter opens up a richer spectrum of possible discontinuities.

Before stating the basic results for the discontinuities of $\arctan x$, let us first fix some notation. For the purposes of this paper we will restrict our attention to correcting discontinuities of $\log(f(z))$ and $\arctan(f(z))$ for functions f that have a Laurent expansion, at least locally around a branch point or cut. For any such function $f(z)$, at each point $c \in \mathbb{C}$ there is some $k \in \mathbb{Z}$ such that we can express the function as

$$f(z) = (z - c)^k \sum_{i=0}^{\infty} a_i (z - c)^i. \tag{2.8}$$

We then have the following

Definition 2.9 *Let $f(z)$ be a meromorphic function of a complex variable z and let $c \in \mathbb{C}$. If the Laurent series of f about the point c exists then the* order of f at c, *denoted* $\operatorname{ord}_c(f)$, *is the value of k in equation (2.8) and the coefficient a_0 will be called the* dominant coefficient at c, *which we denote by* $\operatorname{dc}_c(f)$.[7]

Where the point c of expansion is clear from the context, we will omit the subscripts and use the notation $\operatorname{ord}(f)$ for the order and $\operatorname{dc}(f)$ for the dominant coefficient. We then have the following

Lemma 2.10 (One-Argument Arctangent) *Let $u(x)$ and $v(x)$ be real analytic functions except at isolated poles such that, at some point $c \in \mathbb{R}$, $\operatorname{ord}(v) = p$, $\operatorname{ord}(u) = q$. Then for* $\arctan(v/u)$, *$\Delta\mathcal{K}_r \neq 0$ at c if $p - q$ is odd and $p < q$. In such a case,*

$$\Delta\mathcal{K}_r = -\operatorname{sign}(\operatorname{dc}(u)\operatorname{dc}(v)).$$

Proof. Under the assumptions of the theorem we have that $v(x) = (x-c)^p(a_0+a_1(x-c)+\cdots)$ and $u(x) = (x - c)^q(b_0 + b_1(x - c) + \cdots)$, so that

$$\arctan(v/u) = \arctan((x - c)^{p-q}(a_0/b_0 + \cdots)),$$

which is asymptotic to $\arctan((x - c)^{p-q}(a_0/b_0))$ as $x \to c$. For this to produce a discontinuity we must have an odd order pole, meaning $p - q < 0$ is odd, implying that $p < q$. The direction of the jump is determined by the sign of a_0/b_0. If $a_0/b_0 > 0$, so that $\operatorname{sign}(\operatorname{dc}(u)\operatorname{dc}(v)) = 1$, then $\lim_{x \to c^-} \arctan(v/u) = -\frac{\pi}{2}$ and $\lim_{x \to c^+} \arctan(v/u) = \frac{\pi}{2}$.

[7] We could use the term *leading coefficient* for a_0, since this is the appropriate term for formal power series with no infinite descending chains in the term ordering. However, this terminology could easily be confused with the same term in the context of polynomials, which we also use and has a different meaning. Hence, we elect for the term *dominant coefficient* here.

Thus, the function increases by π, so that $\Delta\mathcal{K}_r = -1 = -\mathrm{sign}(\mathrm{dc}(u)\mathrm{dc}(v))$. Similarly, if $a_0/b_0 < 0$, the function decreases by π, and $\Delta\mathcal{K}_r = 1 = -\mathrm{sign}(\mathrm{dc}(u)\mathrm{dc}(v))$. □

The two-argument arctangent function $\arctan(y,x)$, defined as

$$\arctan(y,x) = \theta, \quad -\pi < \theta \le \pi,$$

with $z = x + i\,y = r\,e^{i\theta}$, is the natural one when working over the entire complex plane, since the only obvious discontinuity occurs along the logarithmic branch cut on the negative real axis, requiring the angular unwinding number to restore continuity (see figure 2.7b). Unfortunately, this function suffers from a variety of more subtle forms of discontinuity resulting from poles and common zeros of its two arguments, both leading to changes of θ of $\pm\pi$ and both requiring the radial unwinding number to correct. It also has changes of θ of $\pm 2\pi$ along the negative real axis, requiring the angular unwinding number to correct. Thus, the computation of the jump conditions required to render it continuous is much more involved than the one-argument arctangent. The following lemma details the conditions under which $\arctan(y,x)$ can be discontinuous.

Lemma 2.11 (Two-Argument Arctangent) *Let $u(x)$ and $v(x)$ be real analytic functions except at isolated poles, neither identically zero, such that, at some point $c \in \mathbb{R}$, $\mathrm{ord}(v) = p$ and $\mathrm{ord}(u) = q$. Then $\arctan(v,u)$ has non-zero unwinding numbers under the following conditions. If p is even, q is odd, and $p > q$, then $\Delta\mathcal{K}_r \ne 0$ at c and*

$$\Delta\mathcal{K}_r = -\mathrm{sign}(\mathrm{dc}(u)\mathrm{dc}(v)).$$

If either (a) p and q are odd, or (b) p is odd, q is even and $p < q$, then $\Delta\mathcal{K}_r \ne 0$ at c and

$$\Delta\mathcal{K}_r = -\mathrm{sign}(\mathrm{dc}(v)).$$

If p is odd, q is even, $p > q$ and $\mathrm{sign}(\mathrm{dc}(u)) = -1$, then $\Delta\mathcal{K}_\theta \ne 0$ at c and

$$\Delta\mathcal{K}_\theta = -\mathrm{sign}(\mathrm{dc}(v)).$$

Proof. Under the assumptions of the theorem we have that $v(x) = (x-c)^p(a_0 + a_1(x-c) + \cdots)$ and $u(x) = (x-c)^q(b_0 + b_1(x-c) + \cdots)$, so that

$$\arctan(v,u) = \arctan((x-c)^p(a_0 + \cdots), (x-c)^q(a_0 + \cdots)),$$

which is asymptotic to

$$\arctan((x-c)^p a_0, (x-c)^q b_0) \text{ as } x \to c. \tag{2.12}$$

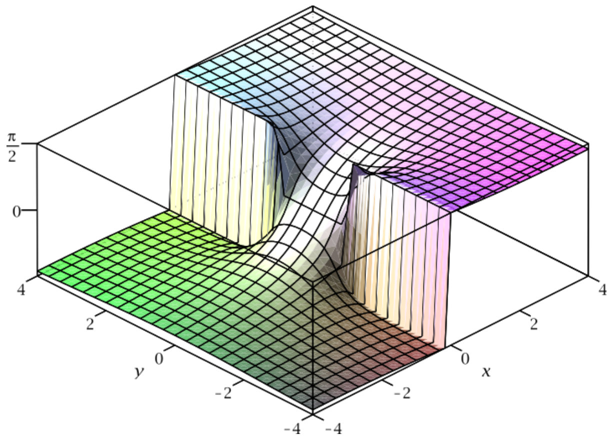
Suppose that p is even, so that $(x-c)^p$ is always positive and $\text{sign}(v) = \text{sign}(a_0)$. Then (2.12) simplifies to $\arctan(a_0, (x-c)^{q-p} b_0)$. If $q-p > 0$ there is no discontinuity, since $\arctan(v,u)$ is continuous at $u = 0$ for $v \neq 0$. If $q-p < 0$, however, so that $p > q$, and $q-p$ is odd, so that u has an odd order pole, then u passes through ∞ and θ changes by $\pm\pi$. Note that $\lim_{x \to c^-} \arctan(a_0, (x-c)^{q-p} b_0)$ is 0 if $b_0 > 0$ and $\pm\pi$ if $b_0 < 0$, taking the value π if $a_0 > 0$ and $-\pi$ if $a_0 < 0$. Consequently, θ increases by π if $a_0 > 0$ and $b_0 > 0$ or if $a_0 < 0$ and $b_0 < 0$, *i.e.*, if $\text{sign}(a_0 b_0) = 1$. Thus, $\Delta\mathcal{K}_r = -\text{sign}(\text{dc}(u)\text{dc}(v))$. Similarly, if $\text{sign}(a_0 b_0) = -1$, $\Delta\theta = -\pi$ and $\Delta\mathcal{K}_r = -\text{sign}(\text{dc}(u)\text{dc}(v))$.

Suppose then that p is odd. If q is also odd then both u and v cross either 0 or ∞. Since p and q differ by an even number, equation (2.12) reduces to $\arctan((x-c)^{p-q+1} a_0, (x-c) b_0)$. Using this expression, if $p > q$, u is infinitesimal relative to v, and both u and v change sign. Since the sign of a_0 determines which side of the logarithmic branch cut we are on (before the zero crossing if $b_0 > 0$ or after the zero crossing if $b_0 < 0$), $\Delta\theta = \pi$ if $\text{sign}(a_0) = 1$ and $\Delta\theta = -\pi$ if $\text{sign}(a_0) = -1$. Thus, $\Delta\mathcal{K}_r = -\text{sign}(\text{dc}(v))$. If $p < q$, then essentially the same analysis applies to $\arctan((x-c)a_0, (x-c)^{q-p+1} b_0)$.

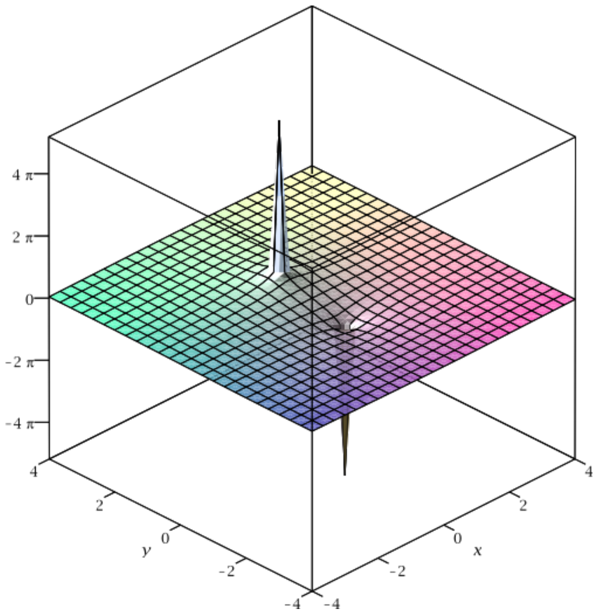
Suppose then that q is even with p odd, in which case equation (2.12) reduces to $\arctan((x-c)^{p-q} a_0, b_0)$. If $p-q < 0$, then v has an odd order pole and v passes through ∞ changing θ by $\pm\pi$. Since this jump avoids the logarithmic branch cut, it is solely the sign of a_0 that determines which direction v goes to ∞, going to $+\infty$ if $a_0 < 0$ and $-\infty$ if $a_0 > 0$. Thus, $\Delta\theta = \pi\text{sign}(a_0)$, so that $\Delta\mathcal{K}_r = -\text{sign}(a_0) = -\text{sign}(\text{dc}(v))$. If $p-q > 0$, then v crosses 0, encountering the logarithmic branch cut if $b_0 < 0$. In such conditions, $\Delta\theta = 2\pi \cdot \text{sign}(a_0)$, so that $\Delta\mathcal{K}_\theta = -\text{sign}(\text{dc}(v))$. □

The two-argument arctangent is notable as an example of a real function that requires both the angular and radial unwinding number to restore continuity. This shows that both unwinding numbers are needed in general. But because the one-argument arctangent exists as an alternative, and it not only does not need the angular unwinding number but also has non-zero $\Delta\mathcal{K}_r$ under more restricted conditions, it is simpler to compute. It also has the advantage of being more familiar to the typical user of CAS. For these reasons we will rely more heavily on the one-argument arctangent function for the results in the following section.

The above two lemmas give the general conditions under which unwinding numbers



(a) The real part of the complex arctangent function arctan(z) has discontinuities of magnitude π along the imaginary axis outside of $[-i, i]$.



(b) The imaginary part of the arctangent function arctan(z) has logarithmic singularities at $z = \pm\pi$.

Figure 2.13: Discontinuities over the complex plane, with $z = x + i\,y$, for the (a) the real part and (b) imaginary part of the complex arctangent function.

can restore spurious discontinuities of the real arctangent functions. We now turn to consider the complex arctangent function $\arctan z$. For $z \in \mathbb{C}$ as before, we can define the complex arctangent function as

$$\arctan z = \frac{i}{2}\left(\log(1 - i\,z) - \log(1 + i\,z)\right). \tag{2.14}$$

A plot of this function is given as figure 2.13. The two logarithms have branch cuts on the imaginary axis extending downward from $-i$ and upward from i, respectively.

Restoring continuity requires that continuous motions in the domain produce continuous motions in the value in the codomain, such that the correct analytic behaviour of the function is obtained. This can be implemented in various ways, however. As such, the discontinuities of $\arctan z$ can be corrected directly from the form (2.14) using the unwinding numbers for the two-argument arctangent function, but it will prove easier to adopt a different approach. By taking mathematically equivalent expressions for the function we can move around branch cut and point locations without changing the local geometry, so that once the discontinuities are removed the same continuous manifold is obtained. Accordingly, we are free to use such equivalent expressions to compute the continuous codomain if they lead to simpler means of computing the corrections to discontinuity.

Thus, we analyze the jump conditions for the radial unwinding number by rewriting the complex arctangent in the form, *viz.*,

$$\arctan(z) = \frac{1}{2}\left(\arctan\left(\frac{x}{1+y}\right) + \arctan\left(\frac{x}{1-y}\right)\right) + \frac{i}{4}\log\left(\frac{(1+y)^2+x^2}{(1-y)^2+x^2}\right). \tag{2.15}$$

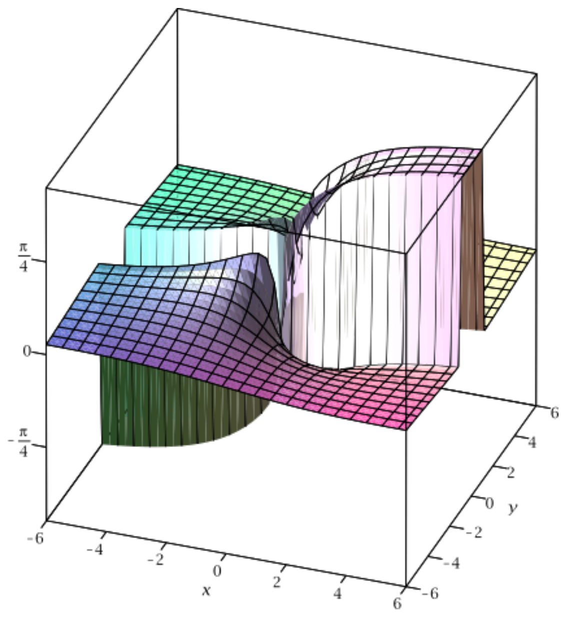
This function has discontinuities in different locations but, as we just indicated, this is not an issue because once a continuous invtangent is defined, it yields the same multisheeted two-dimensional complex manifold (or pair of mulitsheetsed two-dimensional real manifolds constituting the real and imaginary parts). Since the complex invtangent is understood to be a data structure, it can be implemented in various ways and we are free to choose a form of it that leads to very simple jump conditions to facilitate computation. We therefore define the *complex invtangent manifold*[8] for a meromorphic function $w(z) = u(x,y) + i\,v(x,y)$,

$$\operatorname{invtan}(w) = \frac{1}{2}\left(\operatorname{invtan}\left(\frac{u}{1+v}\right) + \operatorname{invtan}\left(\frac{u}{1-v}\right)\right) + \frac{i}{4}\log\left(\frac{(1+v)^2+u^2}{(1-v)^2+u^2}\right). \tag{2.16}$$

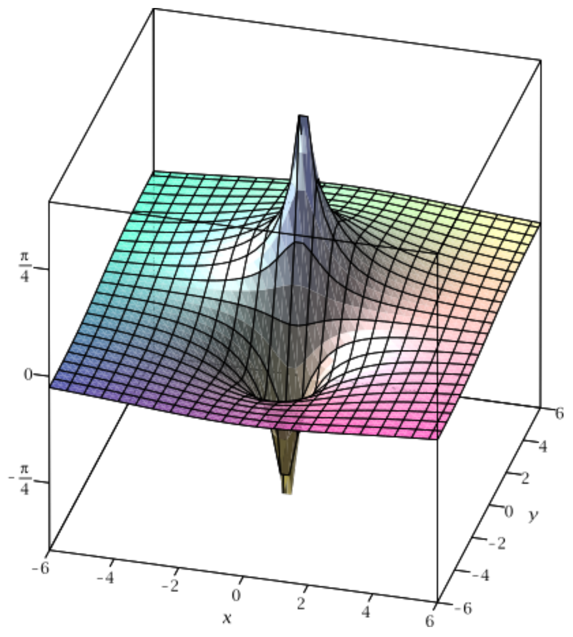
With the arctangent function written as (2.15), we only need to determine the conditions under which the arguments of the real arctangent functions can be singular. It is intuitively evident from figure 2.17a that if the strip of the arctangent defined by $-1 < y < 1$ is raised by $\pi/2$ (for $x < 0$) or lowered by $\pi/2$ (for $x > 0$), like a "vertical lift bridge", then paths moving parallel to the y-axis will pass smoothly across. Moreover, it can be seen intuitively that by moving onto the "vertical lift bridge" we can obtain access to sheets of the two-dimensional manifold that are higher up or lower down, but moving along the slope of the bridge and crossing $x = 0$. We therefore obtain a clear image of the complex invtangent manifold as having infinitely many sheets separated by $\pm\pi$, connected by shifting the interior of the strip $-1 < y < 1$ up and down like a vertical lift bridge to restore continuity.

To compute the complex invtangent manifold defined as (2.16) we need to compute the jump conditions for the radial unwinding numbers needed to patch together branches of $\arctan(z)$ defined as (2.15), which are given by the following

[8]Note that by using expression (2.15) rather than expression (2.14) to define the complex invtangent manifold we have changed the clew branch behaviour, meaning that for $z \in \mathbb{C}$, $\operatorname{invtan} z \neq \arctan z$ for all z. In an implementation of this approach, then, the behaviour as presented to the user should be adjusted to make the clew branch appear to be $\arctan z$, which is straightforward to ensure.



(a) The real part has discontinuities of magnitude $\pi/2$ along the lines $y = \pm 1$.



(b) The imaginary part.

Figure 2.17: Discontinuities of $\frac{1}{2}\left(\arctan\left(\frac{x}{1+y}\right) + \arctan\left(\frac{x}{1-y}\right)\right) + \frac{i}{4}\log\left(\frac{(1+y)^2+x^2}{(1-y)^2+x^2}\right)$ over the complex plane.

Lemma 2.18 (Complex Arctangent) *Let $w(t) = u(t) + i\,v(t)$ be a meromorphic function of a real parameter. Then for some $t^* \in \mathbb{R}$, let $c(t^*) = u(t^*) + i\,v(t^*)$, $p = \operatorname{ord}_c(u)$ and $q = \operatorname{ord}_c(v \pm 1)$. Then with $\operatorname{invtan}(w)$ defined as (2.16), $\Delta\mathcal{K}_r \neq 0$ when $v = \mp 1$ at c, if $p - q$ is odd and $p < q$, in which case*

$$\Delta\mathcal{K}_r = \mp\operatorname{sign}\left(\operatorname{dc}_c(u)\operatorname{dc}_c(1 \pm v)\right).$$

Proof. Since the argument of the logarithm in (2.16) is always real and positive, the logarithmic branch cut is never crossed, so the angular unwinding number is always zero.

If $v = \mp 1$ at c, then with $p = \operatorname{ord}(u)$ and $q = \operatorname{ord}(1 \pm v)$, $u = (x-c)^p(a_0 + \cdots)$ and $v = \mp 1 + (x-c)^q)(b_0 + \cdots)$, so that $1 \pm v = (x-c)^q(\pm b_0 + \cdots)$. Thus, $u/(1 \pm v) = (x-c)^{p-q}(\pm a_0/b_0 + \cdots)$. The result then follows from lemma 2.10. □

The continuous complex arctangent manifold can be defined in a second way using the two-argument arctangent, based on the expression for $\arctan z$ obtained by replacing the one-argument arctangents in (2.15) with their two-argument equivalents. Once we have corrected for spurious discontinuities, we obtain the same invtangent manifold defined alternatively as

$$\operatorname{invtan}(w) = \frac{1}{2}\left(\operatorname{invtan}(u, 1+v) + \operatorname{invtan}(u, 1-v)\right) + \frac{i}{4}\log\left(\frac{(1+v)^2+u^2}{(1-v)^2+u^2}\right). \qquad (2.19)$$

This defines exactly the same manifold, since the unwinding numbers stitch the patches of the two-argument arctangents in such a way as to produce the same result. The benefit of this formulation is mostly conceptual, since for its clew branch automatically has its branch cuts in the same location as the standard definition of the arctangent function in terms of equation (2.14). As such, with the invtan function replaced by arctan it has the same graph over the complex plane as the standard complex arctangent, as shown in figure 2.13.[9] This formulation is problematic from a computational perspective, however, since it gives rise to much more complex jump conditions than specified in lemma 2.18. Since we will rely on the definition of the invtangent manifold as equation (2.16) in the sequel, we omit the analogue of lemma 2.18 for equation (2.19).

The final function we need to consider is the complex logarithm function. As we saw before, the principal branch of the complex logarithm function is given by

$$\log z = \log(x + i\,y) = \frac{1}{2}\log(x^2+y^2) + i\ \arctan(y, x),$$

where we have removed the square-root from the logarithm to avoid the need to compute logarithmic branch cut crossings. The two-argument arctangent is needed here to have the obvious discontinuity appear at the branch cut. Just as for the continuous arctangent function, however, we are free to use the either the one- or two-arctangent functions since they both give rise to the same manifold. Thus, we can define the continuous logarithm manifold, the *invexponential manifold*, either as

$$\operatorname{invexp}(w) \equiv \frac{1}{2}\log(u^2+v^2) + i\ \operatorname{invtan}\left(\frac{v}{u}\right), \qquad (2.20)$$

with the jump conditions for the one-argument arctangent function in lemma 2.10, or as

$$\operatorname{invexp}(w) \equiv \frac{1}{2}\log(u^2+v^2) + i\ \operatorname{invtan}(v, u), \qquad (2.21)$$

with the jump conditions for the two-argument arctangent function in lemma 2.11. For

[9]As a result, this could be presented to the user as the definition of the complex invtangent manifold, even if the manifold is computed behind the scenes using (2.16), provided the front-end clew branch behaviour is designed to match that of (2.19).

either definition, the only non-zero unwinding numbers occur in the imaginary part. The conceptual virtue of the two-argument invtangent definition is that has its branch changes at the same locations the principal branch of the complex logarithm function. As we have seen, however, the two-argument arctangent has much more complex jump conditions to compute. For this reason, analogously to the invtangent manifold, the one-argument invtangent definition will certainly be preferable for implementation in the back-end, designed so that the front-end behaviour matches the two-argument invtangent definition.

2.4 Continuous Real and Complex Integration of Rational Functions

With the ability to restore continuity to logarithms and arctangents of meromorphic functions, we can now consider the computation of continuous expressions of integrals. We restrict our attention here to rational function integrands since it is straightforward to provide computable jump conditions in this case. Using changes of variables this includes a much larger class of functions than simply quotients of polynomials. We consider the computations of integrals of functions $f(z) = p(z)/q(z)$ over a contour $\mathcal{C}$ on the Riemann sphere. If $\mathcal{C} : [0,1] \to \mathsf{C} \equiv \mathbb{C} \cup \{\infty\}$ runs from c to z, and $\varphi(t)$ is a parameterization of the contour $\mathcal{C}$ such that $\varphi(0) = c$ and $\varphi(1) = z$, then the problem is to compute a continuous antiderivative $F(z)$ of $f(z)$ so that

$$\int_{\mathcal{C}} f(w)dw = \int_c^z f(w(\varphi(t)))dt = F(z) - F(c) \tag{2.22}$$

is continuous as a function of z for all z where the integral is continuous. This will be accomplished in the following way.

First, we assume that $p(z), q(z) \in \mathbb{K}[z]$ for some real closed field $\mathbb{K}$.[10] Then, using Hermite reduction, we can compute the rational part of the integral leaving only the transcendental part to find. We can obtain in this way an expression for the antiderivative as

$$\int \frac{p(z)}{q(z)}dz = c_0\frac{p_0(z)}{q_0(z)} + \int \frac{p_r(z)}{q_r(z)}dz,$$

where $p_r(z), q_r(z) \in \mathbb{K}[z]$, $\deg(p_r) < \deg(q_r)$ and $q_r(z)$ is squarefree. Using what Bronstein [1] calls the Lazard-Rioboo-Trager algorithm, the integral can then be expressed

[10]The case where the coefficients lie in $\mathbb{K}(i)$ reduces to the prior case by considering the real and imaginary parts of the rational function separately.

as

$$\int \frac{p(z)}{q(z)}dz = c_0\frac{p_0(z)}{q_0(z)} + \sum_{k=1}^{p} \sum_{t\,|\,U_i(t)=0} t\log(S_i(t,z)),$$

with $U_i(t) \in \mathbb{K}[t]$ and $S_i \in \mathbb{K}[t,z]$. By moving to an algebraic extension $\tilde{\mathbb{K}}$ of $\mathbb{K}$ that adjoins the roots of the polynomials $U_i(t)$, we can then express the integral in the form

$$\int \frac{p(z)}{q(z)}dz = c_0\frac{p_0(z)}{q_0(z)} + \sum_{k=1}^{m} c_k \log\left(p_k(z)\right) + \sum_{k=m+1}^{m+n} 2c_k \arctan\left(\frac{p_k(z)}{q_k(z)}\right),$$

where $c_k \in \tilde{\mathbb{K}}$ and $p_k, q_k \in \tilde{\mathbb{K}}[z]$. The field $\tilde{\mathbb{K}}$ is the minimal algebraic extension of $\mathbb{K}$ needed to express the integral in this form [1]. From this expression, we can then use the angular and radial unwinding numbers to convert the logarithms and arctangents into their continuous versions to obtain a continuous antiderivative

$$F(z) = c_0\frac{p_0(z)}{q_0(z)} + \sum_{k=1}^{m} c_k \,\text{invexp}\left(p_k(z)\right) + \sum_{k=m+1}^{m+n} 2c_k \,\text{invtan}\left(\frac{p_k(z)}{q_k(z)}\right), \tag{2.23}$$

where all of the rational functions in this equation are in reduced form. The continuous invexponential and invtangent manifolds can be computed using the lemmas in the previous section. Since the jump conditions are simpler, the theorems in this section will assume that the the continuous complex manifolds are defined as (2.16) and (2.20).

We first consider real integrals, *i.e.*, where $z = x \in \mathbb{K}$, $p(x), q(x) \in \mathbb{K}[x]$ and $\mathcal{C}$ is just an interval of $\mathbb{R}$. For real integrals the only spurious discontinuities occur within the arctangent function. Thus, for real integrals we focus on the jump conditions for the radial unwinding number.

Theorem 2.24 (Continuous Integrals of Real Rational Functions) *For $\mathbb{K}$ a real closed field, let $p(x)$, $q(x) \in \mathbb{K}[x]$, with $q(x) \neq 0$ on $I = [a,x] \subseteq \mathbb{R}$ and gcd$(p,q) = 1$. Then, the functions (unwinding numbers) $\mathcal{K}_r : I \to \mathbb{Z}$ needed to compute the invtangent manifold required by equation (2.23) are given by the following jump conditions. If x^* is a root of $q_k(x)$ with odd multiplicity m, then $\Delta\mathcal{K}_r \neq 0$ at x^* and*

$$\Delta\mathcal{K}_r\left(\text{invtan}\left(\frac{p_k(x^*)}{q_k(x^*)}\right)\right) = -\text{sign}\left(p_k(x^*)\right)\text{sign}\left(q_k^{(m)}(x^*)\right),$$

where the superscript (m) denotes the m-th derivative.

Proof. Since p_k and q_k are polynomials, they cannot have poles. Thus, by lemma 2.10 the only non-zero unwinding numbers occur for odd order roots of q_k. Let x^* be a root of q_k with odd multiplicity m. Then at x^*, $\mathrm{dc}(p_k) = p_k(x^*)$, since $\gcd(p_k, q_k) = 1$, and $\mathrm{sign}(\mathrm{dc}(q_k)) = \mathrm{sign}(q_k^{(m)}(x^*))$ since if $q_k = (x - x^*)^m(a_0 + a_1(x - x^*) + \cdots)$ then $q_k^{(m)} = m!a_0 + (m+1)!a_1(x - x^*) + \cdots$. The jump condition then follows from an application of lemma 2.10. □

Consider the integral computed by the LRT algorithm and converted to the form (2.23):

$$\int \left(\frac{x^4 - 9x^2 + 14x - 6}{x^6 - 6x^5 + 16x^4 - 20x^3 + 11x^2 - 6x + 5} \right) dx = \arctan\left(\frac{2 - x^2}{(x-1)^3} \right).$$

The expression for the integral has a spurious discontinuity at $x = 1$. With $p_1(x) = 2 - x^2$ and $q_1(x) = (x-1)^3$. Since $\mathrm{sign}(p_1(1)) = 1$ and $\mathrm{sign}(q_1^{(3)}(1)) = 1$, $\Delta\mathcal{K}_r(1) = -1$. Thus, we obtain a continuous expression for the integral in terms of

$$\mathrm{invtan}\left(\frac{2 - x^2}{(x-1)^3} \right) = \arctan\left(\frac{2 - x^2}{(x-1)^3} \right) - \pi \,\mathrm{Heaviside}(x - 1),$$

where $\mathcal{K}_r\left(\mathrm{invtan}\left(\frac{2-x^2}{(x-1)^3}\right)\right) = -\mathrm{Heaviside}(x-1)$. This shows how the radial unwinding number can be expressed in terms of the Heaviside step-function.[11] This also shows how the radial unwinding number is a form of the method of Jeffrey [3], revealing the association of this latter method with unwinding branch point crossings.

Now, although theorem 2.24 illustrates the method in a simple case, it is unlikely to be needed in practice on account of the Rioboo singularity removal method (RSR) for real rational function integrals, not to be confused with the LRT algorithm that also bears Rioboo's name. Rioboo [6] showed how to construct a recursive algorithm based on the extended Euclidean algorithm to convert an arctangent of a rational function to a sum of arctangents of polynomials, such that the input and output of the algorithm have the same derivative. This RSR method does not, however, deal with the case where x can become singular through a change of variable. This is the situation that gives rise to cases like that of equation (2.1) in the introduction. Discontinuities of this kind can also be corrected by the radial unwinding number, as the following theorem shows.

[11] Note that there is no universally accepted value for Heaviside(0). For our purposes it must be chosen to ensure a continuous result. Assuming closure on the right for each branch of the arctangent function, so that $\arctan(\infty) = \pi/2$, we require that $\mathrm{Heaviside}(0) = 0$.

Theorem 2.25 (Continuous Integrals of Real Rational Functions of a Transformed Variable) *Let $f(x)$, $g(x)$ and $t(x)$ be real analytic functions except possibly at known isolated poles on $I = [a,x]$ such that $\int_a^x \frac{f(u)}{g(u)}du = \int_{t(a)}^{t(x)} \frac{p(v)}{q(v)}dv$, where $p(x), q(x) \in \mathbb{K}[x]$, with $\mathbb{K}$ a real closed field. Then, the functions (unwinding numbers) $\mathcal{K}_r : I \to \mathbb{Z}$ needed to compute the invtangent manifold required by equation (2.23) are given by the following jump conditions. For odd order poles x^* of $t(x)$, if deg(p_k) − deg(q_k) is positive and odd, then $\Delta\mathcal{K}_r \neq 0$ at x^* and*

$$\Delta\mathcal{K}_r\left(\text{invtan}\left(\frac{p_k(t(x^*))}{q_k(t(x^*))}\right)\right) = -\text{sign}\left(\text{lc}(p_k)\right)\text{sign}\left(\text{lc}(q_k)\right)\text{sign}\left(\text{dc}(t)\right). \tag{2.26}$$

For x^* such that $q_k(t(x^*)) = 0$, if the multiplicity m of $t^* = t(x^*)$ as a root of q_k is odd, then $\Delta\mathcal{K}_r \neq 0$ at x^*. In this case, if $t^* \neq 0$, then

$$\Delta\mathcal{K}_r\left(\text{invtan}\left(\frac{p_k(t(x^*))}{q_k(t(x^*))}\right)\right) = -\text{sign}\left(p_k(t(x^*))\right)\text{sign}\left(q_k^{(m)}(t(x^*))\right), \tag{2.27}$$

and if x^* is a root of $t(x)$ with odd multiplicity (so that q_k has an odd order root at 0), then

$$\Delta\mathcal{K}_r\left(\text{invtan}\left(\frac{p_k(t(x^*))}{q_k(t(x^*))}\right)\right) = -\text{sign}\left(p_k(t(x^*))\right)\text{sign}\left(q_k^{(m)}(t(x^*))\right)\text{sign}\left(\text{dc}(t)\right). \tag{2.28}$$

Proof. Since the transformation $t(x)$ can have poles, we need to consider two cases. The first case is an x^* such that $t(x^*)$ has a pole of order m. When substituted into a polynomial P, the order of the resulting pole is $m\deg(P)$. The order of $p_k(t(x^*))/q_k(t(x^*))$ is then $-m(\deg(p_k)-\deg(q_k))$. For the result to be a pole, we must have $\deg(p_k) > \deg(q_k)$. Then, the conditions of lemma 2.10 are satisfied if both m and $(\deg(p_k) - \deg(q_k))$ are odd. In this case, $\text{dc}(p_k) = \text{dc}(t)^{\deg(p_k)}\text{lc}(p_k)$ and $\text{dc}(q_k) = \text{dc}(t)^{\deg(q_k)}\text{lc}(q_k)$. The result follows from lemma 2.10 together with the fact that $\text{sign}(\text{dc}(t)) = \text{sign}(\text{dc}(t))^\ell$ for ℓ odd.

The second case is an x^* such that $t^* = t(x^*)$ is a root of $q_k(t(x))$ with multiplicity m. Since $\gcd(p_k, q_k) = 1$ this will not also be a root of $p_k(t(x))$. For an odd order pole we must have m odd. If $t^* \neq 0$ this is the same situation as for theorem 2.24, which determines the jump condition. If $t^* = 0$, however, then x^* is a root of t with multiplicity ℓ. In this case, $\text{ord}(q_k(t(x^*)) = m\ell$, so ℓ must be odd for an odd order pole. Then at x^* $\text{dc}(q_k(t)) = \text{dc}(q_k)\text{dc}(t)^m$, so that at t^* $\text{sign}(\text{dc}(q_k(t))) = \text{sign}(q_k^{(m)}(t(x^*)))\text{sign}(\text{dc}(t))$. □

Again, the RSR algorithm can get rid of the discontinuities resulting from zeros of q, which eliminates conditions (2.27) and (2.28) from consideration. This is a considerable advantage in this case since conditions (2.27) and (2.28) can require transcendental rootfinding. Though it is possible to do this rigorously using an interval analysis method as described by Johnson *et al.* [5], it is best to avoid when possible. Using the RSR method also simplifies condition (2.26) to $\Delta\mathcal{K}_r\left(\operatorname{invtan}\left(p_k(t(x^*))\right)\right) = -\operatorname{sign}\left(\operatorname{lc}(p_k)\operatorname{dc}(t)\right)$.

On the basis of theorem 2.25 we can now treat the discontinuities appearing in (2.1). Rewriting the integral as

$$\int \frac{2dx}{1+3\sin^2 x} = \int \frac{2(\sin^2 x+\cos^2 x)}{4\sin^2 x+\cos^2 x}dx = \int \frac{2}{4\tan^2 x+1}(1+\tan^2 x)dx,$$

we can use the transformation $t = \tan x$, with $dt = (1+\tan^2 x)dx$, to obtain

$$\int \frac{2dx}{1+3\sin^2 x} = \int \frac{2}{4t^2+1}dt = \arctan(2t) = \arctan(2\tan x).$$

Since it is known that the poles of $\tan x$ lie at $\frac{(2n+1)\pi}{2}$, $n \in \mathbb{Z}$, we need to compute the jump conditions for these poles. For this we need only consider one pole, and the rest are similar by symmetry. Considering $x = \frac{\pi}{2}$ we can find that $\operatorname{sign}\left(\operatorname{dc}_{\pi/2}(\tan x)\right) = -1$ by expanding $\tan x$ in a series at $x = \frac{\pi}{2}$, or simply by observing that $\tan x \to +\infty$ as $x \to \frac{\pi}{2}^-$. Thus, $\operatorname{sign}\left(\operatorname{dc}_{(2n+1)\frac{\pi}{2}}(\tan x)\right) = -1$. Thus, since $\operatorname{lc}(p_1) = 2$, we can express the continuous integral as

$$\operatorname{invtan}(2\tan x) = \arctan(2\tan x) + \pi \sum_{k=-\infty}^{\infty} \operatorname{Heaviside}\left(x - \frac{(2k+1)\pi}{2}\right).$$

Theorem 2.25 opens up our computational strategy for computing continuous integrals to a much wider range of integrands. In particular, using the Weierstrass substitution $t = \tan(x/2)$ it is possible to convert any rational combination of trigonometric functions into a rational function of t. Thus, a continuous antiderivative can be computed for any rational trigonometric integral using our method.

As another example, consider the function $\frac{\Psi(x)\Gamma(x)}{1+\Gamma(x)^2}$, where $\Psi(x)$ is the digamma function, which satisfies $d\Gamma = \Psi\Gamma$. Using the substitution $t = \Gamma(x)$, we obtain

$$\int \frac{\Psi(x)\Gamma(x)}{1+\Gamma(x)^2}dx = \int \frac{dt}{1+t^2} = \operatorname{invtan}(\Gamma(x)).$$

The unwinding numbers needed to compute the invtangent manifold, correcting the discontinuities that would appear in $\arctan(\Gamma(x))$ at $x = -n, n \in \mathbb{N}$, can be computed using the pole conditions specified in theorem 2.25. Since all the poles of $\Gamma(z)$ are simple and

$\mathrm{Res}_{z=-k}\Gamma(z) = \frac{(-1)^k}{k!}$, $\mathrm{sign}(\mathrm{dc}_{-k}(\Gamma(x)) = (-1)^k$. Since $p_1(t) = t$ in this case, we find that

$$\int \frac{\Psi(x)\Gamma(x)}{1+\Gamma(x)^2}dx = \mathrm{invtan}(\Gamma(x)) = \arctan(\Gamma(x)) - \pi\sum_{k=0}^{\infty}(-1)^k \,\mathrm{Heaviside}(x+k).$$

We remark that according to the definition of derivative from differential algebra, the derivative of a piecewise constant function is zero. As such, the Heaviside step functions needed to render integrals continuous differentiate to zero, and are thus invisible to the Risch algorithm and its variants. Moving to the theory of distributions, it is possible to define the Heaviside step function as an integral of the delta function, *i.e.*,

$$\mathrm{Heaviside}(x-a) = \int_{-\infty}^{x} \delta(t-a)dt,$$

so that

$$\frac{d}{dx}\mathrm{Heaviside}(x-a) = \delta(x-a),$$

reiterating the need to consider the source of the integration problem in analysis in order to adequately remove spurious discontinuities.

Our unwinding number method can also be used to compute continuous contour integrals of complex functions. The usual method for computing contour integrals in terms of a parameterization of the contour is to convert the contour integral into a pair of real integrals in terms of the parameter, meaning that if $f(z)$ is holomorphic along the parameterized contour $\mathcal{C} : I \to \mathcal{C}$, where I is the unit interval, and $\mathcal{C}(t) = z(t)$, then

$$\int_{\mathcal{C}} f(z)dz = \int_0^1 f(z)z'dt,$$

and this integral splits into a pair of real integrals, the real and imagainary parts of the contour integral. From a computational standpoint, however, it is preferable to compute an antiderivative of f and use this to evaluate the contour integral. The way the antiderivative of a complex function is usually defined, however, requires f to be holomorphic. Yet the validity of the equation

$$\int_{\mathcal{C}} f(z)dz = F(z) - F(c), \tag{2.29}$$

where $F'(z) = f(z)$ and the endpoints of the contour $\mathcal{C}$ are c and z, requires only that $F(z)$ is a (complex) differentiable function for all z on $\mathcal{C}$, which is the case provided that f is holomorphic *along the contour*.

Now, the problem usually identified with this idea is that for a simple closed curve, the endpoints of the contour are the same, $z = c$, so that

$$\oint_{\mathcal{C}} f(z)dz = F(z) - F(c) = 0,$$

which can only happen for holomorphic functions. And this would indeed be so were it the case that all complex functions were single valued, but they are not. Accordingly, we can consider a simple closed curve in the domain of the function F, but as a result of changing branches of $F(z)$ it need not be the case that $F(z) = F(c)$. Indeed, if the interior of $\mathcal{C}$ contains a pole of $f(z)$ at d, then $F(z)$ will contain an invexponential term corresponding to the pole and the path will cross the logarithmic branch cut (for the two-argument arctangent definition of invexp). Since we must take this into consideration to keep $F(z)$ differentiable hence continuous (since the f is assumed to be holomorphic along the contour), then the integral picks up a factor of $\pm 2\pi i \operatorname{Res}(f, d)$ to ensure continuity, since $\operatorname{Res}(f, d)$ will be the coefficient of the invexponential term. Accordingly, for a counterclockwise turn around the pole we find in such a case that

$$\oint_{\mathcal{C}} f(z)dz = F(z) - F(c) = 2\pi i \operatorname{Res}(f, d),$$

which is precisely the expression of the residue theorem in this case.

Thus, the use of unwinding numbers to restore continuity, allows the definition of antiderivative to be extended to meromorphic functions. There is, of course, no contradiction implied here. We preserve the usual formulations of standard theorems (*i.e.*, residue theorem, Cauchy-Goursat, *etc.*) provided the contour is defined in the domain of the antiderivative. We recover the usual result about holomorphic functions over closed contours if we require the closure of the path in the image of F, since then we really do return to the starting point of the curve. This can only happen for contractible curves in the codomain of F. Regarded in terms of the image path of F, then, the "closed curve" around a pole is actually an open curve on a multisheeted surface. Therefore, the antiderivative of a meromorphic function f is well-defined provided that the contour does not encounter any poles of f,[12] and we have established the following

[12]We could incorporate this case as well by considering functions to always be defined over C wherever possible, but we do not explore this possibility here. In any event, such poles are boundary points to where the integral is finite, defining boundaries of connected components of the integral in the usual sense.

Theorem 2.30 (Antiderivative of a Meromorphic Function) *Let $f(z)$ be a meromorphic function, $F(z)$ satisfy $F'(z) = f(z)$ and the expression of $F(z)$ be corrected for continuity. Then for a contour $\mathcal{C}$ from c to z in $\mathbb{C}$ such that $f(z)$ is analytic along $\mathcal{C}$,*

$$\int_{\mathcal{C}} f(z) = F(z) - F(c),$$

i.e., *$F(z)$ is a complex antiderivative of $f(z)$.*

Some comment is in order here concerning the view of complex functions implied. For a meromorphic function $f(z) = u(x,y) + i\,v(x,y)$ that has been corrected for continuity, in the manner we have considered throughout the paper, the functions u and v define 2 dimensional manifolds parameterized by x and y. Insofar as each branch of a complex function is single valued, and can be stitched together using unwinding numbers, then it is possible to move around on these manifolds without encountering any source of non-analyticity, except at isolated poles or locations where the function ceases to be meromorphic. Thus, it is analogous to treating complex functions in terms of Riemann surfaces, but rather than extending the domain of the function into a one dimensional complex manifold, we extend the codomain of the function into a two-dimensional complex manifold, or pair of two-dimensional real manifolds as we consider here. As such, given a path in $\mathbb{C}$ over which the function is holomorphic we obtain a *bona fide* function from the path in $\mathbb{C}$ to its image in the codomain manifold.

The only trick with the application of theorem 2.30, then, is that one cannot simply evaluate the integral by substitution of the value of z and c into the algebraic antiderivative F. $F(z)$ must be analytic along the entire contour, which requires that the intersection of the contour and any branch crossings must be computed to obtain the correct value of $F(z)$.

With this in mind, we now consider the jump conditions to ensure continuity for contour integrals of complex rational functions. We restrict attention to only the jump conditions for the invexponential, leaving the formulation and proof of conditions for the invtangent, based on lemma 2.18 as an exercise for the reader.

Theorem 2.31 (Continuous Contour Integrals of Rational Functions) *Let $p(z), q(z) \in \mathbb{C}[z]$, with $q(x) \neq 0$ on an analytic simple contour $\mathcal{C} : I \to \mathbb{C}$, where I is the unit interval and* $\gcd(p,q) = 1$, *and let $\mathcal{C}(1) = z$. Let $\varphi(t) + i\,\psi(t)$ be a parameterization of $\mathcal{C}$. Then, the functions (unwinding numbers) $\mathcal{K}_r : I \to \mathbb{Z}$ needed to compute the invexponential manifold required by equation (2.23) are given by the following jump conditions: Let*

$p_k(\varphi(t) + i\psi(t)) = u_k(t) + i\, v_k(t)$, then $\Delta\mathcal{K}_r \neq 0$ at t^* if (a) $\mathcal{C}(t^*) \neq 0$ and t^* is a real root of u_k with odd multiplicity m, in which case

$$\Delta\mathcal{K}_r\left(\operatorname{invexp}\left(u_k(t^*) + i\, v_k(t^*)\right)\right) = -\operatorname{sign}\left(v_k(t^*)\right)\operatorname{sign}\left(u_k^{(m)}(t^*)\right),$$

or (b) $\mathcal{C}(t^*) = 0$ with t^* a real root of both v_k and u_k with respective multiplicities multiplicity m_1 and m_2, and $m_1 < m_2$ with $m_1 - m_2$ odd, in which case

$$\Delta\mathcal{K}_r\left(\operatorname{invexp}\left(u_k(t^*) + i\, v_k(t^*)\right)\right) = -\operatorname{sign}\left(v_k^{(m_1)}(t^*)\right)\operatorname{sign}\left(u_k^{(m_2)}(t^*)\right).$$

Proof. The jump conditions for the complex invexponential defined as (2.20) are determined by lemma 2.10. The criterion for a jump is met when $v_k(t)/u_k(t)$ has an odd order pole. This can happen in one of two ways. The first is if t^* is a root of $u_k(t)$ while not also a root of $v_k(t)$, and the multiplicity m of t^* is odd. Since $u_k^{(m)}(t^*) = m!\,\mathrm{dc}_{t^*}(u_k(t))$, $\Delta\mathcal{K}_r = -\operatorname{sign}(v_k(t^*))\operatorname{sign}\left(u_k^{(m)}(t^*)\right)$. The second is if t^* is a root of both $v_k(t)$ and $u_k(t)$ with respective multiplicities m_1 and m_2 with $m_1 < m_2$ and $m_1 - m_2$ odd. The jump condition follows from a similar observation about $\mathrm{dc}_{t^*}(v_k(t))$ and $\mathrm{dc}_{t^*}(u_k(t))$. □

As an example, consider the contour integral of $f(z) = \frac{3z^2+1}{z(z^2+1)}$, which has poles of order 1 at $z = 0, \pm i$. Suppose we take our contour to be the circle of radius 2 around the origin in the complex plane, integrating in a counterclockwise direction. Using the algcurves package in Maple we can compute the parameterization

$$z(t) = \varphi(t) + i\,\psi(t) = \frac{-t^2 + 2t + 1}{t^2 + 1} + i\,\frac{t^2 + 2t - 1}{t^2 + 1}. \tag{2.32}$$

This paramaterization has domain $\mathbb{R}$, but we can scale it to $[0, 1]$ by taking $t(s) = \tan\left(\pi s - \frac{\pi}{2}\right)$. We can easily compute the algebraic antiderivative of $f(z)$ to give

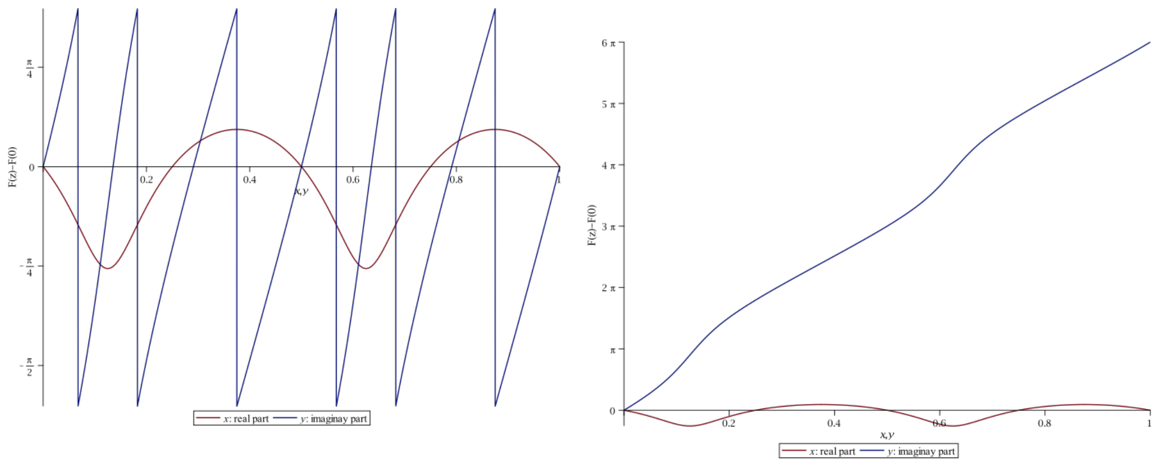
$$\int \frac{3z^2 + 1}{z(z^2 + 1)} dz = \log(z(z^2 + 1)).$$

Substituting the parameterization yields

$$\int_{\mathcal{C}} \frac{3z^2 + 1}{z(z^2 + 1)} dz = \log(u_1(t) + i\, v_1(t))$$

with

$$u_1(t) = \frac{3t^6 + 10t^5 - 29t^4 - 44t^3 + 29t^2 + 10t - 3}{(t^2 + 1)^3}$$



(a) (b)

Figure 2.34: Contour integral of $f(z) = \frac{3z^2+1}{z(z^2+1)}$ (a) before and (b) after being corrected for continuity.

and

$$v_1(t) = \frac{t^6 - 14t^5 - 31t^4 + 36t^3 + 31t^2 - 14t - 1}{(t^2+1)^3}.$$

It can be verified that u_1 and v_1 have no common roots by computing their gcd. We find that u_1 has six real roots,

$$t_{1,2} = -\frac{2}{3} \pm \frac{\sqrt{7}}{2}, \quad t_{3,4} = -\frac{2}{3} \pm \frac{\sqrt{7}}{2}, \quad t_{5,6} = 1 \pm \sqrt{2},$$

leading to six jump conditions for the radial unwinding number. Computing the derivative of u_1 and evaluating v_1 and u_1' at each of these points, we find that $\Delta\mathcal{K}_r = 1$ for all the roots of u_1. Thus, abbreviating the Heaviside step function as $H(x)$, we can express the continuous contour integral as

$$F(z) = \int_C \frac{3z^2+1}{z(z^2+1)} dz = \operatorname{invexp}(z(t)(z(t)^2+1)) = \log(z(t)(z(t)^2+1)) + \pi \sum_{k=1}^{6} H(t - t_k), \tag{2.33}$$

with $z(t)$ given by (2.32).

We can now evaluate the integral at any point on the contour simply by evaluating the expression for $F(z)$ in (2.33) at any value of t and subtracting $F(0)$. Before computing the unwinding numbers, the plot of $\log(z(z^2+1))$ produces the graph appearing in figure 2.34a, where we have used the transformation $t(s) = \tan\left(\pi s - \frac{\pi}{2}\right)$ to take the domain onto $[0, 1]$. Once we correct for continuity, we obtain the graph appearing in figure 2.34b. From the graph it appears that the difference between $F(z(t(1))) - F(z(t(0)))$ is

$6\pi i$. And, indeed, when we take $\lim_{t\to 1^-} F(z(t)) - \lim_{t\to 0^+} F(z(t))$ we find exactly $6\pi i$ so that this method furnishes a rigorous proof that

$$\oint_C \frac{3z^2+1}{z(z^2+1)}dz = 6\pi i,$$

in agreement with the residue theorem, since the contour we chose encloses the three poles of $f(z)$, each having residue 1.

Of course we would never use this method to compute closed-contour integrals for functions with known poles. An inspection of theorem 2.31 reveals, however, that the jump conditions will apply for any cases where the integral involves a terms of the form $\mathrm{invexp}(\xi(z))$ for any meromorphic function $\xi(z)$ that is analytic along the contour. Similarly, the extension of the theorem to the arctangent terms of the form $\mathrm{invtan}(\xi(z))$ will apply for any cases where $\xi(z)$ is analytic along the contour. These results amount to a combination of theorem 2.31 with theorem 2.25. Thus, for functions where the poles are not known, this method could be useful for the algebraic detection of poles.

Since there is nothing about this contour integration method that is specific to complex functions, it can also be applied to line integrals of functions over any rational algebraic curve, *i.e.*, algebraic curve of genus zero, since such curves can be rationally parameterized. With the aforementioned extension of theorem 2.31 to transformed variables, we can compute contour integrals over more general functions. Adapting this latter result to the line integral case then allows any line integral that can be transformed to a rational function to be computed with the tools developed here, provided the path is a rational algebraic curve.

2.5 Conclusion

We have considered a solution strategy to the problem of obtaining expressions of integrals on domains of maximum extent. We showed how a pair of unwinding numbers can be employed to eliminate spurious discontinuities in the logarithm and arctangent functions, which are fundamental functions in the integration problem as a result of Liouville's theorem. We provided general conditions for the appearance of discontinuities in these functions and the resulting values of the unwinding numbers needed to restore continuity. We then illustrated the computability of the unwinding numbers for the integration of rational functions.

There are a number of related ways in which this approach stands to be generalized.

First of all, to be fully general, the jump conditions need to be analyzed in terms of Puiseux series rather than Laurent series. This will allow the consideration of continuity preservation at the branch points of algebraic functions. The approach as it exists now can be applied for any meromorphic functions with a Laurent expansion at the branch points, but this fails to be the case for the branch points of general algebraic functions. The method also needs to be extended to work with algorithms for the integration of algebraic and transcendental functions. The ability to work with Puiseux series and to consider integration of algebraic functions will also allow line and contour integration over algebraic curves of positive genus. Extending the methods to algebraic and transcendental function integration will require the consideration of algorithms for the unwinding numbers that can appear in the integrands and integrals for those symbolic integration methods. This will thus require a general method to compute unwinding numbers for a class of functions. If these interrelated approaches can be accomplished with reasonable algorithmic efficiency, then this approach could lead to a general solution to the problem of obtaining expressions for real, line and contour integrals that are continuous on domains of maximum extent.

2.6 Bibliography

[1] Manuel Bronstein. *Symbolic integration I: Transcendental Functions*, volume 1. Springer, 2005.

[2] Robert M Corless and David J Jeffrey. The unwinding number. *ACM SIGSAM Bulletin*, 30(2):28–35, 1996.

[3] David J Jeffrey. Integration to obtain expressions valid on domains of maximum extent. In *Proceedings of the 1993 international symposium on Symbolic and algebraic computation*, pages 34–41. ACM, 1993.

[4] David J Jeffrey and Albert D Rich. The evaluation of trigonometric integrals avoiding spurious discontinuities. *ACM Transactions on Mathematical Software (TOMS)*, 20(1):124–135, 1994.

[5] Tomas Johnson and Warwick Tucker. Enclosing all zeros of an analytic functiona rigorous approach. *Journal of Computational and Applied Mathematics*, 228(1):418–423, 2009.

[6] Renaud Rioboo. Quelques aspects du calcul exact avec les nombres réels. *Thèse de Doctorat de l'Université de Paris 6, Informatique*, 1991.

[7] Walter Rudin. *Principles of Mathematical Analysis (International Series in Pure & Applied Mathematics).* McGraw-Hill Publishing Co., 1976.

CHAPTER 3

Symbolic-Numeric Integration of Rational Functions

Symbolic-Numeric Integration of Rational Functions[1]

3.1 Introduction

In this chapter we consider two algorithms for the approximate symbolic integration of univariate rational functions in $\mathbb{Q}(x)$ using a combination of symbolic and numerical methods. We provide in section 3.4 a forward and backward error analysis of the symbolic result, showing that the algorithms are forward and backward stable in a structured sense and that both the forward and backward error are proportional to a user-supplied tolerance. Both algorithms have been implemented in the open-source *Basic Polynomial Algebra Subprograms*, or BPAS, package (`http://bpaslib.org`), which is discussed in section 3.5. The results of experiments on the implementations are discussed in section 3.6. Although one of the algorithms emerges as stronger overall, both algorithms are seen to have advantages in different symbolic computing contexts.

3.1.1 Symbolic-Numeric integration of Rational Functions

Hybrid symbolic-numeric integration of rational functions is interesting for several reasons. First, a formula, not a number or a computer program or subroutine, may be desired, perhaps for further analysis, such as by taking asymptotics. In this case one typically wants an exact symbolic answer, and for rational functions this is in principle always possible. However, an exact symbolic answer may be too cluttered with algebraic numbers or lengthy rational numbers to be intelligible or easily analyzed by further symbolic manipulation. See, *e.g.*, figure 3.1. Discussing symbolic integration, Kahan [7] in his typically dry way gives an example "atypically modest, out of consideration for the typesetter", and elsewhere has rhetorically wondered: "Have you ever used a computer algebra system, and then said to yourself as screensful of answer went by, "I wish I hadn't asked." " Fateman has addressed rational integration [5], as have Noda and Miyahiro [8, 9], for this and other reasons.

Second, there is interest due to the potential to carry symbolic-numeric methods for rational functions forward to transcendental integration, since the rational function algorithm is at the core of more advanced algorithms for symbolic integration. Particularly in the context of *exact* input, which we assume, it can be desirable to have an intelligible

[1]A version of this paper will be submitted for publication in an appropriate refereed journal.

$$\begin{aligned}
&-\frac{19}{84}\ln\left(x^2+\sqrt{2\sqrt{7}-5}\,x+\sqrt{7}\right)\sqrt{2\sqrt{7}-5}\,\sqrt{7}-\frac{7}{12}\ln\left(x^2+\sqrt{2\sqrt{7}-5}\,x+\sqrt{7}\right)\sqrt{2\sqrt{7}-5}\\
&+\frac{19}{42}\,\frac{\arctan\left(\frac{2\,x+\sqrt{2\sqrt{7}-5}}{\sqrt{2\sqrt{7}+5}}\right)\left(2\sqrt{7}-5\right)\sqrt{7}}{\sqrt{2\sqrt{7}+5}}+\frac{7}{6}\,\frac{\arctan\left(\frac{2\,x+\sqrt{2\sqrt{7}-5}}{\sqrt{2\sqrt{7}+5}}\right)\left(2\sqrt{7}-5\right)}{\sqrt{2\sqrt{7}+5}}\\
&-\frac{1}{7}\,\frac{\arctan\left(\frac{2\,x+\sqrt{2\sqrt{7}-5}}{\sqrt{2\sqrt{7}+5}}\right)\sqrt{7}}{\sqrt{2\sqrt{7}+5}}+\frac{19}{84}\ln\left(x^2-\sqrt{2\sqrt{7}-5}\,x+\sqrt{7}\right)\sqrt{2\sqrt{7}-5}\,\sqrt{7}+\frac{7}{12}\ln\Big(x^2\\
&-\sqrt{2\sqrt{7}-5}\,x+\sqrt{7}\Big)\sqrt{2\sqrt{7}-5}+\frac{19}{42}\,\frac{\arctan\left(\frac{2\,x-\sqrt{2\sqrt{7}-5}}{\sqrt{2\sqrt{7}+5}}\right)\left(2\sqrt{7}-5\right)\sqrt{7}}{\sqrt{2\sqrt{7}+5}}\\
&+\frac{7}{6}\,\frac{\arctan\left(\frac{2\,x-\sqrt{2\sqrt{7}-5}}{\sqrt{2\sqrt{7}+5}}\right)\left(2\sqrt{7}-5\right)}{\sqrt{2\sqrt{7}+5}}-\frac{1}{7}\,\frac{\arctan\left(\frac{2\,x-\sqrt{2\sqrt{7}-5}}{\sqrt{2\sqrt{7}+5}}\right)\sqrt{7}}{\sqrt{2\sqrt{7}+5}}
\end{aligned}$$

Figure 3.1: Maple output for the integral $\int \frac{x^2-1}{x^4+5x^2+7}dx$.

approximate expression for an integral while retaining the exact expression of the integral for subsequent symbolic computation. The ability to do this is a feature of one of our algorithms that alternative approaches, particularly those based on partial fraction decomposition, do not share.

Besides intelligibility and retention of exact results, one might be concerned with numerical stability, or perhaps efficiency of evaluation. We consider stability issues in Sections 3.4 and 3.6. We remark that the algorithm we present here has quite superior numerical stability in many cases, and has good structured backward error and highly accurate answers, while providing the more intelligible answers we desire.

We emphasize that the goal of these algorithms is not to produce numerical values of definite integrals of rational functions, although it can be used for such. The goal is to produce an intelligible formula for the antiderivative which is correct in an approximate sense: the derivative of the answer produced will be another rational function near to the integrand, and, importantly, of the same form in that the denominator will have the correct degrees of its factors in its squarefree factorization and the residues in its partial fraction decomposition will also have the correct multiplicity.[2]

As indicated above, the combination of symbolic and numerical methods in the integration of rational functions is not new. Noda and Miyahiro [8, 9] developed a symbolic-numeric, or *hybrid*, method to integrate rational functions based on the use of the ap-

[2]Note that strict preservation of the form of the integrand is not quite achieved for the PFD method described below, since the derivative cannot be simplified into this form without using approximate gcd. Thus, with exact computation, the degree of the numerator and denominator of the nearby integrand is larger in general than the exact integrand.

proximate symbolic algorithms for noisy data, numerical rootfinding and exact symbolic integration methods. Fateman [5] advocates a simpler hybrid approach, largely to produce a fast method that makes symbolic results more useful and more palatable, avoiding the "surd" or "RootOf" notation in the results of symbolic integrations. Both approaches work with the assumption that the input rational function has floating point coefficients.

For the existing symbolic-numeric algorithms for rational function integration, the approach is to be as sparing as possible in the use of symbolic algorithms to minimize their expense, in particular given that floating point input is assumed to be imprecise. In contrast, given that our working assumption is that the input rational function is *exact*, the present chapter is dealing with a somewhat different problem, *viz.*, the approach involves the injection of numerical methods into an otherwise purely symbolic context. As was mentioned above, the reasons such an approach is desirable include intelligibility, retention of exact results and stable or efficient evaluation.Since it is accuracy, speed and stability that matter in the context of scientific computing, a symbolic package that provides a suitable balance of these desiderata in a way that can be merged seamlessly with other scientific computations, as our implementation provides, has considerable advantages over CAS style symbolic computation with exact roots.

The usual approach to symbolic integration here begins with a rational function $f(x) = A(x)/B(x) \in \mathbb{Q}(x)$, with $\deg(A) < \deg(B)$ (ignoring any polynomial part, which can be integrated trivially) and computes an integral in two stages:

- rational part: computes a rational function C/D such that

$$\int f(x)\,dx = \frac{C(x)}{D(x)} + \int \frac{G(x)}{H(x)}dx, \tag{3.2}$$

  where the integral on the right hand side evaluates to a transcendental function (log and arctan terms);

- transcendental part: computes the second (integral) term of the expression (3.2) above yielding, after post-processing,

$$\int f(x)dx = \frac{C(x)}{D(x)} + \sum v_i \log(V_i(x)) + \sum w_j \arctan(W_j(x)), \tag{3.3}$$

  $V_i, W_j \in \mathbb{K}[x]$, with $\mathbb{K}$ being some algebraic extension of $\mathbb{Q}$.

In symbolic-numeric algorithms for this process some steps are replaced by numeric or quasi-symbolic methods. Noda and Miyahiro use an approximate Horowitz method (involving approximate squarefree factorization) to compute the rational part and either the Rothstein-Trager (RT) algorithm or (linear/quadratic) partial fraction decomposition (PFD) for the transcendental part (see Section 3.2 for a brief review of these algorithms). The algorithm considered by Fateman avoids the two stage process and proceeds by numerical rootfinding of the denominator $B(x)$ (with multiplicities) and PFD to compute both rational and transcendental parts. In both cases, the working assumption is that the input uses floating point numbers that are subject to uncertainty or noise and the numerical algorithms use double precision.

Part of the power of symbolic algorithms is their ability to preserve structural features of a problem that may be very difficult to preserve numerically. Particularly given our focus on exact input, we are interested in preserving as much structure of the problem as possible if we are to resort to the use of numerical methods for rootfinding. Our implementation relies on the sophisticated rootfinding package MPSolve (`http://numpi.dm.unipi.it/mpsolve`), which provides *a posteriori* guaranteed bounds on the relative error of all the roots for a user-specified tolerance ε. To balance efficiency of computation and structure-preservation, we use more efficient symbolic algorithms where possible, such as the Hermite method for the rational part, and consider two methods of computing the transcendental part, one that computes the exact integral using the Lazard-Rioboo-Trager (LRT) method followed by numerical approximation, and the other that uses a multiprecision PFD method to compute a nearby integrand that splits over $\mathbb{Q}$ and then performs a structured integration. For more details on the symbolic algorithms, see Section 3.2. The symbolic-numeric algorithms are discussed in Section 3.3.

The advantage of combining multiprecision numerical software with symbolic algorithms is that it allows the user to specify a tolerance on the error of the symbolic-numeric computation. This, together with *structured* backward and forward error analysis of the algorithm, then allows the result to be interpreted in a manner familiar to users of numerical software but with additional guarantees on structure-preservation. We provide such an analysis of the structured error of our algorithm in Section 3.4.

An interesting feature of the backward stability of the algorithm is that it follows that the computed integral can be regarded as the *exact* integral of a slightly perturbed input integral, and, as stated previously, of the correct form (modulo the possible need for an approximate gcd). Insofar as the input rational function is the result of model construction or an application of approximation theory it is already subject to error even though its coefficients are not floats. Thus, the input, though formally exact, is nevertheless still an approximation of a system or problem it represents. Assuming the model approximation error is small, this means that the rational function that best approximates the system or problem represented by the input is some nearby rational function in a small neighbourhood of $f(x)$, for example in the sense of the space determined by variation of the coefficients of f or in the sense determined by an integral norm, as we consider below. The backward stability therefore shows that the integral actually computed is also a nearby rational function within another small neighbourhood, for which we have some control over its size. In a manner similar to numerical analysis, then, by an appropriate choice of tolerance, we can ensure that the latter neighbourhood is smaller than the former, so that the numerical perturbation of the problem is smaller than the model approximation error. The upshot of this is that the use of backward error analysis shows how a symbolic-numeric algorithm can be compatible with the *spirit* of uncertain input, even if the input has non-float coefficients. That is, we are assuming that the modeling procedure got a rational function with the same structure as an exact model, even if its data is uncertain in other ways.

This shows how a backward error analysis can be useful even in the context of exact integration. In the general case of exact input, however, a backward error analysis alone is not enough. This is why we provide a forward error analysis, to provide *a posteriori* assurance of a small forward error sufficiently far away from singularities in the integrand. We also provide such an analysis in Section 3.4.

Our algorithm may be adapted to take floating point input by using additional symbolic-numeric routines, such as approximate GCD and approximate squarefree factorization, in order to detect nearby problems with increased structure. Such an approach would shift the problem onto the *pejorative manifold*, *i.e.*, the nearest most singular problem. This protects against ill-conditioning of the problem on account of the fact that ill-conditioning for roots of polynomials is the result of the ability of small perturbations leading to changes in multiplicities [6]. The symbolic-numeric structured PFD integration we propose already uses this approach for the transcendental part of the integral. The structured error analysis of our algorithms entails that the problem stays on remains on the pejorative manifold after the computation of the integral. Since there have been considerable advances in algorithms for approximate polynomial algebra since the time of writing of [9], such as the ApaTools package of Zeng [12], the combination of error control and singular problem detection could yield a considerable advance over the early approach of Noda and Miyahiro.

3.2 Methods for Exact Integration of Rational Functions

We begin by reviewing symbolic methods for integrating rational functions.[3] Let $f \in \mathbb{R}(x)$ be a rational function over $\mathbb{R}$ with a denominator of positive degree. There exist polynomials $P, A, B \in \mathbb{R}[x]$ such that we have $f = P + A/B$ with $\gcd(A, B) = 1$ and $\deg(A) < \deg(B)$. Since P is integrated trivially, we ignore the general case and assume that $f = A/B$ with $\deg(A) < \deg(B)$. Furthermore, thanks to Hermite reduction, one can extract the rational part of the integral, leaving a rational function G/H, with $\deg(G) < \deg(H)$ and H squarefree, remaining to integrate. For the remainder of this section, then, we will assume that the function to integrate is given in the form G/H, with $\deg(G) < \deg(H)$ and H squarefree.

Partial-fraction decomposition (PFD) algorithm. The partial fraction decomposition algorithm for rational functions in $\mathbb{R}(x)$ can be presented in different ways, depending on whether one admits complex numbers in expressions. We present a method based upon a complete factorization of the denominator over $\mathbb{C}$, followed by its conversion into an expression containing only constants from $\mathbb{R}$.

Consider the splitting of H expressed in the form

$$H = p\prod_{i=1}^{n}(x-\alpha_i)\prod_{j=n+1}^{n+m}\left[(x-(\alpha_j+i\,\beta_j))(x-(\alpha_j-i\,\beta_j))\right],$$

separating real roots from complex conjugate pairs, where $p, \alpha_k, \beta_k \in \mathbb{R}$. Then there exist a_k and b_k such that

$$\frac{G}{H} = \sum_{i=1}^{n}\frac{a_i}{x-\alpha_i} + \sum_{j=n+1}^{n+m}\left[\frac{a_j+i\,b_j}{(x-(\alpha_j+i\,\beta_j))} + \frac{a_j-i\,b_j}{(x-(\alpha_j-i\,\beta_j))}\right]. \tag{3.4}$$

The numerator quantities $c_k = a_k + i\,b_k$ corresponding to the roots $\gamma_k = \alpha_k + i\,\beta_k$ we call *residues* by analogy to complex analysis. Note that in the case here where H is squarefree, the residues can be computed by the formula $c_k = c(\gamma_k) = G(\gamma_k)/H'(\gamma_k)$.

The real root terms are easily integrated to yield terms of the form $a_i\log(x-\alpha_i)$. Extracting terms of the form

$$a_j\left[(x-(\alpha_j+i\,\beta_j))^{-1} + (x-(\alpha_j-i\,\beta_j))^{-1}\right]$$

[3]The following review is based in part on the ISSAC 1998 tutorial [3] and the landmark text book [2] of M. Bronstein.

from (3.4) we obtain pairs of complex log terms that can be combined to form a single real log term of the form $a_j \log(x^2 - 2\alpha_j x + \alpha_j^2 + \beta_j^2)$. Extracting terms of the form

$$i\, b_j \left[(x - (\alpha_j + i\, \beta_j))^{-1} - (x - (\alpha_j - i\, \beta_j))^{-1} \right]$$

from (3.4) and making use of the observation of Rioboo that

$$\frac{d}{dx} i \log\left(\frac{X + iY}{X - iY}\right) = \frac{d}{dx} 2 \arctan(X/Y), \tag{3.5}$$

for $X, Y \in \mathbb{R}[x]$ (see [2], pp. 59*ff.*), we obtain a term of the form $2b_j \arctan\left(\frac{\alpha_j - x}{\beta_j}\right)$.

Where there are repeated residues in the PFD it is possible to combine terms of the integral together. The combination of logarithms with common a_k simply requires computing the product of their arguments. For the arctangent terms the combination of terms with common b_k can be accomplished by iterated application of the rule

$$\arctan\left(\frac{X}{Y}\right) + \arctan\left(\frac{\alpha - x}{\beta}\right) \to \arctan\left(\frac{X(\alpha - x) - \beta Y}{Y(\alpha - x) + \beta X}\right), \tag{3.6}$$

which is based on the fact that $\log(X + i\, Y) + \log((\alpha - x) + i\, \beta) = \log((X(\alpha - x) - \beta Y) + i\,(Y(\alpha - x) + \beta X))$ and equation (3.5).

A major computational bottleneck of the symbolic algorithms based on a PFD is the necessity of factoring polynomials into irreducibles over $\mathbb{R}$ or $\mathbb{C}$ (and not just over $\mathbb{Q}$) thereby introducing algebraic numbers even if the integrand and its integral are both in $\mathbb{Q}(x)$. Unfortunately, introducing algebraic numbers may be necessary: any field containing an integral of $1/(x^2 + 2)$ contains $\sqrt{2}$ as well. A result of modern research are so-called *rational* algorithms that compute as much of the integral as can be kept within $\mathbb{Q}(x)$, and compute the minimal algebraic extension of $\mathbb{K}$ necessary to express the integral.

The Rothstein-Trager algorithm.

It follows from the PFD of G/H, *i.e.*, $G/H = \sum_{i=1}^{n} c_i/(x - \gamma_i)$, $c_i, \gamma_i \in \mathbb{C}$, that

$$\int \frac{G}{H}\, dx \quad = \quad \sum_{i=1}^{\deg(H)} c_i \log(x - \gamma_i) \tag{3.7}$$

where the γ_i are the zeros of H in $\mathbb{C}$ and the c_i are the residues of G/H at the γ_i. Computing those residues without splitting H into irreducible factors is achieved by the Rothstein-Trager theorem, as follows. Since we seek roots of H and their corresponding residues given by evaluating $c = G/H'$ at the roots, it follows that the c_i are exactly the zeros of the *Rothstein-Trager resultant* $R := \text{resultant}_x(H, G - cH')$, where c here is an indeterminate. Moreover, the splitting field of R over $\mathbb{Q}$ is the minimal algebraic extension of $\mathbb{Q}$ necessary to express $\int f$ in the form given by Liouville's theorem, *i.e.*, as a sum of logarithms, and we have

$$\int \frac{G}{H} dx = \sum_{i=1}^{m} \sum_{c|U_i(c)=0} c \log(\gcd(H, G - cH')) \tag{3.8}$$

where $R = \prod_{i=1}^{i=m} U_i^{e_i}$ is the irreducible factorization of R over $\mathbb{Q}$.

The Lazard-Rioboo-Trager algorithm. Consider the subresultant pseudo-remainder sequence R_i, where $R_0 = R$ is the resultant (see p. 115 in [4]) of H and $G - cH'$ w.r.t. x. Observe that the resultant R is a polynomial in c of degree $\deg(H)$, the roots of which are the residues of G/H. Let $U_1 U_2^2 \cdots U_m^m$ be a square-free factorization of R. Then, we have

$$\int \frac{G}{H} dx = \sum_{i=1}^{m} \sum_{c|U_i(c)=0} c \log(\gcd(H, G - cH')), \tag{3.9}$$

which groups together terms of the PFD with common residue, as determined by the multiplicity of U_i in the squarefree factorization. We compute the inner sum as follows. If all residues of H are equal, there is a single nontrivial squarefree factor with $i = \deg(H)$ yielding $\sum_{c|U_i(c)=0} c \log(H)$; otherwise, that is, if $i < \deg(H)$, the sum is $\sum_{c|U_i(c)=0} c \log(S_i)$, where $S_i = \text{pp}_x(R_k)$, where $\deg_x(R_k) = i$ and pp_x stands for primitive part w.r.t. x. Consequently, this approach requires isolating only the complex roots of the square-free factors $U_1, U_2, \ldots, U_m$, whereas methods based on the PFD require isolating the real or complex roots of the polynomial H, where $\deg(H) \geq \sum_i \deg(U_i)$. However, the coefficients of R (and possibly those of $U_1, U_2, \ldots, U_m$) are likely to be larger than those of H. Overall, depending on the example, the computational cost of root isolation may put at advantage any of those approaches in comparison to the others.

3.3 The Algorithms

We consider two symbolic-numeric algorithms, both based on Hermite reduction for the rational part and using two distinct methods for the transcendental part, one based on structured partial fraction decomposition and the other the Lazard-Rioboo-Trager algorithm, both reviewed in Section 3.2. Following the notation used in equation (3.2), we assume the rational part C/D has been computed and we consider how the transcendental part is computed by the two methods. Both algorithms use MPSolve to control the precision on the root isolation step.

Following the notations used in equation (3.9), the LRT-based method proceeds by computing the sub-resultant chain $(R_0, R_1, \ldots)$ and deciding how to evaluate each sum

$$\sum_{c \,|\, U_i(c)=0} c \, \log(R_k), \qquad \deg(R_k) = i,$$

by applying the strategy of Lazard, Rioboo and Trager. However, we compute the complex roots of the polynomials $U_1, U_2, \ldots, U_m$ numerically instead of representing them symbolically as in [11, 10]. Then, we evaluate each sum $\sum_{c|U_i(c)=0} c \log(R_k)$ by an algorithm adapted to this numerical representation of the roots. This method is presented as algorithm 1 (see page 49).

The PFD-based method begins by computing numerically the roots γ_i of the denominator $H(x)$ and then computes exactly the resulting residues $c_i = c(\gamma_i) = G(\gamma_i)/H'(\gamma_i)$. The numerical rootfinding can destroy the structure of repeated residues, which we restore by detecting residues that differ by less than ε, the user-supplied tolerance. The resulting partial fraction decomposition can then be integrated using the structure-preserving strategy presented in section 3.2 above. This strategy allows to algorithm to replicate the structure of the final output from the LRT algorithm as a sum of real logarithms and arctangents. This method is presented as algorithm 2 (see page 49).

We remark that there can be an issue here in principle as a result of roots of H that are closer than ε. Given the properties of MPSolve, however, this is not an issue in practice, given the ability to compute residues exactly or with sufficiently high precision, because MPSolve isolates roots within regions where Newton's method converges quadratically. In the unlikely event of residues that are distinct but within ε of each other, the algorithm still results in a small error and is advantageous in terms of numerical stability. This is because identifying nearby roots shifts the problem onto the pejorative manifold, as mentioned above.

Both methods take as input a univariate rational function $f(x) = A(x)/B(x)$ over $\mathbb{Q}$ with $\deg(B) > \deg(A)$, and a tolerance $\varepsilon > 0$. Both $A(x)$ and $B(x)$ are expressed in the monomial basis. They yield as output an expression

$$\int \hat{f}(x)\, dx = \frac{C}{D} + \sum v_i \log(V_i) + \sum w_j \arctan(W_j), \tag{3.10}$$

where $V_i, W_j \in \mathbb{Q}[x]$ and $\hat{f}(x)$ is the nearby integrand corresponding to the computed integral, along with a linear estimate of the forward and backward error. The backward error on an interval $[a,b]$ is measured in terms of $\|\delta(x)\|_\infty = \max_{a\le x\le b}|\delta(x)|$, where $\delta(x) = f(x) - \frac{d}{dx}\int \hat{f}(x)\,dx = f(x) - \hat{f}(x)$. The forward error on $[a,b]$ is measured in terms of $\left\|\int (f(x) - \hat{f}(x))\,dx\right\|_\infty = \|\int \delta(x)dx\|_\infty$, where $\int f(x)dx$ and $\int \hat{f}(x)dx$ are assumed to have the same constant of integration. Where f has no real singularities, the results in the following section provide error bounds over $\mathbb{R}$, and where f has real singularities the bounds can be used to determine how close to the singularity the error exceeds the tolerance.

The main steps of algorithm 1 and algorithm 2 are listed below, where the numbers between parentheses refer to lines of the pseudo-code below. Both algorithms begin with:

(1-4:) Decompose $\int f\,dx$ into $\frac{C}{D}$ (rational part) and $\int \frac{G}{H}\,dx$ (transcendental part) using Hermite reduction;

Algorithm 1 then proceeds with:

(5-6:) Compute symbolically the transcendental part $\int \frac{G}{H}\,dx = \sum_i \sum_{c|U_i(c)=0} c\cdot\log(S_i(t,x))$ using Lazard-Rioboo-Trager algorithm; in the pseudo-code $\boldsymbol{U}$ is a vector holding the square-free factors of the resultant while $\boldsymbol{S}$ holds the primitive part of elements of the sub-resultant pseudo-remainder sequence corresponding to elements of $\boldsymbol{U}$, *viz.*, such that corresponding to U_i is $S_i = \mathrm{pp}_x(R_k)$, where $\deg_x(R_k) = i$;

(7:) Compute the roots c_k of $U_i(c)$ numerically using MPSolve to precision ε.

(8-9:) Compute the log and arctan terms using symbolic post-processing in bpas.

After Hermite reduction, algorithm 2 continues with:

(5-6:) Compute the roots γ_k of $H(x)$ numerically using MPSolve to precision ε.

(7:) Compute the residues c_k of $G(x)/H(x)$ corresponding to the approximate roots of $H(x)$ and detect their identity within ε.

(8:) Compute identical residues within ε and then compute a correspondence φ (one-many relation) between a representative residue and its corresponding roots. φ correlates indices of selected elements of $\boldsymbol{c}$, the vector of residues, and indices of elements of $\boldsymbol{\gamma}$, the vector of roots.

(9-10:) Compute symbolically the transcendental part $\int \frac{\hat{G}}{\hat{H}}\,dx = \sum v_i \log(V_i) +$ $\sum w_j \arctan(W_j)$ from the PFD of $\hat{G}(x)/\hat{H}(x)$.

Both algorithms complete the integration by processing the arctangent terms, which can be written as $\arctan\left(\frac{X}{Y}\right)$ or $\arctan(X,Y)$, for polynomials X and Y, after the integration is complete, using Rioboo's singularity removal (RSR) method (described in [2]) based on equation (3.5) and the extended Euclidean algorithm to remove spurious singularities. The result is the conversion of the arctangent of a rational function or two-argument arctangent into a sum of arctangents of polynomials.

Algorithm 1 symbolicNumericIntegrateLRT(f,ε)
$f \in \mathbb{Q}(x)$, $\varepsilon > 0$

```
1: $(g,h) \leftarrow$ hermiteReduce(num($f$), den($f$)) // Note: $g,h \in \mathbb{Q}(x)$
2: $(Quo, Rem) \leftarrow$ euclideanDivide(num($h$), den($h$)) // Note: $Quo, Rem \in \mathbb{Q}[x]$
3: if $Quo \neq 0$ then
4:     $P \leftarrow$ integrate($Quo$)
5: if $Rem \neq 0$ then
6:     $(\boldsymbol{U}, \boldsymbol{S}) \leftarrow$ integrateLogPart($Rem$, den($h$)) // Note: $\boldsymbol{U} = (U_i, 1 \le i \le m)$ and $\boldsymbol{S} = (S_i)$ are vectors with coefficients in $\mathbb{Q}[t]$ and $\mathbb{Q}[t,x]$ respectively
7: $\boldsymbol{c} \leftarrow$ rootsMP($\boldsymbol{U}, \epsilon$) // Note: $\boldsymbol{c} = (c_k)$ are the roots of $U_i$, as returned by MPSolve
8: $(\boldsymbol{L}, \boldsymbol{A2}) \leftarrow$ logToReal($\boldsymbol{c}, \boldsymbol{S}$) // Note: $\boldsymbol{L}$ and $\boldsymbol{A2}$ are, respectively, vectors of logs and two-argument arctangent terms
9: $\boldsymbol{A} \leftarrow$ atan2ToAtan($\boldsymbol{A2}$)
10: return $(P, g, \boldsymbol{L}, \boldsymbol{A})$
```

Algorithm 2 symbolicNumericIntegratePFD(f,ε)
$f \in \mathbb{Q}(x)$, $\varepsilon > 0$

```
1: $(g,h) \leftarrow$ hermiteReduce(num($f$), den($f$)) // Note: $g,h \in \mathbb{Q}(x)$
2: $(Quo, Rem) \leftarrow$ euclideanDivide(num($h$), den($h$)) // Note: $Quo, Rem \in \mathbb{Q}[x]$
3: if $Quo \neq 0$ then
4:     $P \leftarrow$ integrate($Quo$)
5: if $Rem \neq 0$ then
6:     $\boldsymbol{\gamma} \leftarrow$ rootsMP(den($h$), $\epsilon$) // Note: $\boldsymbol{\gamma} = (\gamma_k)$ are the roots of den($h$), as returned by MPSolve
7:     $\boldsymbol{c} \leftarrow$ residues($Rem$, den($h$), $\boldsymbol{\gamma}$) // Note: $\boldsymbol{c} = (c_k)$ are the residues corresponding to the $\gamma_i$
8:     $\varphi \leftarrow$ residueRootCorrespondence($\boldsymbol{c}, \boldsymbol{\gamma}, \varepsilon$) // Note: $\varphi \subseteq \mathbb{N} \times \mathbb{N}$
9:     $(\boldsymbol{L}, \boldsymbol{A2}) \leftarrow$ integrateStructuredPFD($\boldsymbol{c}, \boldsymbol{\gamma}, \varphi$) // Note: $\boldsymbol{L}$ and $\boldsymbol{A2}$ are, respectively, vectors of logs and two-argument arctangent terms
10:     $\boldsymbol{A} \leftarrow$ atan2ToAtan($\boldsymbol{A2}$)
11: return $(P, g, \boldsymbol{L}, \boldsymbol{A})$
```

3.4 Analysis of the Algorithm

We now consider the error analysis of the symbolic-numeric integration using LRT and PFD. We present a linear forward and backward error analysis for both methods.[4]

Theorem 3.11 (Backward Stability) *Given a rational function $f = A/B$ satisfying* $\deg(A) < \deg(B)$, $\gcd(A,B) = 1$ *and input tolerance ε, algorithm 1 and algorithm 2 yield an integral of a rational function $\hat{f}$ such that for $\Delta f = f - \hat{f}$,*

$$\|\Delta f\|_\infty = \max_x \left| \sum_k \mathsf{Re}\left(\Xi(x, r_k)\right) \right| + O(\varepsilon^2),$$

[4]Note that throughout this section we assume that the error for the numerical rootfinding for a polynomial $P(x)$ satisfies the relation $|\Delta r| \le \varepsilon |r|$, where r is the value of the computed root and Δr is the distance in the complex plane to the exact root. This can be accomplished using MPSolve by specifying an error tolerance of ε. Given the way that MPSolve isolates and then approximates roots, the bound is generally satisfied by several orders of magnitude.

where the principal term is $O(\varepsilon)$, r_k ranges over the evaluated roots and the function Ξ defined below is computable. This expression for the backward error is finite on any closed, bounded interval not containing a root of $B(x)$.

We remark that the result is expressed in terms of an unspecified function $\Xi(x, r_k)$ both because the PFD-based and LRT-based methods result in different expressions and because the two methods compute roots of different quantities (roots of the denominator of the integrand for the PFD-based method, and roots of the Rothstein-Trager resultant for the LRT-based method).

Proof. (PFD-based backward stability) The PFD method begins by using Hermite reduction to obtain

$$\int f(x)\,dx = \frac{C(x)}{D(x)} + \int \frac{G(x)}{H(x)}\,dx, \tag{3.12}$$

where $H(x)$ is squarefree. Given the roots γ_i of $H(x)$ we may obtain the PFD of $G(x)/H(x)$, yielding

$$\frac{G(x)}{H(x)} = \sum_{i=1}^{\deg(H)} \frac{c_i}{x - \gamma_i}, \tag{3.13}$$

where $c_i = c(\gamma_i)$ with $c(x) = G(x)/H'(x)$. Taking account of identical residues, the expression (3.13) can then be integrated using the structured PFD algorithm described in Section 3.2. Since we approximate the roots of H, we replace the exact roots γ_i with the approximations $\hat{\gamma}_i$. This destroys the symmetry of the exactly repeated residues, thus the (exact) c_i are modified in two ways: by evaluating $c(x)$ at $\hat{\gamma}_i$; and by adjusting the list of computed residues to restore symmetry, so that residues within ε of each other are coalesced. This strategy requires some method of selecting a single representative for the list of nearby residues; the error analysis then estimates the error on the basis of the error of this representative.[5] We then represent this adjusted computed list of residues by $\hat{c}_i$. Since the Hermite reduction and PFD are equivalent to a rewriting of the input function $f(x)$ as

$$f(x) = \frac{C'(x)}{D(x)} - \frac{C(x)D'(x)}{D(x)^2} + \sum_{i=1}^{\deg(H)} \frac{c_i}{x - \gamma_i},$$

the modified input $\hat{f}(x)$ that algorithm 2 integrates exactly is obtained from the above expression by replacing c_i and γ_i with $\hat{c}_i$ and $\hat{\gamma}_i$.

[5]Note that we assume that ε is sufficiently small to avoid spurious identification of residues in this analysis. Even with spurious identification, however, the backward error analysis would only change slightly, *viz.*, to use the maximum error among the nearby residues, rather than the error of the selected representative residue.

To compute the backward error we first must compute the sensitivity of the residues to changes in the roots. Letting $\Delta\gamma_i = \gamma_i - \hat{\gamma}_i$, then to first order we find that

$$c_i = c(\gamma_i) = c(\hat{\gamma}_i) + c'(\hat{\gamma}_i)\Delta\gamma_i + O(\Delta\gamma_i^2),$$

where $c' = \frac{G'}{H'} - \frac{GH''}{H'^2}$. So the backward error for a given term of the PFD is

$$\begin{aligned}
\frac{c_i}{x-\gamma_i} - \frac{\hat{c}_i}{x-\hat{\gamma}_i} &= \frac{(c_i - \hat{c}_i)(x-\hat{\gamma}_i) + \hat{c}_i\Delta\gamma_i}{(x-\gamma_i)(x-\hat{\gamma}_i)} + O(\Delta\gamma_i^2) && (3.14)\\
&= \frac{c'(\hat{\gamma}_i)\Delta\gamma_i}{(x-\hat{\gamma}_i-\Delta\gamma_i)} + \frac{\hat{c}_i\Delta\gamma_i}{(x-\hat{\gamma}_i)(x-\hat{\gamma}_i-\Delta\gamma_i)} + O(\Delta\gamma_i^2) && (3.15)\\
&= \frac{c'(\hat{\gamma}_i)\Delta\gamma_i}{(x-\hat{\gamma}_i)} + \frac{\hat{c}_i\Delta\gamma_i}{(x-\hat{\gamma}_i)(x-\hat{\gamma}_i)} + O(\Delta\gamma_i^2). && (3.16)
\end{aligned}$$

Since any identified residues all approximate the same exact residue c_k, we use the error $c'(\gamma_k)$ for the residue $\hat{c}_k$ selected to represent the identical residues.

Now, because the rational part of the integral is computed exactly, only the PFD contributes to the backward error. Given that γ_i is an exact root of $H(x)$

$$H(\gamma_i) = 0 = H(\hat{\gamma}_i) + H'(\hat{\gamma}_i)\Delta\gamma_i + O(\Delta\gamma_i^2),$$

where $H(\hat{\gamma}_i) \neq 0$ unless the exact root is computed, and $H'(\gamma_i) \neq 0$ (and hence $H'(\hat{\gamma}_i) \neq 0$) because H is squarefree. Thus, we have that $\Delta\gamma_i = -H(\hat{\gamma}_i)/H'(\hat{\gamma}_i)$ to first order, where $|\Delta\gamma_i| \leq \varepsilon|\hat{\gamma}_i|$. We therefore find that

$$\Delta f = f - \hat{f} = -\sum_{i=1}^{\deg(H)} \left(\frac{c'(\hat{\gamma}_i)}{x-\hat{\gamma}_i} + \frac{\hat{c}_i}{(x-\hat{\gamma}_i)^2}\right)\frac{H(\hat{\gamma}_i)}{H'(\hat{\gamma}_i)} + O(\varepsilon^2). \qquad (3.17)$$

Since the summand is a rational function depending only on x and $\hat{\gamma}_i$, for fixed x the imaginary parts resulting from complex conjugate roots will cancel, so that only the real parts of the summand contribute to the backward error. We therefore find a first order expression of the backward error in the form of the theorem statement with

$$\boxed{\Xi(x, r_k) = \left(\frac{c'(r_k)}{x-r_k} + \frac{c(r_k)}{(x-r_k)^2}\right)\frac{H(r_k)}{H'(r_k)}},$$

where r_k ranges over the computed roots of $H(x)$. This expression is $O(\varepsilon)$ because $\frac{H(r_k)}{H'(r_k)}$ is $O(\varepsilon)$. □

Note that, to properly account for the adjusted residue, applying the formula for Ξ in the PFD case requires taking r_k to be the γ_k used to evaluate the representative residue.

Proof. (LRT-based backward stability) The LRT algorithm produces an exact integral of the input rational function in the form

$$\int f(x)\,dx = \frac{C(x)}{D(x)} + \sum_{i=1}^{n} \sum_{c \,|\, U_i(t)=0} c \cdot \log(S_i(c,x)). \tag{3.18}$$

Given a list $c_{ij} \in \mathbb{C}$, $1 \le j \le \deg(U_i)$ of roots of $U_i(t)$, we can express the integral in the form

$$\int f(x)\,dx = \frac{C(x)}{D(x)} + \sum_{i=1}^{n} \sum_{j=1}^{\deg(U_i)} c_{ij} \cdot \log(S_i(c_{ij},x)),$$

where n is the number of nontrivial squarefree factors of $\mathrm{resultant}_x(H, G - cH')$. Taking the derivative of this expression we obtain an equivalent expression of the input rational function as

$$f(x) = \frac{C'(x)}{D(x)} - \frac{C(x)D'(x)}{D(x)^2} + \sum_{i=1}^{n} \sum_{j=1}^{\deg(U_i)} c_{ij} \frac{S_i'(c_{ij},x)}{S_i(c_{ij},x)}. \tag{3.19}$$

The modified input $\hat{f}(x)$ that algorithm 1 integrates exactly is obtained from this expression by replacing the exact roots c_{ij} with their approximate counterparts $\hat{c}_{ij}$.

To compute the backward error, we must compute the sensitivity of (3.19) to changes of the roots. Considering f as a function of the parameters c_{ij}, and letting $\Delta c_{ij} = c_{ij} - \hat{c}_{ij}$, the difference between the exact root and the computed root, we find by taking partial derivatives with respect to the c_{ij} that

$$\begin{aligned} f(x, c_{11}, \ldots, c_{n\deg(U_n)}) &= f(x, \hat{c}_{11}, \ldots, \hat{c}_{n\deg(U_n)}) + \\ &\sum_{i=1}^{n} \sum_{j=1}^{\deg(U_i)} \left[\frac{\frac{\partial S_i(c,x)}{\partial x}}{S_i(c,x)} + c \left(\frac{\frac{\partial^2 S_i(c,x)}{\partial x \partial c}}{S_i(c,x)} - \frac{\frac{\partial S_i(c,x)}{\partial x}\frac{\partial S_i(c,x)}{\partial c}}{S_i(c,x)^2} \right) \right] \Bigg|_{c=\hat{c}_{ij}} \Delta c_{ij} + O(\Delta c_{ij}^2). \end{aligned} \tag{3.20}$$

Since $f(x, \hat{c}_{11}, \ldots, \hat{c}_{n\deg(U_n)}) = \hat{f}(x)$, letting the rational function in square brackets be denoted by $\xi_i(c,x)$, we have that

$$\Delta f = f - \hat{f} = \sum_{i=1}^{n} \sum_{j=1}^{\deg(U_i)} \xi_i(\hat{c}_{ij}, x) \Delta c_{ij} + O(\Delta \hat{c}_{ij}^2).$$

Given that $U_i(c_{ij}) = 0 = U_i(\hat{c}_{ij}) + U_i'(\hat{c}_{ij})\Delta c_{ij} + O(\Delta c_{ij}^2)$, we have that $\Delta c_{ij} = -U_i(\hat{c}_{ij})/U_i'(\hat{c}_{ij})$ to first order, where $|\Delta c_{ij}| \le \varepsilon |\hat{c}_{ij}|$. Since, as for the PFD case, the imaginary terms from complex roots cancel, we therefore find a first order expression for the backward error in the form required by the theorem with

$$\boxed{\Xi(x,r_k) = \left[\frac{\frac{\partial S_i(r,x)}{\partial x}}{S_i(r,x)} + r\left(\frac{\frac{\partial^2 S_i(r,x)}{\partial x \partial r}}{S_i(r,x)} - \frac{\frac{\partial S_i(r,x)}{\partial x}\frac{\partial S_i(r,x)}{\partial r}}{S_i(r,x)^2}\right)\right]\Bigg|_{r=r_k} \frac{U_i(r_k)}{U_i'(r_k)}},$$

where r_k runs over the roots $\hat{c}_{ij}$. This expression is $O(\varepsilon)$ because $\frac{U_i(r_k)}{U_i'(r_k)}$ is $O(\varepsilon)$. □

Note that the backward error is structured, because the manner in which the integral is computed preserves certain structure in the integrand for both the LRT-based algorithm 1 and the PFD-based algorithm 2. For the LRT-based method, the fact that the residues are computed without perturbing the roots entails that the use of Hermite reduction guarantees that the roots of the denominator of $\hat{f}(x)$ have the same multiplicity as the roots of $\hat{f}$. The use of subresultants and the Rothstein-Trager resultant in algorithm 1 also ensures that the approximated residues have the same multiplicity as the exact residues. For the PFD-based method, the approximation of the roots of $H(x)$ entails that the roots in the denominators of the derivatives of the rational and transcendental parts are slightly different. Thus, the denominator root multiplicity is only preserved approximately, which is why approximate gcd is needed to simplify the derivative of the integral. The identification of nearby computed residues in algorithm 2, however, ensures that the multiplicity of residues in the PFD of G/H is preserved exactly, so that for the transcendental part the PFD of f and $\hat{f}$ have the same structure. The preservation of residue multiplicity translates into higher degree arguments in the log and arctan terms of the integral than would be obtained by a standard PFD algorithm, leading to structured forward error as well.

It is important to reflect on the behaviour of these error terms $\Xi(x,r_k)$ near singularities of the integrand, which correspond to real roots of $H(x)$ (and $B(x)$). For both algorithms, Ξ contains a particular polynomial in the denominator that evaluates to zero at the real roots, specifically $x-\gamma_i$ and $S_i(c_{ij},x)$. In both cases, the expression of Ξ has a term with the particular polynomial squared, which therefore asymptotically dominates the value of the error term near the singularity. This fact is important for efficient computation of the size of the error term near a singularity, since the scaling behaviour can be used to quickly locate the boundary around the singularity where the error starts to exceed the tolerance. Our implementation discussed in Section 3.5 uses this scaling to compute such boundaries.

We turn now to the consideration of forward stability of the algorithms. We note that a full forward error analysis on this problem has subtleties on account of the numerical sensitivities of the log function. This does not affect the validity of the forward error results to follow (that contain both a log term and a simple pole) because near singularities the log term is dwarfed by the pole term, so can be safely ignored in the computation of singularity boundaries. It is a concern, however, when it comes to evaluation of the expressions of the integral. This issue is reflected in the mastery that went into Kahan's "atypically modest" expression in [7], which is written to optimize numerical stability of evaluation. We can, however, sidestep such concerns through the careful use of multiprecision numerics where the value is needed.

Theorem 3.21 (Forward Stability) *Given a rational function $f = A/B$ and tolerance ε, algorithm 1 and algorithm 2 yield an integral of a rational function $\hat{f}$ in the form (3.3) such that*

$$\|\Delta \textstyle\int f\,dx\|_\infty = \max_x \left|\sum_k \left(\Xi(r_k, s_k, x) + \Theta(r_k, s_k, x)\right)\right| + O(\varepsilon^2),$$

where the leading term is $O(\varepsilon)$, r_k and s_k range over the real and imaginary parts of evaluated roots, and the functions Ξ and Θ defined below, corresponding to log and arctangent terms, respectively, are computable. This expression for the forward error is finite on any closed, bounded interval not containing a root of $B(x)$.

Proof. (LRT-based forward stability) Given the exact roots $c_{j\ell}$ of the $U_j(c)$ so that we can express the integral of the input rational function in the form

$$\int f(x)\,dx = \frac{C(x)}{D(x)} + \sum_{j=1}^{n} \sum_{\ell=1}^{\deg(U_j)} c_{j\ell} \cdot \log(S_j(c_{j\ell}, x)).$$

Since the roots $c_{j\ell}$ are complex, to get a real expression for the integral we can convert the transcendental part into a sum of logarithms and arctangents using the real and imaginary parts of the $c_{j\ell}$.

For the remainder of the proof we work with c_k, a subsequence of the roots $c_{j\ell}$ of the squarefree factors of the Rothstein-Trager resultant such that only one of each complex conjugate pair is included, and define φ to be a mapping given by $k \mapsto j$ so that $S_{\varphi(k)}(c_k, x)$ is the term of the integral corresponding to the residue c_k. For each c_k we let a_k and b_k be its real and imaginary parts, respectively. This allows us to express the integral in terms of logarithms and arctangent terms such that

$$\int f\,dx = \frac{C}{D} + \sum_{k=1}^{m} \left[a_k \log(V_k) + 2b_k \arctan(W_{1k}, W_{2k})\right], \tag{3.22}$$

where V_k, W_{1k} and W_{2k} are functions of a_k, b_k and x, and m is the size of the set $\{c_k\}$ of residues.

Once again, since the rational part of the integral is computed exactly, it does not contribute to the forward error. The forward error is the result of the evaluation of the above expression at approximate values for the a_k and b_k. Therefore, considering the variation of equation (3.22) with respect to changes in the a_k and b_k we obtain

$$\begin{aligned}&\Delta\!\int f\,dx = \int (f - \hat{f})(x)\,dx = \\ &\quad\sum_{k=1}^{m}\left\{\left[\left(\frac{\partial V_k}{\partial a_k}\Delta a_k + \frac{\partial V_k}{\partial b_k}\Delta b_k\right)\frac{a_k}{V_k} + \log(V_k)\Delta a_k\right] + \left[\left(W_{2k}\frac{\partial W_{1k}}{\partial a_k} - W_{1k}\frac{\partial W_{2k}}{\partial a_k}\right)\Delta a_k + \right.\right.\\ &\quad\left.\left.\left(W_{2k}\frac{\partial W_{1k}}{\partial b_k} - W_{1k}\frac{\partial W_{2k}}{\partial b_k}\right)\Delta b_k\right]\frac{2b_k}{W_{1k}^2 + W_{2k}^2} + 2\mathrm{arctan}(W_{1k}, W_{2k})\Delta b_k\right\} + o(\Delta a_k, \Delta b_k).\end{aligned} \tag{3.23}$$

We now consider how to determine the values of V_k, W_{1k}, W_{2k} and their partial derivatives from information in the computed integral. To simplify notation we let $j = \varphi(k)$. If c_k is real, then we obtain a term of the form $a_k \log(S_j(a_k, x))$. In the complex case, each c_k stands for a complex conjugate pair. As such, we obtain terms of the form

$$(a_k + i\,b_k)\log(S_j(a_k + i\,b_k, x)) + (a_k - i\,b_k)\log(S_j(a_k - i\,b_k, x).$$

Expressing $S_j(a_k + i\,b_k, x)$ in terms of real and imaginary parts as $W_{1k}(x) + i\,W_{2k}(x) \equiv W_{1k}(a_k, b_k, x) + i\,W_{2k}(a_k, b_k, x)$, so that $S_j(a_k - i\,b_k, x) = W_{1k}(x) - i\,W_{2k}(x)$, the expression of the term in the integral becomes

$$a_k \log\left(W_{1k}(x)^2 + W_{2k}(x)^2\right) + i\,b_k \log\left(\frac{W_{1k}(x) + i\,W_{2k}(x)}{W_{1k}(x) - i\,W_{2k}(x)}\right).$$

The observation that $i \log\left(\frac{X+iY}{X-iY}\right)$ has the same derivative as $2\arctan(X, Y)$ allows the term of the integral to be converted into the form of the summand in (3.22) with $V_k(x) = W_{1k}(x)^2 + W_{2k}(x)^2$.

To facilitate implementation, we can express V_k, W_{1k} and W_{2k} and their partials in terms of $S_j(c,x)$ and $\partial S_j(c,x)/\partial c$ as follows. First of all we have that

$$W_{1k}(x) = \mathsf{Re}(S_j(c_k,x)), \quad W_{2k}(x) = \mathsf{Im}(S_j(c_k,x)). \tag{3.24}$$

Then, because c is an indeterminate in $S_j(c,x)$, $\frac{\partial S_j(c,x)}{\partial c}\Big|_{c=c_k} = \frac{\partial S_j(c_k,x)}{\partial a_k}$ with $\frac{\partial S_j(c_k,x)}{\partial a_k} = \frac{\partial W_{1k}(c_k,x)}{\partial a_k} + i\,\frac{\partial W_{2k}(c_k,x)}{\partial a_k}$, so that

$$\frac{\partial W_{1k}}{\partial a_k} = \mathsf{Re}\left(\frac{\partial S_j(c,x)}{\partial c}\bigg|_{c=c_k}\right), \quad \frac{\partial W_{2k}}{\partial a_k} = \mathsf{Im}\left(\frac{\partial S_j(c,x)}{\partial c}\bigg|_{c=c_k}\right). \tag{3.25}$$

In a similar way, and because the derivative with respect to b_k picks up a factor of i, $\frac{\partial W_{1k}}{\partial b_k} = -\frac{\partial W_{2k}}{\partial a_k}$ and $\frac{\partial W_{2k}}{\partial b_k} = \frac{\partial W_{1k}}{\partial a_k}$. It follows, then, that

$$\frac{\partial V_k}{\partial a_k} = 2\left(W_{1k}\frac{\partial W_{1k}}{\partial a_k} + W_{2k}\frac{\partial W_{2k}}{\partial a_k}\right) \text{ and } \frac{\partial V_k}{\partial b_k} = 2\left(W_{2k}\frac{\partial W_{1k}}{\partial a_k} - W_{1k}\frac{\partial W_{2k}}{\partial a_k}\right).$$

For the complex root case, given the error bound $|\Delta c| \le \varepsilon|\hat{c}|$ on the complex roots, we have the same bound on the real and imaginary parts, *viz.*, $|\Delta a| \le \varepsilon|\hat{a}|$, $|\Delta b| \le \varepsilon|\hat{b}|$. Since $\Delta c_k = -U_j(\hat{c}_k)/U_j'(\hat{c}_k)$ to first order, and $\Delta c_k = \Delta a_k + i\Delta b_k$, from (3.23) we therefore obtain an expression for the linear forward error in the form required by the theorem with

$$\boxed{\Xi(\hat{a}_k,\hat{b}_k,x) = \left(2\hat{a}_k\Gamma + \log\left(W_{1k}^2 + W_{2k}^2\right)\right)\mathsf{Re}\left(\frac{U_j}{U_j'}\right) + 2\hat{a}_k\Lambda\,\mathsf{Im}\left(\frac{U_j}{U_j'}\right)}$$

when $a_k \ne 0$, otherwise $\Xi(\hat{a}_k,\hat{b}_k,x) \equiv 0$, and with

$$\boxed{\Theta(\hat{a}_k,\hat{b}_k,x) = 2\hat{b}_k\Lambda\,\mathsf{Re}\left(\frac{U_j}{U_j'}\right) + 2\left(\operatorname{artcan}\left(W_{1k},W_{2k}\right) - \hat{b}_k\Gamma\right)\mathsf{Im}\left(\frac{U_j}{U_j'}\right)},$$

where $\Gamma = \frac{W_{1k}\frac{\partial W_{1k}}{\partial a_k} + W_{2k}\frac{\partial W_{2k}}{\partial a_k}}{W_{1k}^2+W_{2k}^2}$, $\Lambda = \frac{W_{2k}\frac{\partial W_{1k}}{\partial a_k} - W_{1k}\frac{\partial W_{2k}}{\partial a_k}}{W_{1k}^2+W_{2k}^2}$, W_{1k} and W_{2k} are given by (3.24), $\frac{\partial W_{1k}}{\partial a_k}$ and $\frac{\partial W_{2k}}{\partial a_k}$ are given by (3.25), and all expressions, including U_j and U_j', are evaluated at $\hat{c}_k = \hat{a}_k + i\,\hat{b}_k$. These terms are $O(\varepsilon)$ because $\frac{U_j(\hat{c}_k)}{U_j'(\hat{c}_k)}$ is $O(\varepsilon)$.

For the real root case we have a much simpler expression, since $\Theta(\hat{a}_k,\hat{b}_k,x) \equiv 0$ and since $\hat{c}_k = \hat{a}_k$,

$$\boxed{\Xi(\hat{a}_k, \hat{b}_k, x) = \left(\hat{a}_k \frac{\left.\frac{\partial S_j}{\partial c}\right|_{c=\hat{a}_k}}{S_j(\hat{a}_k, x)} + \log(S_j(\hat{a}_k, x)) \right) \frac{U_j(\hat{\alpha}_k)}{U_j'(\hat{\alpha}_k)}},$$

which is also $O(\varepsilon)$. □

Proof. (PFD-based forward stability) Proceeding as we did for the LRT method, if we assume that the roots of the denominator of the polynomial $H(x)$ are computed exactly, then we obtain an exact expression of the integral of f in the form

$$\int f(x)\,dx = \frac{C(x)}{D(x)} + \sum_{i=1}^{\deg(H)} c_i(\gamma_i)\log(x - \gamma_i). \tag{3.26}$$

As in the LRT-based proof, we assume γ_k is a subsequence of the γ_i that includes only one conjugate of each complex root. Then the same techniques for converting this to a sum of logarithms and arctangents can be applied here. Since $H(x)$ is squarefree, all of the $\gamma_k = \alpha_k + i\,\beta_k$ are simple roots, which entails that the integral can be expressed in the form (3.22) where the $V_j(x)$ are equal to $x - \alpha_k$ for a real root and $x^2 - 2\alpha_k + \alpha_k^2 + \beta_k^2$ for a complex root with $a_k = \mathsf{Re}\,(c(\gamma_k))$, with $c(x) = G(x)/H'(x)$. Using the RSR method the $W_{1k}(x) = \alpha_k - x$ and $W_{2k} = \beta_k$ and $b_k = \mathsf{Im}\,(c(\gamma_k))$. Even though the structured integral is not expressed in this form, it is still an exact integral that we approximate, where all subsequent computation we perform is exact. Analyzing the error in this form has the advantage of using information available immediately after the completion of the rootfinding task. Thus, we will analyze the forward error in this form.

Because the residues are now obtained by computation, and we compute the roots $\gamma_k = \alpha_k + i\,\beta_k$ of $H(x)$, we obtain a modified version of the first order forward error formula (3.23), *viz.*,

$$\begin{aligned}\Delta\!\int f\,dx = \int (f - \hat{f})(x)\,dx = \\ \sum_{k=1}^{m} \Bigg\{ \Bigg[\left(\frac{\partial V_k}{\partial \alpha_k}\Delta\alpha_k + \frac{\partial V_k}{\partial \beta_k}\Delta\beta_k \right) \frac{a_k}{V_k} + \left(\frac{\partial a_k}{\partial \alpha_k}\Delta\alpha_k + \frac{\partial a_k}{\partial \beta_k}\Delta\beta_k \right) \log(V_k) \Bigg] + \\ \frac{2\beta_k b_k(\Delta\alpha_k + \Delta\beta_k)}{(\alpha_k - x)^2 + \beta_k^2} + 2\left(\frac{\partial b_k}{\partial \alpha_k}\Delta\alpha_k + \frac{\partial b_k}{\partial \beta_k}\Delta\beta_k \right) \arctan(\alpha_k - x, \beta_k) \Bigg\} + o(\Delta\alpha_k, \Delta\beta_k).\end{aligned} \tag{3.27}$$

Since $c(x) = G(x)/H'(x)$, $c'(x) = \frac{G'(x)}{H'(x)} - \frac{G(x)H''(x)}{H'(x)^2}$, and so it follows that $\frac{\partial a_k}{\partial \alpha_k} = \mathsf{Re}(c'(\gamma_k))$ and $\frac{\partial b_k}{\partial \alpha_k} = \mathsf{Im}(c'(\gamma_k))$. Similarly, $\frac{\partial a_k}{\partial \beta_k} = -\mathsf{Im}(c'(\gamma_k))$ and $\frac{\partial b_k}{\partial \beta_k} = \mathsf{Re}(c'(\gamma_k))$.

For the complex root case, then, since $\Delta\gamma_j = -H(\hat{\gamma}_j)/H'(\hat{\gamma}_j)$ to first order, $\Delta\alpha_j = -\mathsf{Re}(H(\hat{\gamma}_j)/H'(\hat{\gamma}_j))$ and $\Delta\beta_j = -\mathsf{Im}(H(\hat{\gamma}_j)/H'(\hat{\gamma}_j))$. Collecting terms with $\Delta\alpha_k$ together and terms with $\Delta\beta_k$ together, we obtain from equation (3.27) an expression for the linear forward error in the form required by the theorem with $\Xi(\hat{\alpha}_k, \hat{\beta}_k, x) = \Xi_\alpha + \Xi_\beta$, where

$$\boxed{\Xi_\alpha = \left(\frac{2\hat{a}_k(\hat{\alpha}_k - 1)}{(\hat{\alpha}_k - x)^2 + \hat{\beta}_k^2} + \mathsf{Re}\left(c'(\hat{\gamma}_k)\right)\log\left((\hat{\alpha}_k - x)^2 + \hat{\beta}_k^2\right)\right)\mathsf{Re}\left(\frac{H}{H'}\right)}$$

and

$$\boxed{\Xi_\beta = \left(\frac{2\hat{a}_k\hat{\beta}_k}{(\hat{\alpha}_k - x)^2 + \hat{\beta}_k^2} + \mathsf{Im}\left(c'(\hat{\gamma}_k)\right)\log\left((\hat{\alpha}_k - x)^2 + \hat{\beta}_k^2\right)\right)\mathsf{Im}\left(\frac{H}{H'}\right)}$$

when $\alpha_k \neq 0$, otherwise $\Xi(\hat{\alpha}_k, \hat{\beta}_k, x) \equiv 0$, and with $\Theta(\hat{\alpha}_k, \hat{\beta}_k, x) = \Theta_\alpha + \Theta_\beta$, where

$$\boxed{\Theta_\alpha = \left(\frac{2\hat{\beta}_k\hat{b}_k}{(\hat{\alpha}_k - x)^2 + \hat{\beta}_k^2} - \mathsf{Im}\left(c'(\hat{\gamma}_k)\right)\arctan(\hat{\alpha}_k - x, \hat{\beta}_k)\right)\mathsf{Re}\left(\frac{H}{H'}\right)}$$

and

$$\boxed{\Theta_\beta = \left(\frac{2\hat{\beta}_k\hat{b}_k}{(\hat{\alpha}_k - x)^2 + \hat{\beta}_k^2} + \mathsf{Re}\left(c'(\hat{\gamma}_k)\right)\arctan(\hat{\alpha}_k - x, \hat{\beta}_k)\right)\mathsf{Im}\left(\frac{H}{H'}\right)},$$

with H and H' being evaluated in all cases at $\hat{\gamma}_k$. All of these terms are $O(\varepsilon)$ because $\frac{H}{H'}$ is.

In the case of real roots,

$$\boxed{\Xi(\hat{\alpha}_k, \hat{\beta}_k, x) = \left(c'(\hat{\alpha}_k)\log\left(x - \hat{\alpha}_k\right) - \frac{\hat{a}_k}{x - \hat{\alpha}_k}\right)\frac{H(\hat{\alpha}_k)}{H'(\hat{\alpha}_k)}},$$

which is also $O(\varepsilon)$. □

We note again that the forward error is structured for both algorithms. In the LRT-based case, the exact integral is computed and the approximation only perturbs the values of coefficients of polynomials in the integral, such that the (root and residue) multiplicity structure of the PFD of the derivative is preserved. In the PFD-based case only the residue structure of the PFD of the derivative of the transcendental part is preserved due to the identification of the residues that are within ε of each other,[6] though the

[6]This means that when several computed residues are coalesced, the chosen representative root $\hat{\gamma}_{\tilde{k}}$ must be used to evaluate $\hat{a}_k$, $\hat{b}_k$, and $c'(\hat{\gamma}_k)$ must be used to evaluate all of the error terms corresponding to the roots $\hat{\gamma}_k$ that have the same residue.

integral of the rational part is exact. Thus the PFD-based method achieves a result only approximately on the pejorative manifold, while the LRT-based method keeps the result exactly on the pejorative manifold.[7]

Once again, note that the scaling behaviour for the error term for real roots can be used to efficiently compute the boundaries around the singularities in the integral. In this case, the error scales as $(x-\alpha)^{-1}$ and $S_j(a_k, x)^{-1}$, since the quadratic terms appearing the backward error have been integrated. As a result, the forward error grows much more slowly than the backward error does as we approach a singularity, and we get much smaller bounds before the error exceed the tolerance.

3.5 Implementation

We have implemented the algorithms presented in Section 3.3. In our code, the symbolic computations are realized with the *Basic Polynomial Algebra Subprograms* (BPAS) publicly available in source at `http://bpaslib.org`. The BPAS library offers polynomial arithmetic operations (multiplication, division, root isolation, etc.) for univariate and multivariate polynomials with integer, rational or complex rational number coefficients; it is written in C++ with CilkPlus extension for optimization on multicore architectures and built on top of the GMP library and MPSolve, which we now describe.

The numerical portion of our computation relies on MPSolve, publicly available in source at `http://numpi.dm.unipi.it/mpsolve`. The MPSolve library, which is written in C and built upon the GMP library, offers arbitrary precision solvers for polynomials and secular equations of the form $S(x) = 0$ with $S(x) = \sum_{i=1}^{n} \frac{a_i}{x-b_i} - 1$, *a posteriori* guaranteed inclusion radii, even on restricted domains, such as the real line. For a requested output precision of 2^{-d}, it can ensure at least d correct digits in the returned roots (see [1] for more details).

The implementation of both algorithm 1 and algorithm 2 are integrated into the BPAS library; algorithms 1 and 2 can be called, respectively, through the `realSymbolicNumericIntegrate` and `realSymbolicNumericIntegratePFD` methods of the `UnivariateRationalFunction` template class. We abbreviate the `realSymbolicNumericIntegrate` method to `snIntLRT` and the `realSymbolicNumericIntegratePFD` method as `snIntPFD` in the sequel. The following output formats are available in the `UnivariateRatio-`

[7]Note that this may account for the differences in stability between the LRT-based and PFD-based methods observed in section 3.6 below. Because roots are isolated using multiprecision arithmetic, however, both methods nonetheless produce stable results.

`nalFunction` class: approximate (floating point number) or symbolic (rational number) expressions (in either Maple or Matlab syntax); see Figure 3.28 for a combination of floating point and Maple output formats.

```
      -
   /    -1+x^2
   | ----------- dx =
   / 7+5*x^2+x^4
  -
   -0.638053*ln(2.64575+0.53991*x+x^2) + 0.638053*ln(2.64575-0.53991*x+x^2) + 0
  .0969495*arctan(0.168299+0.623434*x) + 0.0969495*arctan(-0.168299+0.623434*x)
```

Figure 3.28: Sample output of `snIntLRT` and `snIntPFD`.

For the integral $\int \frac{(x^2-1)dx}{x^4+5x^2+7}$, Maple provides the expression appearing in Figure 3.1. For the same integral, the bpas/MPSolve routines `snIntLRT` and `snIntPFD` both return the output shown in Figure 3.28 in the default floating point output format. In the data structures, however, the coefficients are stored as multiprecision rational numbers, which can be displayed by changing the output format.

It must be noted that there are differences between algorithms 1 and 2 and their implementations in bpas. The key difference is that `snIntPFD` and `snIntLRT` do additional post-processing. As such, the forward error analysis detailed in section 3.4 assumes a different output expression than is produced finally in the implementations. Both `snIntPFD` and `snIntLRT` do, however, compute the integral in the form assumed in the forward error analysis. There are therefore several reasons why the additional post-processing will not significantly affect the conclusions drawn from the error analysis.

First of all, after the integral is computed in the form of equation (3.22), all further computation in bpas is done using exact computation. As such, the final expression, which uses the RSR method to remove spurious singularities from the arctangents, is mathematically equivalent to the integral in the form of (3.22) from the perspective of the integration problem, *viz.*, they have the same derivative and hence differ only by a constant.

Another reason why the additional post-processing will not affect the forward error evaluation is that converting two-argument arctangent functions of polynomials (or one-argument arctangents of rational functions) to one-arguments arctangents of polynomials increases their numerical stability. This is because the derivative of $\arctan(x)$ is $1/(1+x^2)$, which can never be zero, or even small, for real integrals, whereas the derivative of $\arctan(x_1, x_2)$ and $\arctan(x_1/x_2)$ is $(x_1'x_2 - x_1x_2')/(x_1^2 + x_2^2)$, which can approach zero for nearly real roots of the denominator of the integrand. This changes the denominators of the expressions for $\Theta(r_k, s_k, x)$ appearing in the proof of theorem 3.21. Thus, the application of the RSR method improves the stability of the integral. As such, the worst that can happen in this situation is that the forward error appears large from the perspective of the error analysis when it actually is not. Though this issue will need to be resolved in refinements of the implementation, it is very unlikely to be a significant issue on account of the fact that the error is dominated by the error in the roots, and in practice this error is many orders of magnitude less than the tolerance.

Since the forward error analysis is reliable, modulo the issue just stated, even though we could compute the forward error on the final output, it is a design decision not to do so. This is because a design goal of the algorithm is to hide the error analysis in the main computation of the integral by performing the error analysis in parallel. This is only possible if the error analysis can proceed on information available before the integral computation is completed.

3.6 Experimentation

We now consider the performance of Algorithms 1 and 2 based on their implementations in BPAS. For the purposes of comparing their runtime performance we will consider integration of the following functions:

1. $f_1(x) = \frac{1}{x^n - 2}$;
2. $f_2(x) = \frac{1}{x^n + x - 2}$;
3. $f_3(x) = [n, n]_{e^x/x}(x)$,

where $[m, n]_f(x)$ denotes the Padé approximant of order $[m/n]$ of f. Since $\int \frac{e^x}{x} dx = \mathrm{Ei}(x)$, the non-elementary exponential integral, integrating f_3 provides a way of approximating $\mathrm{Ei}(x)$. These three problems test different features of the integrator on account of the fact that $f_1(x)$ has a high degree of symmetry, while $f_2(x)$ breaks this symmetry, and $f_3(x)$ contains very large integer coefficients for moderate size n. Note that unless otherwise stated, we are running `snIntPFD` and `snIntLRT` with the error analysis computation turned on.

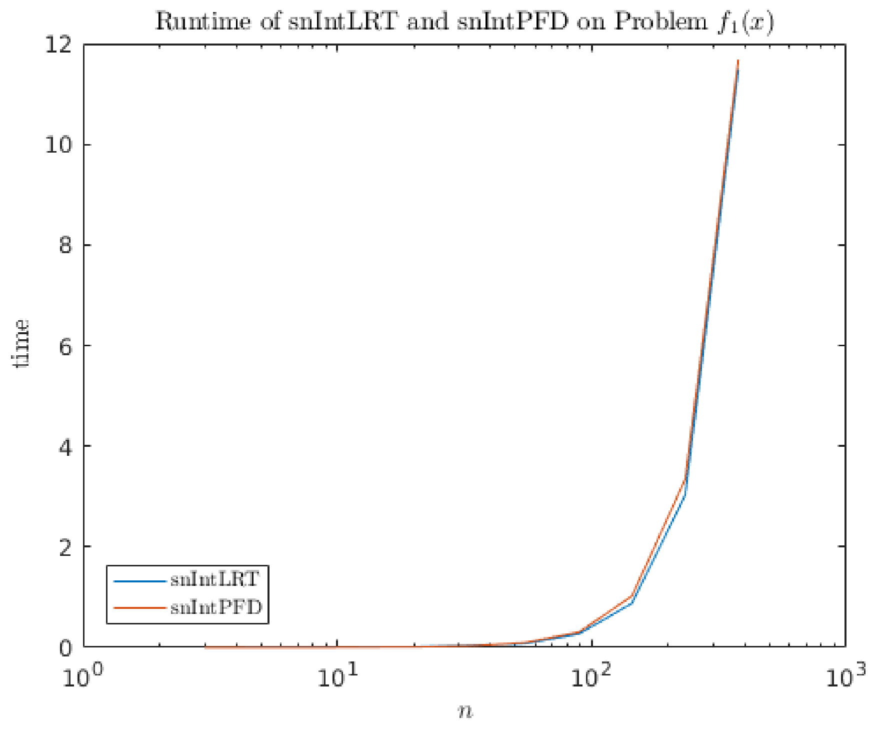
Comparing `snIntPFD` and `snIntLRT` on functions f_1 and f_2 for Fibonacci values of n, from $n = 3$ to $n = 377$, we find the results shown in Figure 3.29. We see from Figure 3.29a that the performance of the two algorithms is nearly identical on function $f_1(x)$. Figure 3.29b shows, however, that on function $f_2(x)$, `snIntPFD` performs considerably poorer than `snIntLRT`. The reason for this is that the size of the coefficients in the Rothstein-Trager resultant grows exponentially for $f_2(x)$. This causes no significant issues for the subresultant computation, but it significantly slows the rootfinding algorithm, leading to large rational number roots, which slows the post-processing algorithms. In contrast, the difference in runtime for snIntPFD on functions $f_1(x)$ and $f_2(x)$ is negligible. This is because the speed of snIntPFD is determined by the degree of the denominator (after the squarefree part has been extracted by Hermite reduction) and the height of the coefficients. Since the denominators of f_1 and f_2 have the same degree and height bound, we expect the performance to be similar.

If we run the same computation with the error analysis turned off, we see that snIntLRT actually performs better than snIntPFD on problem 1. With the performance improvement of snIntLRT relative to snIntPFD being similar to the performance of snIntPFD relative to snIntLRT on problem 2 with the error analysis turned on. Thus, there are some problems on which snIntLRT performs better than snIntPFD. The performance of snIntPFD is easier to predict from the degree and height bound of the input polynomials.

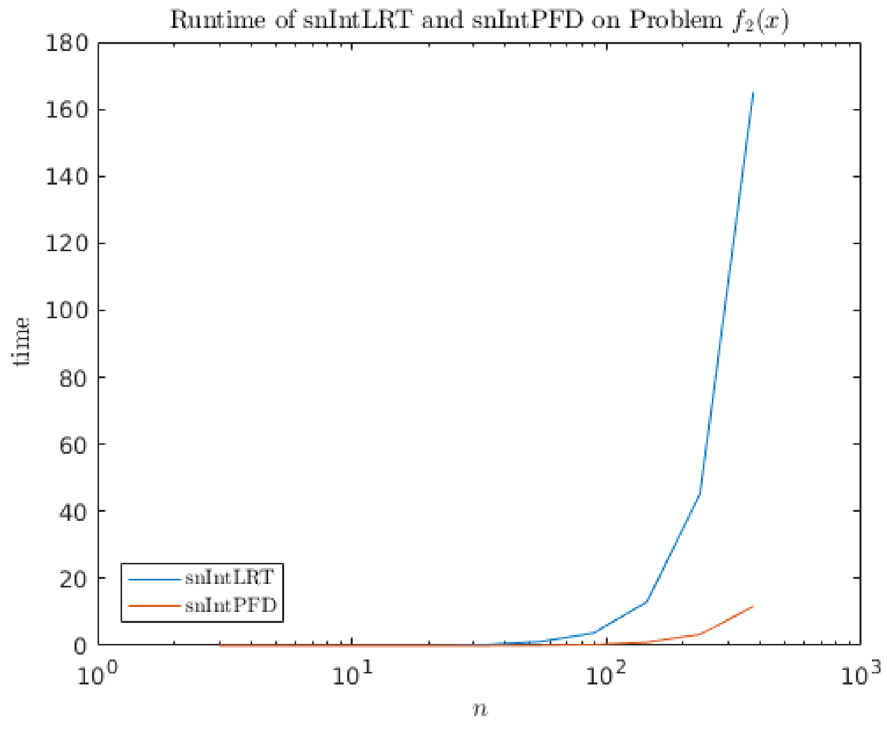
That there is a difference in performance in snIntLRT when the error analysis computation is turned off shows that the current implementation can only partially hide the error analysis computation for snIntLRT on some problems. The error analysis computation is successfully completely hidden for problem 2 with snIntLRT, which is to be expected. For snIntPFD, however, there is a negligible difference in the runtime with the error analysis turned on and off. Thus, once again, snIntPFD has the more reliable and desirable behaviour.

snIntPFD also performs better on problem 3, which leads to coefficients with height bound that grows exponentially with n. For $n = 8$, snIntLRT computes the integral in about 0.04 s, whereas snIntPFD computes it in about 0.01 s. For $n = 13$, the respective runtimes increase to 0.18 s and 0.02 s, and by $n = 21$, around 2.5 s and 0.04 s. This shows that snIntLRT is considerably slowed down by large input coefficients, since this leads to even larger coefficients in the subresultants. This is reflected in the subresultant computation taking 0.6 s for $n = 21$ and slowing down the exact integration to 2.4 s. Thus, when it comes to runtime performance, snIntPFD is the clear winner.

Turning to the error analysis, we now consider the behaviour of the error under variation of the input tolerance ε. For integrands without real singularities, we can compute a global forward and backward error bound over the entire real line. For the non-singular problem $\int \frac{dx}{x^{128}+2}$, a variant of problem 1, we see from table 3.1 that both snIntLRT and snIntPFD exhibit tolerance proportionality as the tolerance is reduced.



(a)



(b)

Figure 3.29: Runtime comparison of `snIntPFD` and `snIntLRT` on problems (a) $f_1(x)$ and (b) $f_2(x)$.

	`snIntLRT`		`snIntPFD`	
ε	forward error	backward error	forward error	backward error
$6\cdot10^{-11}$ (2^{-34})	$8\cdot10^{-16}$	$1\cdot10^{-15}$	$2\cdot10^{-15}$	$2\cdot10^{-12}$
$3\cdot10^{-17}$ (2^{-55})	$3\cdot10^{-55}$	$2\cdot10^{-53}$	$1\cdot10^{-39}$	$1\cdot10^{-38}$
$2\cdot10^{-27}$ (2^{-89})	$1\cdot10^{-75}$	$2\cdot10^{-73}$	$9\cdot10^{-59}$	$8\cdot10^{-58}$
$4\cdot10^{-44}$ (2^{-144})	$6\cdot10^{-95}$	$7\cdot10^{-93}$	$3\cdot10^{-77}$	$2\cdot10^{-76}$

Table 3.1: Tolerance proportionality of the global forward and backward error bounds for `snIntLRT` and `snIntPFD` on $\int \frac{dx}{x^{128}+2}$.

	`snIntLRT`		`snIntPFD`	
ε	forward error ∂	backward error ∂	forward error ∂	backward error ∂
2^{-34}	$4\cdot10^{-3}$	$6\cdot10^{-2}$	$1\cdot10^{-14}$	$6\cdot10^{-7}$
2^{-55}	$7\cdot10^{-23}$	$9\cdot10^{-12}$	$8\cdot10^{-20}$	$3\cdot10^{-10}$
2^{-89}	$4\cdot10^{-32}$	$2\cdot10^{-16}$	$2\cdot10^{-28}$	$1\cdot10^{-14}$
2^{-144}	$2\cdot10^{-34}$	$1\cdot10^{-17}$	$3\cdot10^{-31}$	$6\cdot10^{-16}$

Table 3.2: Tolerance proportionality of the singularity boundary widths for `snIntLRT` and `snIntPFD` on problem 3 with $n = 8$ for the singularity at $x \doteq 10.949$. The symbol ∂ is used to abbreviate "boundary width".

Here `snIntLRT` generally outperforms `snIntPFD` for a given input tolerance by several orders of magnitude, but both algorithms perform strongly.

On problems that do have real singularities, we obtain boundaries around the singularities past which the error exceeds the input tolerance. On problem 3 for $n = 8$, there is a real singularity at $x \doteq 10.949$. For this singularity, we see from table 3.2 that both `snIntLRT` and `snIntPFD` exhibit tolerance proportionality of the singularity boundaries as the tolerance is reduced. Thus, we can get as close to the singularity as desired by decreasing the input tolerance. With the exception of $\varepsilon = 2^{-34}$, `snIntLRT` outperforms `snIntPFD`, but the difference in performance between the two algorithms is not as extreme as with the non-singular case. For the default precision of $\varepsilon = 2^{-53}$ and above, both algorithms get extremely close to the singularity before the error exceeds the tolerance.

For testing the numerical stability, we will consider two additional problems, along with problem 3 above:

4. $f(x) = \frac{2x}{x^2-(1+\epsilon)^2}, \epsilon \to 0$ (singular just outside $[-1,1]$);;
5. $f(x) = \frac{2x}{x^2+\epsilon^2}, \epsilon \to 0$ (nearly real singularities on the imaginary axis).

Note that the small parameter ϵ in problems 4 and 5 is conceptually distinct from the input tolerance ε. These problems are useful for testing the stability of the integration algorithms near singularities.

On problems 4 and 5, `snIntLRT` computes the exact integral, because the integral contains only rational numbers, so there is no need to do any rootfinding. Thus, the forward and backward error are exactly zero, and the evaluation of the integral is insensitive to how close the singularities are to the boundary of the interval of integration, provided a numerically stable method of evaluating the logarithm is used.

On the same problems `snIntPFD` computes very nearly the exact integral. On problem 4, with $\varepsilon = 2^{-53}$, it is possible to get to within about $1.6 \cdot 10^{-23}$ of the singularities at $\pm 1 \pm \epsilon$ before the error exceeds the tolerance. Thus, even with $\epsilon = \varepsilon$, the error does not affect the evaluation of the integral on the interval $[-1, 1]$. `snIntPFD` also performs exceedingly well on problem 5. With the same input tolerance, the forward error bound is $1.9 \cdot 10^{-57}$ for $\epsilon = 0.1$ and increases only to $1.7 \cdot 10^{-42}$ for $\epsilon = 10^{-16}$. Indeed, the difference between the a in $\log(x^2 + a)$ computed by `snIntLRT` and `snIntPFD` is about $1.7 \cdot 10^{-74}$.

Since problem 3 requires rootfinding for both algorithms, it provides a more fair comparison. `snIntLRT` fares slightly better than `snIntPFD` on this problem, but only slightly and not in a way that significantly affects numerical evaluation. We make the comparison for $\varepsilon = 2^{-53}$. With $n = 8$, for `snIntLRT` the backward and forward error bounds away from the real singularities are about $2.1 \cdot 10^{-38}$ and $1.8 \cdot 10^{-36}$, respectively. For `snIntPFD` the backward error bound increases only to $2.6 \cdot 10^{-35}$ and the forward error bound decreases slightly to $1.2 \cdot 10^{-36}$. As for evaluation near the real singularities, `snIntLRT` can get within about $1.8 \cdot 10^{-23}$ of the singularity at around $x = 10.949$ before the forward error exceeds the tolerance. `snIntPFD` can get within about $2 \cdot 10^{-20}$. Of course, this is not a true concern anyway, because the Padé approximant ceases to be a good approximation of e^x/x before reaching the real root.

We see, therefore, that `snIntPFD` performs strongly against `snIntLRT` even when `snIntLRT` gets the exact answer and `snIntPFD` does not. Indeed, the differences in numerical stability between the two methods are relatively small. Given the performance benefits of `snIntPFD`, the PFD-based algorithm is the clear overall winner between the two algorithms. `snIntPFD` is therefore the preferred choice, except in cases where the exact integral needs to be retained.

3.7 Conclusion

We have identified two methods for the hybrid symbolic-numeric integration of rational functions on exact input that adjust the forward and backward error the integration according to a user-specified tolerance, determining the intervals of integration on which the integration is numerically stable. The PFD-based method is overall the better algorithm, being better overall in terms of runtime performance while maintaining excellent numerical stability. The LRT-based method is still advantagous in contexts where the exact integral needs to be retained for further symbolic computation. We believe these algorithms, and the extension of this approach to wider classes of integrands, has potential to increase the utility of symbolic computation in scientific computing.

3.8 Bibliography

[1] Dario A Bini and Leonardo Robol. Solving secular and polynomial equations: A multiprecision algorithm. *Journal of Computational and Applied Mathematics*, 272:276–292, 2014.

[2] Manuel Bronstein. *Symbolic integration*, volume 3. Springer, 1997.

[3] Manuel Bronstein. Symbolic integration tutorial. https://www-sop.inria.fr/cafe/Manuel.Bronstein/publications/issac98.pdf, 1998. Accessed: 2017-12-19.

[4] Bruno Buchberger and Rüdiger Georg Konrad Loos. *Algebraic Simplification*, pages 11–43. Springer Verlag, Wien–New York, 1982.

[5] Richard Fateman. Revisiting numeric/symbolic indefinite integration of rational functions, and extensions. https://people.eecs.berkeley.edu/~fateman/papers/integ.pdf, 2008. Accessed: 2017-12-19.

[6] William Kahan. Conserving confluence curbs ill-condition. Technical report, DTIC Document, 1972. http://www.dtic.mil/get-tr-doc/pdf?AD=AD0766916, Accessed: 2017-12-19.

[7] William M Kahan. Handheld calculator evaluates integrals. *Hewlett-Packard Journal*, 31(8):23–32, 1980.

[8] Matu-Tarow Noda and Eiichi Miyahiro. On the symbolic/numeric hybrid integration. In *Proceedings of the international symposium on Symbolic and algebraic computation*, page 304. ACM, 1990.

[9] Matu-Tarow Noda and Eiichi Miyahiro. A hybrid approach for the integration of a rational function. *Journal of computational and applied mathematics*, 40(3):259–268, 1992.

[10] Renaud Rioboo. Real algebraic closure of an ordered field: Implementation in *Axiom*. In Paul S. Wang, editor, *Proceedings of the 1992 International Symposium on Symbolic and Algebraic Computation, ISSAC '92, Berkeley, CA, USA, July 27-29, 1992*, pages 206–215. ACM, 1992.

[11] Renaud Rioboo. Towards faster real algebraic numbers. *J. Symb. Comput.*, 36(3-4):513–533, 2003.

[12] Zhonggang Zeng. Apatools: a software toolbox for approximate polynomial algebra. In *Software for Algebraic Geometry*, pages 149–167. Springer, 2008.

CHAPTER 4

Effective Validity: A Logical Model for Stable Inference

Effective Validity: A Logical Model for Stable Inference[1]

4.1 Introduction

There is a long tradition in philosophy of science of using logical tools and methods to gain insight into scientific theories, their structure, concepts, methods and their ability to represent the world. Classical formal logic in particular, specifically classical first order logic (FOL) paired with set theory (ST), has been enormously influential in philosophy as a model for correct inference in mathematics, science and inference in general. It informs scores of traditional, though still very influential, views in the philosophy of science. Notable examples of this influence are the Oppenheim-Putnam view of the logical structure of science [8], which has strongly influenced views of reduction and inter-theoretic relations in science, the Hempelian view of scientific explanation [4], which has deeply influenced views of scientific explanation and continues to be influential in its own right, as well as myriad strategies of rational reconstruction of scientific theories, from the early proof-theoretic approaches of the logical empiricists (see [9] for a good historical discussion of these views) to the model-theoretic approaches of Suppes [10, 11], van Fraassen [14, 13] and others.

A common feature of these uses of formal logic as presentations or representations of science or aspects of science, is a projection of actual scientific theories, concepts and methods into a logical framework, typically classical FOL, so that the products of science are presented in a uniform language. The benefit of such a projection is that imprecise concepts and methods can be reformulated in a precise language, clarifying their structure, content and justification, or lack thereof as the case may be, leading to valuable insights into science. A nice example of how a logical analysis can provide useful insights is Popper's doctrine of falsificationism, contributing to the demarcation problem (between science and pseudoscience) and providing a useful heuristic for many practising scientists who are engaged in the development of testable new theories and models.

This strategy of projecting science into logic also has its limitations. A major limitation is that the logic used is invariably deductive, which is not a problem in itself except for the fact that many theories, concepts and methods in scientific practice, use non-deductive forms of approximation. Not only are approximation methods used in

[1] A version of this paper will be submitted for publication in *Fields Institute Communications*.

practice, but in most cases in which they appear, they are essential for making problem solving feasible in the restricted inferential contexts of scientific practice; without approximation methods, much of science would be impossible in practice [7]. Thus, the actual processes of description, prediction, explanation and control in science often use approximation, as many philosophers of science, including Wimsatt [17], Harper [3], Batterman [1] and Wilson [16], now emphasize.

There is a tension here between the supposed purpose of logical reconstructions, which is to distill a product of science to its logical essence, and the capacity to describe and account for actual inferential processes in science. Viewed in relation to the actual theories, methods and concepts of science, logical reconstructions provide a means of abstracting certain logical structure from actual scientific methodology and epistemology while ignoring most of the details. Much like the abstraction processes involved in the construction of mathematical models of natural phenomena, the abstraction processes involved in logical reconstructions introduce forms of error. The price of a clean, clear logical model, where reasoning is exact and deductive, is accurate representation of the methodological and epistemological details of scientific practice.

The error introduced in reconstruction processes is not a problem with the method itself, rather the error places conditions on how these processes can be used reliably as tools to clarify features of scientific method and knowledge. Insofar as rational reconstruction is restricted to a specific purpose, such as conceptual clarification of the nature of theories within a particular theoretical framework, or the elucidation of particularly robust features of scientific theories in general, such as falsifiability, traditional logical reconstructions can be well-suited to the task. One runs into difficulty, however, when the products of such studies form the basis for beliefs about scientific method and knowledge outside of the context in which the reconstructions are valid. Without a keen sensitivity to the error introduced in the abstraction process, one cannot know how broadly the results of a reconstruction provide accurate information about real science, and in which contexts one's beliefs about science are likely to become unreliable.

That the tendency to form beliefs about science on the basis of logical reconstructions is a serious problem in philosophy is reflected by the effort Wilson has expended to bring light to the issue. Wilson [16] identifies a surprisingly common tendency among philosophers to view scientific theories in terms of a generic logical theory T, as formulated with classical first-order logic and set theory. He calls this the "theory T syndrome". Reasoning in this way imposes a massive distortion on one's understanding of science.

Aside from the fact that most of scientific practice cannot be organized into a cleanly presented theory, logical or otherwise, theory T reasoning loses any connection to any structure and content of real scientific theories that is not adequately idealized as a formal system, including any form of approximation. This is not to say that many philosophers conceive of all of science in this way. The point is that insofar as reasoning about science involves theory T reasoning there is little, if any, awareness of the limitations of that reasoning about scientific methods and knowledge, leading to a variety of conceptual distortions and false beliefs.

Recognizing that logical reconstructions are a tool for abstraction away from methodological and epistemological details of real science, we are led to take seriously the forms of error that are introduced in the abstraction. Just as modeling error in mathematical modeling is difficult to quantify, similar difficulties are presented by methodological and epistemological modeling error. This can be mitigated, however, through a strategy that develops an epistemological model by consciously abstracting structural features from a particular scientific context. In this way we know that the epistemological model will be accurate in the context in which it was constructed, and it becomes an empirical question how generally the model applies to scientific practice. It is precisely this sort of task that was undertaken in [6] in the context of mathematical models based on ordinary differential equations, including their computational methods of solution and the rigorous incorporation of measurement data. This approach allows for the development of new kinds of epistemological models that are designed to respect certain structural features of the reasoning or content of particular areas of science.

Given the importance of logical reconstructions in philosophy of science, one research question in this area is whether an epistemological modeling approach can be used to develop models of scientific inference that respect the structure of the actual inference methods used in scientific practice, including approximation. The purpose of this paper, then, is to answer this question in the affirmative, by providing a logical model of stable, approximate inference in terms of a generalization of the deductive concept of logical validity. I call the resulting concept *effective validity*, where the sense of "effective" is that from physics and not computability theory, connoting a capturing of much of the form or functional behaviour of valid inference while producing a concept tolerant of error and approximation. Although one of the main motivations for this generalization of deductive logic, which I call *effective logic*, is to account for approximate inference, the fundamental concept is not approximation. When we reflect on what we require of inferences when approximation is introduced, we may see that we require the inference we wish to make to be *stable* under the *variation* implied by the given kind of approximation. Thus, effective logic is fundamentally about the stability of inferences under different kinds of variation.

Stability is a key property of successful mathematical descriptions of nature. For a mathematical model to successfully describe a phenomenon it must be the case that the description it provides is stable under small changes to the model. If this were not so, then a small change could produce an entirely different description, resulting in a model that no longer describes the phenomenon. A general reason why models capable of describing the world must have this property is that, as was already pointed out, there are always forms of error in the modeling process, so any description we can produce involves error and approximation in relation to the phenomenon we seek to represent.

This idea of stability of mathematical descriptions underlies the technical notion of well-posedness introduced by Hadamard. A problem involving a differential equation is considered to be *well-posed* if there exists a unique solution with the property that the solution varies continuously with small changes to the initial and/or boundary conditions. This continuous variation property is what guarantees that the description (solution) is stable in a mathematical sense, and consequently guarantees its stability as a description of any phenomenon described by a given initial value or boundary value problem.

We may observe that the standard deductive notion of valid inference on its own tells us nothing about whether other nearby inferences, obtained by certain variations of the premises for example, are also valid. Of course if a nearby inference is also an instance of a valid inference form then we know it is also valid, but for inferences in general deductive validity tells us nothing about preservation of validity under changes to propositions. Working in any context in which error and approximation are involved, if our inferences are to be reliable it must be the case, for Hadamard-type reasons, that the inference continues to be valid if the premises are only close to being true. This observation is what underlies the informal definition of effective validity: an inference is *effectively valid* if whenever the premises are nearly true the conclusion is nearly true.[2] Thus, rather than preserving truth, as do valid inferences, effectively valid inferences preserve "near-truth", meaning that nearby inferences continue to be valid.

This informal notion of near-truth is not one that we will work with directly, for the same reasons that the informal notion of truth is avoided in formal logic. Much

[2] I avoid the term "approximate truth" here both because it is a notoriously problematic notion in philosophy and to provide a clearer link to the more precise formulations of effective validity to follow.

as for validity in standard formal logic, this informal notion of effective validity can be formalized in a number of different ways. Accordingly, in the sections that follow, we will consider a number of precise formulations of effective validity in terms of stability of inferences under variations in syntax and semantics of sentences. I will show how this generalization of valid inference captures the form of approximation methods typically used in applied mathematics, though I will suggest that it can capture the form of a much wider range of scientific inference.

Effective logic is a generalization of deductive logic as usually conceived in two different ways. First of all, when the variation is reduced to zero, the concept of effective validity reduces to a deductive form of validity. Thus, in this restricted sense, deductive logic is obtained in an appropriate limit of this generalized logic. This is the sense in which effective logic really is a generalization of deductive logic. It is also a generalization in the sense of an expansion in terms of how a logical representation can elucidate the structure and content of science. Rather than working only with uninterpreted languages and their models, as is standard in the metatheoretical treatment of formal logics, effective logic also makes important use of interpreted languages and mappings between them, since this is standard in much of scientific practice, but particularly in applied mathematics and computational science. Indeed, the focus of this paper is primarily on the effective logic of such interpreted languages. We will see how such an approach can provide a faithful representation of inferential structure in scientific practice, helping to account for how and why scientists develop and employ the methods they do.

The aim of effective logic is not only to provide a more refined tool for capturing the structure of scientific inference. It is hoped that it will be useful for many of the purposes for which formal logic has been used in philosophy of science for more than a century. This includes the elucidation of the structure of theories, clarification of scientific concepts and methods, clarification of the processes of description, prediction, explanation and control, as well as accounting for scientific representation. The idea of effective logic, then, is that with a greater ability to make contact with actual scientific reasoning, a logical representation can provide finer grained philosophical insights into scientific practice that are directly applicable to science itself.

Before we develop the precise formulations of the concepts of effective logic, we must first consider the stability of symbolic expressions and mathematical structures under forms of variation. This is the subject of section 4.2, which examines the stability of syntactic and semantic structures in terms of near-identity transformations. We then move in section 4.3 into the territory of logic proper by considering interpreted languages, which have truth conditions for propositions, and their stability properties under variation of syntactic or semantic structure. We illustrate the ideas through the example of the analytic solution of the simple pendulum problem. We then consider in section 4.4 the effective logic of problem-solving strategies that involve transformations between interpreted languages, taking as an example the numerical integration of the double pendulum problem.

As an indication of the kind of insight that effective logic can make accessible, we will conclude in section 4.5 by discussing the concept of *inferential structure* identified by effective logic and its implications for computational science. It is shown in [7] how a strategy of stable transformation of computational problems underlies strategies of computational complexity reduction in computational science. Notable examples of this sort of strategy are numerical methods, which transform difficult continuous problems into rapidly computable ones using forms of discretization, and modular methods, which transform difficult symbolic problems into many smaller problems that can be rapidly computed within a single machine word, *i.e.*, computed within a single processor cycle. Effective logic then shows that a basic requirement of these strategies is the near-preservation of inferential structure.

4.2 Stability Under Syntactic and Semantic Variation

The shift from deductive validity to effective validity is primarily about introducing a context of variation for the syntax and semantics of sentences together with a consideration of stability of consequence relations under such variations. In this section we will focus on the general kinds of variation that effective logic involves and what such variation looks like for syntactic forms and semantic structures. In particular we will distinguish the kind of variation particular to approximation from other, weaker forms of variation, which will clarify the structure of inferences involving approximation.

Formal logic deals with two precise kinds of validity, corresponding to two kinds of consequence. The first is syntactic validity, which corresponds to deductive consequence. In this case, for a set Γ of assumptions and a sentence p in some formal language, in a formal system based on that language $\Gamma \vdash p$ denotes that the inference from Γ to p is valid because there is a proof of p in the formal system with members of Γ as assumptions. The second is semantic validity, which corresponds to semantic consequence, often called logical consequence. In this case, $\Gamma \models p$ denotes that the inference from Γ to p is valid because in any model of the formal language in which each member of Γ is true, p will also be true. Since it is these precise forms of validity that formal logic works with, it is these concepts, or analogues of them, we seek to generalize to inferences in a context of variation, and approximate inferences in particular.

One of the fundamental reasons for considering approximation in inferences is to be able to draw conclusions when certain amounts of error are unavoidable in assumptions and acceptable in conclusions. Typically when error is introduced the idea is that the inferences can continue to go through provided the size of the error is "small". This sense of smallness is not always precise or even clear, but indicates that some measure of size or distance is needed to quantify it. In the general context of effective logic, however, we seek a way of capturing the sense of "smallness" of a variation without the need of such a measure. This will come down in any given context to "smearing" an object into a (typically well-defined) range of nearly-equivalent objects, leading to similar ranges of propositions containing such objects, and then to inferential relations between such "smeared out" propositions.[3] For the moment, however, we will analyze the notion of smallness in terms of the allied notion of nearness, and in particular *near-identity.*

In the interest of distilling as much as possible the concept of error to its essence, we may observe that this means that, for the purposes of a stable approximate inference, approximately identical assumptions should lead to approximately identical conclusions. Thinking in terms of meaningful assertions, this means that if we make almost the same assertions, then almost the same conclusions should be assertible. Thinking in terms of linguistic assumptions, this means that if we make almost the same assumptions, then almost the same consequences should be provable. We may see from this that the stability of near-identity, in the sense of nearly-identical input producing nearly-identical output, captures the basic relation between assertions/assumptions and conclusions/consequences that effective validity aims to make precise.

Before we consider the notion of near-identity more precisely, there are two important basic features to appreciate concerning this concept of stable approximate inference that make it different from deductive inference. The first of these is that we have replaced a concept of exact consequence with one that requires that a special relation of error obtain between input (assertion/assumption) and output (conclusion/consequence). Naturally enough, the stability of this relation will not always obtain. Thus, approximate inferences run the risk of becoming *unstable.* There are a number of ways in which approximate in-

[3]This is analogous to the mathematical notion of a neighbourhood from topology, but the sets of "smeared out" propositions need not have any topological structure in general.

ferences can become unstable, including: small changes to premises lead to large changes in conclusions (analogous to chaotic behaviour in dynamical systems); the premises themselves become unstable (analogous to decay or decoherence of prepared states in quantum mechanics);[4] or chains of approximate inferences interact in such a way as to no longer be stable (analogous to slippery slope fallacies in probabilistic reasoning).

To be in a position to handle potential instabilities of inferences requires an understanding of the stability of consequence relations. In the context of scientific practice, this can often be handled in terms of support theorems that establish kinds of variation over which consequence relations will be stable. This can provide a means of judging when chains of approximate inferences are stable as well. In general, however, stability is judged by some external means, such as agreement with experiment or observation, when proofs of stability are not available. In such cases, it is the correctness of the expected consequences over a range of variation that establishes stability externally. In the context of this paper we are concerned only with outlining the structure of approximate reasoning. Though it is of crucial importance, the matter of evidence or proofs of stability is beyond our current scope.

The second basic feature of approximate inference is something already alluded to, which is that introducing variation forces a move from considering consequence relations between individual sentences to consequence relations between *ranges* of nearly-identical sentences. Thus, in shifting from deductive to approximate inference we shift to consideration of inferences as correspondences between collections of sentences defined by relations of near-identity. In different terms, this means that to handle approximate inference reliably we need to consider sentences that are nearly-identical, sufficiently close to one another that the same inference form applies to them.

To summarize at this point, we can see that an effectively valid inference will be one that maps a set of ranges of input sentences (corresponding to the set of assertions or assumptions) to a particular range of output sentences (corresponding to the conclusion or consequence), and that joining together effectively valid inferences must be done with care to avoid instability. To become more clear about what this means we need to develop the notion of near-identity responsible for defining these ranges of sentences. The near-identity of sentences will be determined by the near-identity of their components, so that

[4]This case covers statements whose truth depends on the location in a state space, which includes statements whose truth can be inherently variable over time. Since the local truth of a statement can be expressed in terms of its effective validity with no premises or assumptions, this is a special case of the notion of effective validity.

ultimately we need to examine the near-identity of syntactic and semantic entities.

There are two basic ways one can think about near-identity of entities. One is in terms of sets or collections of entities that are nearly-identical to each other, according to some standard; and the other is in terms of the transformations that map an entity to one that is nearly-identical to it. One may see that the near-identity transformations can be understood to *generate* the collections of nearly-identical entities by "smearing out" a given entity. Whichever way we think about ranges of nearly-identical entities, we will always suppose that the near-identity variation is about some fixed centre, corresponding to the particular entity (sentence, equation, expression, structure, object, *etc.*) that is being "smeared out".

A set of near-identity transformations always includes the identity operator $\mathbb{I}$, which leaves any object fixed ($\mathbb{I}a = a = a\mathbb{I}$), specifically the fixed centre of the variation. The case where the identity is the only near-identity transformation corresponds to the case of traditional logic, where all the variation is turned off. For a given non-trivial kind of near-identity variation, we can consider a suitable set of operators that are in a suitable sense "close" to the identity. In some cases such an operator can be written in a form such as $\mathbb{I} + \varepsilon T$, where T is some operator that acts on the given kind of entity and ε is a "small" parameter. When such a near-identity operator acts on an entity it produces a nearby nearly-identical entity ($(\mathbb{I} + \varepsilon T)a = a + \varepsilon Ta = a + \varepsilon b$) that is only slightly different (here by εb) from the centre of variation (here a). A suitably well-defined set of near-identity operators then generates a range of entities nearly-identical to the given entity we are varying around, *i.e.*, the entity we are "smearing out".

To gain a sense of the variety of such transformations, we may observe that such transformations can be continuous or discrete, and can exactly or approximately preserve structural features of the objects they act upon. Consider, for example, the context of continuous groups of transformations, such as the rotations of an object in space. Here the transformations are continuous in the sense that small changes of the input to the transformation yield small changes in the output. For example, for a pair nearly-identical vectors $\mathbf{x}$ and $\mathbf{x}+\varepsilon$ in the plane, rotation by θ about the origin yields another pair of nearly-identical vectors. When such groups also form a manifold, they are called Lie groups, and can be studied in terms of the near-identity transformations of the group. The set of near-identity transformations determine a linear space called the Lie algebra of the group, which can be thought of as specifying all of the possible infinitesimal (near-identity) transformations when the group acts on a collection of objects. Since Lie groups naturally describe continuous symmetry operations (operations that preserve features of an entity or structure), they generally yield structure-preserving transformations. Thus, the Lie algebras of Lie groups give us an example of continuous, exactly structure-preserving near-identity transformations.

For a contrary case of discrete, approximately structure-preserving transformations, consider the case of stable numerical methods for differential equations. Differential equations generally specify how entities in some state space evolve over some continuous transformation. Indeed, if we know the operator that transforms the state space in accordance with the differential equation, we have solved the differential equation.[5] It is very hard, in general, however, to compute the operator that solves the equation. Consequently, it is very useful to approximate this operator by considering small discrete changes that correspond to the continuous changes specified by the differential equation. This is the strategy followed by numerical methods, which consist of a set of (difference) equations that determine a discrete near-identity state transition map on the state space. These difference equations can then be evaluated or approximated by a computer. Since the discrete operations break the structure of the differential equation, they can only produce approximate solutions to the equation. For a stable method, the smaller the discrete change (*e.g.*, time step or interval of a regular spatial mesh), the more accurate the approximation the numerical method yields. Thus, the state transition maps of stable numerical methods give us examples of discrete, approximately structure-preserving near-identity transformations.

Leaving aside for the moment the kind of transformation involved in near-identity variation, let us turn to consider the *inferential* relation articulated by an effectively valid consequence relation. The idea is that nearly-identical inputs (premises, assertions, assumptions) yield nearly-identical outputs (conclusions, consequences). In other words, a “small” change to the input results in a “small” change to the output. This is thus very similar to the relation that a continuous transformation or map must have, where infinitesimal changes to the input result in infinitesimal changes to the output. Thus, an effectively valid consequence relation can be understood as being analogous to considering “continuous” maps between premises and their conclusions. The word ‘continuous’ is in double-quotes because effectively valid inferences are not always continuous in the mathematical sense,[6] since changes can be discrete, but they do exhibit a generalized

[5]Given this, it should not be surprising that the theory of Lie groups and Lie algebras is useful in the theory of differential equations.

[6]This is so even for the sense of continuous map from general topology, despite apparent similarities,

kind of continuity based on the nature of the near-identity relation that is preserved by the inference.

Given the analogy between effectively valid inference and continuous maps, which are stable under infinitesimal variations of the input, we can consider effectively valid inferences to be stable under "micro-local" variations of their premises. All that is meant by the phrase "micro-local variation" is a near-identity variation, which is what effective validity guarantees stability for. In such a case we fix some particular entity as centre and smear it out with near-identity transformations. Though micro-local variations have a fixed centre, in general we are also interested in the stability of inferences for "large" changes to the input (premises, assertions, assumptions, *etc.*). For such "large" changes, the centre of the variation itself moves. When we move the centre of variation of the premises of an effectively valid inference there is no guarantee that we move to another effectively valid inference. This is to say that generically effectively valid inference forms are only *locally stable*, *i.e.*, they are stable only locally to a certain scope of "macro-local" variation. Thus, outside of this scope of variation, effectively valid inference forms will become unstable, in one of the ways described above. In certain special cases an inference form will remain stable for all possible variations (of a given type), in which case the inference form will be called *globally stable* (relative to the type of variation).

One of the natural places to look for examples of globally stable inferences is abstract algebra. Since algebraic theories pertain to a given class of structures that are defined (or definable) axiomatically, a standard way of studying a given kind of structure is to study the structure-preserving maps between structures satisfying the axioms. Such maps are called *homomorphisms*, connoting "same structure". If a homomorphism between objects is invertible, then the objects have identical or "equal" structure, and the map is called an *isomorphism*. Any inference that is valid purely in virtue of the structure of the objects appearing in the premises will continue to be valid if those objects (and corresponding ones in the conclusion) are replaced by isomorphic ones. Accordingly, the inference in question is globally stable under isomorphism transformations of the objects in the premises. From the perspective of effective logic, however, such a case is one where the micro expands to the macro, since all isomorphic transformations of the objects in the premises count as near-identity transformations, in which case the inference is really only a single isolated inference that maps between isomorphism classes of objects.[7] It is

since the kind of relation specified by an effectively valid inference is intended to be considerably more general, applying to inferences outside of mathematics.

[7]Note that even with isomorphism classes we do have a notion of *near*-identity, not identity *simplicity*,

essentially for this reason that theorems in abstract algebra tend to focus on so-called *universal* properties, *i.e.*, those properties that hold for all objects of a given structure or type.[8]

Examples of this phenomenon also obtain in analysis, however, as the following example shows. Consider a meromorphic function $f(z)$ on the complex plane that has a single isolated simple pole. Then consider a two distinct points c_1 and c_2 in the vicinity of the pole. It is a well-known theorem from complex analysis that contour integrals exhibit a form of path-independence. According to this, the truth of the sentence

$$\int_{\mathcal{C}} f(z)dz = c, \tag{4.1}$$

where $c \in \mathbb{C}$ and the endpoints of $\mathcal{C}$ are c_1 and c_2, is invariant under which $\mathcal{C}$ we choose *provided* that for any two contours $\mathcal{C}_1$ and $\mathcal{C}_2$ with endpoints fixed at c_1 and c_2 are homotopic to each other in the sense of a continuous deformation that does not go through the pole. Thus, the truth of (4.1) is stable (invariant) under micro-local change of the contour, and the local stability (invariance) expands to the global within the equivalence classes defined by homotopic contours. Thus, the continuous variety of true sentences of form (4.1) reduces to a discrete set of isolated propositions corresponding to the equivalence classes. Indeed, if we turn the set of isolated propositions into a group with the law of composition determined by the addition or subtraction of loops around the pole, then group of propositions is isomorphic to the fundamental group of the punctured plane.[9]

Another place to look for examples of globally stable inferences is formal logic. In the case of categorical theories, such as the second order theory of the real numbers as the order-complete totally ordered field, all of the models of the theory are isomorphic. If we

because isomorphism is always identity *relative to a type*. Saying two object are isomorphic is only non-trivially meaningful if the two objects have some other structure that distinguishes them.

[8]Note that this extends to theorems with exceptions where the exceptions determine a substructure or subtype over which the result applies universally.

[9]There is a clear connection here to the ideas of homotopy type theory (HoTT), which is an extension of Martin Löf type theory that adds identity types for collections of isomorphic objects, allowing isomorphic objects to be treated as formally identical. HoTT is based on Voevodsky's discovery of a model of type theory in abstract homotopy theory. See [15] for a general introduction to the foundations of the theory. The concept of identity in HoTT has a clear relation to near-identity transformations that exactly preserve the structure of sentences, and exactly preserve truth. Thus, the form of inference is still deductive, as we expect from a logical foundation for pure mathematics. The notion of identity in HoTT does not, therefore, natively capture approximate near-identity or preservation of near-truth, as is required for effective logic. The similarity of the approaches, however, makes the relationship of HoTT and effective logic and interesting subject for future research.

consider (logical) structure-preserving transformations between models of a categorical theory, then any theorem will be globally stable under this class of transformations. On the other hand, from the perspective of effective logic, all of the models are accessible by a near-identity transformation (since all of the transformations are isomorphisms and thus strictly identity preserving), in which case there is really only a single isolated model of the theory. For a non-categorical theory, however, such as the first order theory of the real numbers as a real closed field, not all of the models of the theory are isomorphic. In this case, only certain (logical) structure-preserving transformations between models will preserve stability, *i.e.*, truth in the model. If one considers theorems of the formal theory, they will be globally stable by definition, but other inferences that are valid in some models will only be stable under some transformations between models. Thus, non-categorical theories have statements whose truth is only locally stable.

We see from these last two examples, that where the near-identity variation is generated by isomorphisms, the ranges of nearly-identical entities tend to have sharp boundaries. Indeed, they must in some manner because being related by an isomorphism is an equivalence relation, *i.e.*, a reflexive, symmetric and transitive relation. This shows that interpreting near-identity as isomorphism leads to a strong sense of near-identity and to inferences that are trivial from the perspective of micro-local stability (tend to reduce to a single isolated proposition).[10] Approximations, on the other hand, do not have such strict requirements on structure-preservation, leading to a weaker notion of near-identity transformation involving near-preservation of structure, in a contextually relevant sense of 'near'. To fix language, and to distinguish such maps from those of algebra, we could call a near-structure-preserving map a *continomorphism*, connoting "near structure".[11] We here adapt the modern Greek word *κοντινός*, the adjectival form of 'near' or 'close'. Such a map need not be micro-locally invertible, much like a homomorphism need not be invertible. Thus, we can introduce the term *contisomorphism*, connoting "near equal structure", for an invertible near-structure-preserving map. It is (micro-local) transformations of this kind that generate the near-identity transformations we have been discussing.[12]

[10]Once again, we see the connection between near-identity as isomorphism and the concept of identity types HoTT.

[11]This terminology has the nice property of being bringing to mind the word 'continue', which associates a meaning of "continuing" a structure, a natural idea of "continuous" variation. This is appropriate given that effective validity is articulating a generalized concept of continuity for inferences.

[12]Note that effective validity itself can be understood in terms of continomorphisms. Though the ranges of nearly-identical sentences must be generated by cont*iso*morphisms, the relation between the premises and conclusion is only that of a continomorphism. This is so because effective validity tells us

A key property of contisomorphisms is that as relations between entities they are reflexive and symmetric but not transitive. Thus, unlike isomorphisms, they do not give rise to equivalence classes. This is what allows inferences that are micro-locally stable under contisomorphisms about the centre of variation to become unstable under contisomorphic motions of the centre itself. The idea is that reflexivity and symmetry are necessary for micro-local stability, since the inference must hold for the centre of variation and must continue to hold for any inference accessible by a near-identity contisomorphism from the centre. Transitivity, on the other hand, would imply that the inference continue to hold for any near-identity transformation applied to any inference away from the centre, which if iterated would imply global stability, which does not hold for approximate inferences in general.

To illustrate these ideas we will consider an example of near-identity variations of symbolic expressions and mathematical objects, leaving the case of inference for the next section. For a simple syntactic example, consider the case of bivariate polynomials over the real numbers, *e.g.*, $P = xy^2 + 2x - y$. Traditionally such expressions are only identical if they are mathematically equivalent, in the sense that they are interchangeable by changing the order of the terms and the order of x and y. We can introduce a syntactic form of variation centred at P by allowing small changes to the coefficients of the monomial terms, so that, *e.g.*, each $P_\delta = (1+\delta)xy^2 + 2x - y$, $\delta > 0$, will be nearly-identical to P for $|\delta|$ sufficiently small. This is often done by introducing some *tolerance* $\varepsilon > 0$ such that for changes to the coefficients less than (or equal to) ε in size, the resulting expressions would be effectively equivalent. Thus, a single expression is expanded to a continuous range of nearly-identical expressions defined by changes of coefficients within the tolerance.

An example of a near-identity contisomorphism (NIC), or a micro-local change in expression, in this context is a map from P to an element of the set $\{P_\delta \,|\, |\delta| \leq \varepsilon\}$. Notice that the identity map is included in this since $P \mapsto P_0$ is included in the NICs, so being related by a NIC is a reflexive relation. It is also symmetric, since for any $|\delta| \leq \varepsilon$, there is a NIC from P to P_δ, and a map from P_δ to P is included among the NICs for near-identity variation with P_δ as the centre. It is not, however, the case that being related by a NIC is transitive, since there are many NICs centred at P_δ that are not accessible to NICs centred at P, such as $P_{\delta+\varepsilon}$.

This shows both how approximation can be introduced in purely a purely syntactic

that for any near identical transformation of the premises, *some* near identical conclusion will follow, but it does not say that *any* near identical conclusion follows from a near identical premise.

context and illustrates the non-transitivity of near-identity transformations. Though this example is continuous, there are many other kinds of near-identity transformations that are discontinuous. For example, consider nearly synonymous sentences. We could specify a range of sentences nearly-identical to a given sentence as centre based on nearly-identical meaning, say as judged by some well-trained deep learning algorithm. Once defined, the range is just a collection of sentences, thereby purely syntactic, and the NICs are just the maps from the centre sentence to the members of the range.

There is nothing about the general concept of near-identity transformation that requires the changes to be small in a sense that we might find intuitive. Keep in mind that the basic theme is stability under variation, and near-identity maps allow us to track what is judged in the context to be "nearby". Another kind of example that fits into this frame is the stability of mathematical theorems under change in mathematical context. In this case, what counts as a near-identity transformation may be highly discrete, such as a change in dimension or kind of domain. This allows us to think of the stability of a theorem in terms of how much or how easily (in the sense of how much its formulation has to change) it generalizes. An example we might consider is the fundamental theorem of calculus, which, starting from functions of a single real variable, is stable under a wide range of variations that preserve its basic content, extending to complex variables, vector variables, different kinds of integration (line, surface, *etc.*) and to curved spaces (manifolds).

This shows a variety of ways in which approximation can be introduced in purely syntactic contexts. It is perhaps more obvious how approximation may be introduced in purely semantic contexts. Thinking of semantics in terms of objects in contrast to a symbolic syntax, then it is natural to think of near-identity in terms of indistinugishability. Just as before, we have two distinct kinds of variation. The first kind is exactly indistinguishable objects, in the sense of having no distinguishing properties according to some standard.[13] A nice example of this is identical particles in quantum mechanics. This case corresponds to isomorphism, since the indistinguishability relation is an equivalence relation. But we can also have approximately indistinguishable objects, which are micro-locally indistinguishable but over wider variations can be distinguishable. Though not necessarily picking out "objects", a nice example of this is colours, for which close-by colours can be indistinguishable but are distinguishable over larger changes. This case

[13]Note the connection between exact indistinguishability and the general concept of identity from type theory, *viz.*, identity is always relativized to a particular type, where from a structural perspective all instances of the type have identical structure.

corresponds to contisomorphism, since the near indistinguishability relation is reflexive and symmetric but not transitive.

It should be natural enough to see how small changes to an object treated as nearly-indistinguishable give rise to near-identity transformations of objects. For a mathematical example, we can consider the geometric objects corresponding to bivariate polynomials, namely one dimensional polynomial varieties.[14] These algebraic varieties are curves in the plane that correspond to the zero-sets of bivariate polynomials, *i.e.*, for a polynomial $P(x,y)$, the locus of points in the plane that satisfies the equation $P(x,y) = 0$. We will restrict ourselves to real varieties, *i.e.*, where we only consider real valued solutions to the polynomial equation. Traditionally such geometric objects are uniquely given and do not admit of variation. We may suppose a context, however, where small variations of the curve are acceptable and can be treated as effectively the same curve. For a formal condition, we could require that the maximum separation of the two curves is less than a tolerance ε in order to be considered nearly-identical. Thus, a single curve is expanded into a continuous family of nearly-equivalent curves defined by variations of the curve within the tolerance. This kind of approximate identity is essentially similar to the colours example above, since sufficiently close curves are nearly-indistinguishable but beyond the tolerance differences are noticeable and near-identity fails.

One may see that allowing entities to vary by near-identity transformations can lead to near-identity variations of properties or relations of such objects. In the case of polynomial varieties, an example of a relation is $P(x,y) = 0$. To make the assumption that nearby curves (within the tolerance ε) are nearly-identical coherent with the relation $P(x,y) = 0$, we must allow this relation to be satisfied only approximately for each point on a nearly-identical curve. This leads to the relation $P(x,y) = 0$ being smeared out into a range of nearly-identical relations, *e.g.*, $S = \{P(x,y) - \delta = 0 \,|\, |\delta| \leq \varepsilon\}$. According to the standard specified by the tolerance ε, then, having a curve that satisfies any of the equations in the set S would allow us to say in a precise sense that the relation specified by $P(x,y) = 0$ is approximately satisfied. An effectively valid inference based on the assumption of $P(x,y) = 0$, then, would yield some relation specified by q such that any relation in S yields some relation nearly-identical to q.

We now have a sense of how the notion of effectively valid inferences, as inferences

[14]Generally, the algebraic varieties of bivariate polynomials are geometric objects in what is called in algebraic geometry the *complex plane*, meaning $\mathbb{C}^2$, not the Argand plane of complex analysis, which is a geometric representation of $\mathbb{C}$. Since the algebraic complex plane is difficult to visualize, we restrict to real algebraic varieties, where the variables x and y can only take values in $\mathbb{R}$. Real algebraic varieties of bivariate polynomials are therefore curves in the real plane $\mathbb{R}^2$.

producing stable outputs (conclusions, consequences) under small variations of their inputs (premises, assertions, assumptions, satisfaction of relations), can be made precise in terms of near-identity transformations of sentences, both for purely syntactic sentences and for properties and relations of objects. We have not yet seen proper instances of effectively valid inferences. Showing how such inferences arise naturally when approximation methods are used in scientific practice is part of the focus of the next section.

4.3 Interpreted Languages and Synto-Semantic Stability

In scientific practice it is common to work with interpreted languages, *i.e.*, where we have a particular formal language and a well-defined intended interpretation. So when approximation is introduced in such cases, we need in general to simultaneously track variations of the syntax and semantics. Since here we have interpreted linguistic forms, we are in a context where we can talk properly about near-truth. A natural way to think about this is the following. Suppose that we begin with an individual sentence p, which has a unique fixed meaning m. We may suppose that, with this fixed meaning m, p is true *simpliciter*. We then introduce approximation in the semantics, so that the unique fixed meaning m is smeared out micro-locally into some collection of nearly-identical meanings $\mathcal{M}$. Then, if any $m' \in \mathcal{M}$ is picked out by p then p is *nearly true*. It should be evident that the basic structure of introducing approximation has nothing to do with sentences, and that a more generalized kind of synto-semantic approximation is possible for any syntactic form and corresponding fixed reference.

What we have so far is only half of the story, however, since we also have approximation at the syntactic level. Thus, the individual sentence p is also smeared out into a collection of nearly-identical sentences $\mathcal{P}$. The idea is that if p is nearly true, then so should be any of the nearly-identical sentences in $\mathcal{P}$. For this to be coherent with the semantics, it must be the case that for $p, p' \in \mathcal{P}$, $\mathcal{M}_p \cap \mathcal{M}_{p'} \neq \varnothing$. Notice that it matters which syntactic form was originally judged to be nearly true, *i.e.*, which is the centre of micro-local variation, since the property of witnessing near-truth is not transitive, for essentially the same reasons that near-identity transformations are not transitive. For instance, whichever $m' \in \mathcal{M}_p$ that witnesses the near-truth of p should witness the near-truth of any $p' \in \mathcal{P}_p$, but need not witness the near-truth of all $p'' \in \mathcal{P}_{p'}$, the collection of forms equivalent to p', as the following concrete example will illustrate.

Consider once again the bivariate polynomial varieties example. Suppose that in this case that a given polynomial equation $P(x,y) = 0$ is *true simpliciter* if we find a point

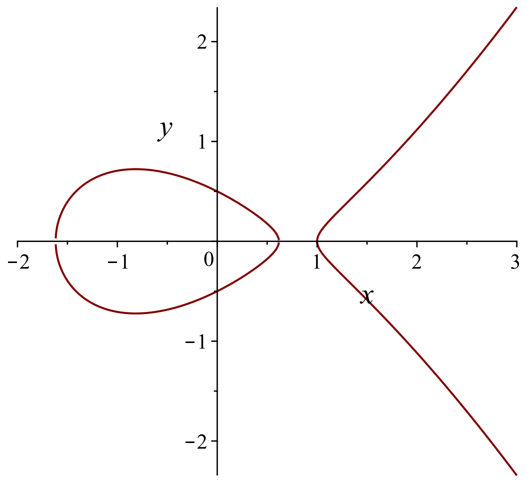


Figure 4.2: An elliptic curve, the real algebraic variety of the polynomial $2y^2 - x^3 + 2x - 1$.

(x^*, y^*) such that $P(x^*, y^*) = 0$ exactly, but that the same equation is *nearly true* if we find a point (x^*, y^*) such that $|P(x^*, y^*)| \leq \varepsilon$ for some small $\varepsilon > 0$. For concreteness, suppose we have a polynomial $P(x, y) = 2y^2 - x^3 + 2x - 1$, whose locus of real zeros picks out an elliptic curve in the plane (see figure 4.2), and that our tolerance is $\varepsilon = 0.001$. Thus, any point in the plane that satisfies the equation $P(x, y) = 0$ with an error within 0.001 is a witness to the near-truth of $P(x, y) = 0$.

Consider the point $(x, y) = (0.618, 0)$. When we substitute this into the expression for P we find that $P(0.618, 0) = -0.000029032$, so that since $|-0.000029032| < 0.001$, picking this point makes $P = 0$ nearly true. It turns out that this point is so close to the curve because the golden ratio $\frac{1}{2}(\sqrt{5} - 1)$ picks out one of the points on the x-axis.

The simplest, though by no means the only, way to introduce equivalent expressions compatible with the approximate semantics in this case is to say that any polynomial P' otherwise identical to P but with a constant term that is within ε of -1 (inclusive) is equivalent to P. Then, the polynomial equation $P'(x, y) = 2y^2 - x^3 + 2x - .999970968$ is equivalent to that of P, which the point $(0.618.0)$ now satisfies exactly. The polynomial equation $P''(x, y) = 2y^2 - x^3 + 2x - 1.001$ is also equivalent to that of P given the definition. But if we substitute the point $(0.618, 0)$ into the equation $P''(x, y) = 0$ we find that it does not witness the near-truth of $P'(x, y) = 0$ according to our standard. This is because the point $(0.618.0)$ witnesses the truth simpliciter of $P'(x, y) = 0$ and the near-truth of $Q(x, y) = 0$ for any Q that differs from P' within the tolerance, which includes the original P. It does not include P'', however, since it differs from P' by $1.001029032 > \varepsilon$. Thus, P'' is not equivalent to P', so witnesses to the near-truth of P' need not witness the near-truth of P''. This shows how witnessing near-truth is not transitive.

This matter of syntactic and semantic approximation being compatible is essential for being able to move back and forth between syntactic and semantic reasoning in an interpreted language. It is a property that languages in applied mathematics commonly have, particularly in well-developed areas where the semantics is well-understood. The compatibility of syntax and semantics is actually another stability property, this time relating syntax and semantics. There are actually two stability properties relating syntax and semantics of an interpreted language. One says that approximations in the syntax lead to approximations in the semantics, which we may call *semantic stability* of the syntax. The other says that approximations in the semantics lead to approximation in the syntax, which we may call *syntactic stability.* Both of these two properties holding together, which we will assume for interpreted languages, may be called *synto-semantic stability* of an interpreted language.

That the two conditions (semantic and syntactic stability) are not equivalent can be made clear in terms of the concept of a continomorphism. In fact, the conditions both imply that the map is a continomorphism, one (semantic stability) from syntax to semantics and the other (syntactic stability) from semantics to syntax. Each condition on its own does not imply that the map is also a cont*iso*morphism, indicating invertibility. Thus, it is possible to have a semantically stable language, meaning that small changes to the syntax produce small changes in the semantics, have *some* semantic changes lead to large changes in the syntax. This is to say, that the condition of semantic stability does not imply that small changes of the syntax lead to *all* possible small changes of the semantics, only certain changes of the semantics. If the language is also syntactically stable, however, then any small change in the semantics leads to a small change in the syntax. Thus, the two conditions holding together, *i.e.*, when the language is synto-semantically stable, imply that the intepretation map for the language is a cont*iso*morphism.

It may appear that the relationship articulated by semantic and syntactic stability is related to a generalization of soundness and completeness. This is only an analogy, however, since the standard notions of soundness and completeness only make sense for *un*interpreted languages. Consideration of stability properties of variations of logical languages in relation to variations of the models would be the appropriate context to consider concepts of effective soundness and completeness. Since we are restricting our attention to interpreted languages for the remainder of this paper, we will not consider soundness and completeness here.

The concept of synto-semantic stability as yet has nothing directly to do with stability of *inference*. It is natural to introduce this through a simple example of scientific inference. To bring out the (effective) logical character of the inferences involved, and to fix concepts, we will introduce a special notation based on an extension of the single turnstile notation in standard logic. For strict syntactic consequence relations we will use the notation

$$\text{assumed sentences} \mathrel{\big|\!\underline{}_{\text{globally imposed sentences}}} \text{valid consequence},$$

so that the (local) assumptions appear on the left of the turnstile, valid consequences appear on the right, and any system-wide (structural) sentences, functioning as constraints, can be displayed underneath the horizontal stroke of the turnstile. For strict semantic consequence relations in an interpreted language we will use the special notation

$$\text{assumed relations} \mathrel{\big\|\!\underline{}_{\text{globally imposed relations}}} \text{valid consequence},$$

where the second vertical stroke indicates we are dealing with inferences from interpreted relations to other interpreted relations.[15]

The notation as it stands indicates an exact consequence relation (syntactic or semantic). Since our consideration of micro-local variation involves treating nearly-identical sentences and relations as effectively equivalent, we can consider being related by a near-identity contisomorphism (micro-locally) as an effective equivalence relation, which we will denote by $\sim$.[16] Thus, for a set of sentences Γ or their interpretations $[\![\Gamma]\!]$, with respective consequences, p or $[\![p]\!]$, and framework constraints, $\mathcal{F}$ or $[\![\mathcal{F}]\!]$, the effectively valid consequence relations can be denoted by

$$\Gamma \mathrel{\big|\!\overset{\sim}{\underset{\mathcal{F}}{\underline{}}}} p, \qquad [\![\Gamma]\!] \mathrel{\big\|\!\overset{\sim}{\underset{[\![\mathcal{F}]\!]}{\underline{}}}} [\![p]\!].$$

Since we are working with a fixed interpretation for an interpreted language and the

[15]This may seem like an odd notation, but this notation has been used before in the interest of specifying a notion of semantic consequence for dynamical systems by [12] for what he calls "semi-interpreted languages". Such notation therefore has a precedent in philosophy of science. Though the notation overlaps with that for forcing, it should be clear from the context what is the intended meaning.

[16]Note that in the case of continuous variation within some range, within that range transitivity *does* hold. Thus, near-identity transformations give rise to micro-local effective equivalence relations. This can be seen as analogous to the local flatness of smooth spaces.

double vertical stroke makes it clear that we are dealing with semantic consequence relations, we will generally omit the interpretation notation $[\![\cdot]\!]$ and simply write

$$\Gamma \mathrel{\|\!\!\!\approx_{\mathcal{F}}} p$$

for an effectively valid semantic consequence.

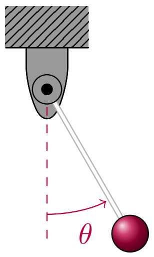


Figure 4.3: The simple pendulum model. It has one configurational degree of freedom specified by the angle θ. Together with the corresponding momentum degree of freedom ℓ_θ of the weight, the system has a two dimensional phase space.

For a simple example of scientific inference let us now consider a simple mechanical system, the simple pendulum, composed of a single massive weight connected to a massless rod frictionlessly connected to a pivot (see figure 4.3). For a rod of unit length, this system specifies a simple differential equation of motion

$$\ddot{\theta} + g \sin\theta = 0, \tag{4.4}$$

where dots denote time derivatives, θ denotes the angle made by the rod relative to the vertical, and g is the gravitational acceleration in units of inverse time squared. It is deceptively simple, however, because it is nonlinear (due to the presence of $\sin\theta$) and so cannot be solved by the usual methods presented in undergraduate differential equations courses, which (for second- and higher-order) work for linear equations. For this reason, one often seeks an approximate solution to this equation by considering the case of small oscillations. In the case that we assume θ is sufficiently small, we can conclude on syntactic grounds that the $\sin\theta$ term can be replaced with its linear approximation θ. This is an instance of the standard technique of linearizing the equation, which replaces a function with its linear Taylor approximation. The result of doing this is that we obtain the equation for the simple harmonic oscillator

$$\ddot{\theta} + g\theta = 0, \tag{4.5}$$

which has circular (sine and cosine) functions as solutions. Treating the original differential equation as an assumption, we add the assumption that the angle measured in radians is small, syntactically $\theta \ll 1$, which corresponds to the condition that the angle remain much smaller than $57.3°$. Under these conditions, the simple harmonic oscillator equation of motion (4.5) is effectively equivalent to the equation for the simple pendulum (4.4). Thus, we have our first example of an effectively valid inference:

$$\ddot{\theta} + g\sin\theta = 0, \theta \ll 1 \mathrel{|\!\!\overset{\sim}{-}} \ddot{\theta} + g\theta = 0.$$

Given the potential for instability of effectively valid inferences, we could not necessarily expect approximate solutions to (4.5) to provide a good approximate solution to (4.4), but if we obtain an *exact* solution to the simple harmonic oscillator, then we should expect a good approximation to the simple pendulum for small angles, which is part of what this statement says. Indeed, this is the reason why we linearize, to obtain a locally valid approximation to the original equation.

Consider, then, the simpler equation (4.5). To produce a unique solution we must supply some other condition, such as an initial condition. We may take $\theta(0) = \theta_0$ as an initial angle and take $\dot{\theta}(0) = 0$ as the initial angular velocity. In this case, we can solve (4.5) to obtain $\theta(t) = \theta_0 \cos(\omega t)$, where $\omega = \sqrt{g}$, which is straightforward to verify using purely syntactic calculations. We can also express this in our logical notation. Suppose that we now work in a framework where the differential equation (4.5) is imposed as a global constraint, then we can express the exact solution to the initial value problem as

$$\theta(0) = \theta_0, \dot{\theta}(0) = 0 \mathrel{|\!\!\underset{\ddot{\theta}+g\theta=0}{\rule{2em}{0.4pt}}} \theta(t) = \theta_0\cos(\omega t),$$

where we now drop the $\sim$ to indicate the fact that the consequence relation is exact. Thus, this expresses that the solution curve $\theta(t) = \theta_0\cos(\omega t)$ is a valid consequence of the initial conditions $\theta(0) = \theta_0$ and $\dot{\theta}(0) = 0$.

Combining this with our conclusion that the simple harmonic oscillator is an effectively valid approximation of the simple pendulum for small angles, we can shift to a framework in which the differential equation (4.4) is imposed as a global constraint, in which case we can express the approximate solution to the corresponding initial value problem as

$$\theta(0) = \theta_0, \dot{\theta}(0) = 0, \theta_0 \ll 1 \mathrel{|\!\!\underset{\ddot{\theta}+g\sin\theta=0}{\overset{\sim}{\rule{3em}{0.4pt}}}} \theta(t) = \theta_0\cos(\omega t),$$

which follows since if the initial angle $\theta_0 \ll 1$ then $\theta(t) \ll 1$ for all times t. Note that this consequence relation only holds locally to a certain time range $t \in [0, T(\theta_0)]$, for some function T, outside of which the inference becomes unstable. This shows how effective logic captures the standard approach of linearizing the simple pendulum.

All of the reasoning so far has been purely syntactic. Thus, we are assuming that the approximation arguments we used syntactically will yield valid approximations in terms of the interpretation, that is to say we assume that semantic stability is preserved when we make approximate inferences. This should imply that the smaller the initial angle, the closer the exact solution of the simple pendulum approaches the corresponding simple harmonic oscillator solution. Indeed this is the case, but showing that this is so is not straightforward. Before we examine how the semantic approximation corresponding to our syntactic argument works, let us first consider how to treat the semantics using our notation.

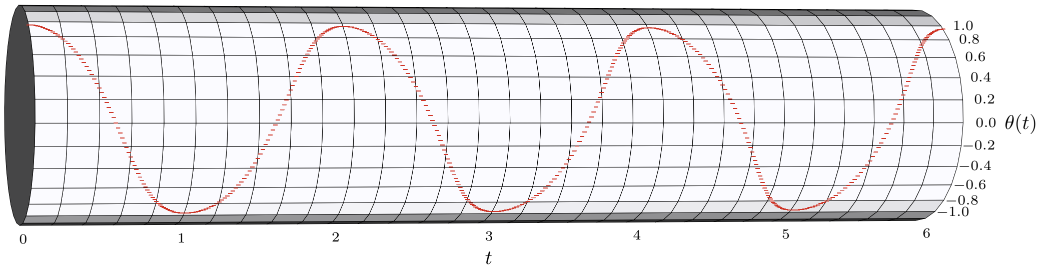
The configuration space is particularly simple for this system, since the mass of the pendulum is restricted to move on a circle of radius equal to the length of the rod, which we have assumed is length 1 for simplicity. Thus, for a description of the motion of the system it is enough to specify the angle $\theta(t)$ as a function of time. For definiteness, we can picture the state space as a cylinder of radius one, where the height is the time and the angular position is θ. The motion of the pendulum then traces out a unique curve on the cylinder over time. The scientific problem to be solved is to specify this curve for any given initial state the pendulum is in.

In this case, considering equation (4.4) as a model of a real pendulum, it is clear why approximation is acceptable, since there is error involved in the construction of the model, so error in its solution can be acceptable provided it does not interfere with the applicability of the result. Thus, we are interested in any curves that stay sufficiently close, say within any experimental error, to the model solution curve for a reasonable period of time. This introduces a near-identity condition very similar to the one introduced in the previous section for polynomial varieties. More specifically, we may require that any acceptable solution $\varphi(t)$ differ from the exact solution by no more than some tolerance ε, *i.e.*, so that $|\theta(t) - \varphi(t)| \leq \varepsilon$. It is such a condition that can only be satisfied for a limited period of time when approximation is allowed.

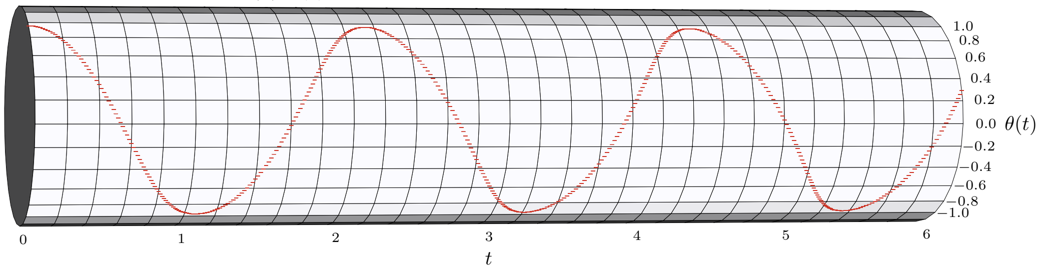
Consider now the nonlinear equation (4.4) for the simple pendulum. What makes this equation different from most nonlinear equations encountered in practice is that it can be solved exactly. For the same initial condition we considered above this equation can be solved in terms of the Jacobi elliptic function $\mathrm{sn}(z, k)$, where z is a complex variable and $k \in [0,1]$ is a parameter. For our purposes we do not need to know much about this function, the following two properties are enough: (1) it is periodic along the real line, *i.e.*, for z real; and (2) in the limit $k \to 0$, $\operatorname{sn}(z,k) \to \sin z$, so it asymptotically approaches the sine function as the second variable k goes to zero. The solution of (4.4) can be expressed in the form $\sin(\theta(t)/2) = A\operatorname{sn}(\omega t + K, A)$,[17] where K is the quarter period and $A = \sin(\theta_0/2)$ contains the initial angle [5]. We can write this in symbols as

$$\theta(0) = \theta_0, \dot{\theta}(0) = 0 \Vdash_{\overline{\ddot{\theta}+g\sin\theta=0}} \sin(\theta(t)/2) = A\operatorname{sn}(\omega t + K, A),$$

noting that this is an exact consequence relation and (4.4) is imposed as a global constraint.



(a) $\theta(t)$ for the simple harmonic oscillator with $\theta_0 = 1$.



(b) $\theta(t)$ for the simple pendulum with $\theta_0 = 1$.

Figure 4.6: The simple harmonic oscillator compared to the simple pendulum for a moderate initial angle. Grid lines are separated by 0.2 radians around the cylinder and 0.2 s along its length. The amplitudes match, but the frequencies are distinct, and the solutions diverge noticeably after only about 0.3 s.

To get a sense for the behaviour of this solution, figure 4.6 compares the behaviour of the simple pendulum and simple harmonic oscillator solutions for a moderate initial angle $\theta_0 = 1$. It is evident for such a large initial angle the solutions are very different, and significantly so after 6 s. We see that the amplitudes of the two solutions are the same, as they should be since they both describe a conservative pendulum system dropped

[17]The solution can be written as a function of θ simply by rewriting it in the form $\theta(t) = 2\sin^{-1}(A\operatorname{sn}(\omega t + K, A))$.

from rest at the same initial angle. The frequencies are quite different, however, and a noticeable difference is observable on this plot after about 0.3 s.

Now, if we restrict ourselves to small angles, so that $\theta \ll 1$ rad ($\theta \ll 57.3°$) as before, then $A \approx \theta_0/2$ and $\sin\theta \approx \theta$, so that the expression

$$\sin(\theta/2) = A\,\mathrm{sn}(\omega t + K, A)$$

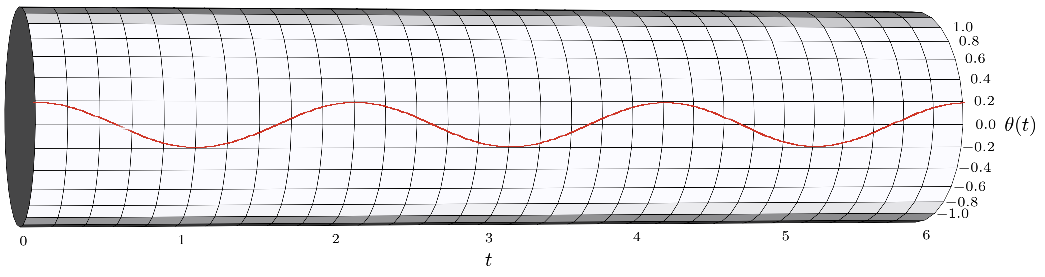
reduces to

$$\theta(t) = \theta_0\,\mathrm{sn}(\omega t + K, \theta_0/2).$$

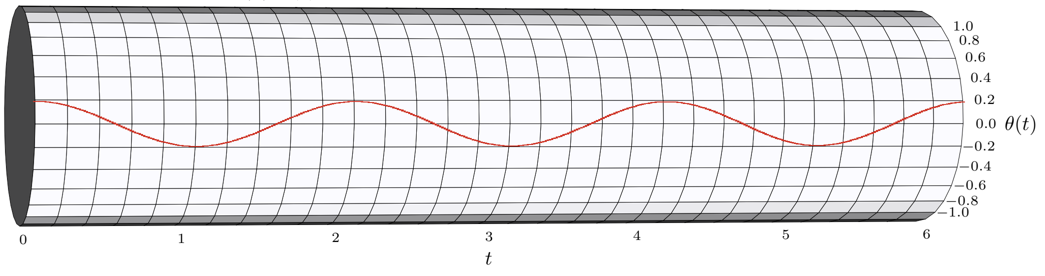
Furthermore, since θ_0 is very small, corresponding to the regime where the parameter $k = \theta_0/2 \to 0$, the elliptic function $\mathrm{sn}(x + K, k)$ approaches $\sin(x + \pi/2)$ (sine advanced by a quarter period).[18] Since $\sin(x + \pi/2) = \cos(x)$, we therefore obtain in the limit of small initial angles,

$$\theta(t) = \theta_0 \cos(\omega t),$$

the solution of the simple harmonic oscillator.



(a) $\theta(t)$ for the simple harmonic oscillator with $\theta_0 = 0.2$.



(b) $\theta(t)$ for the simple pendulum with $\theta_0 = 0.2$.

Figure 4.7: The simple harmonic oscillator compared to the simple pendulum for a small initial angle over the time interval $[0, 6]$. Grid lines are separated by 0.2 radians around the cylinder and 0.2 s along its length. The phase and amplitude match closely, with only a small error discernable after 6 s.

Thus, for small initial angles, and sufficiently short times, the simple harmonic oscilla-

[18]This follows because $\mathrm{sn}(x, k) \to \sin(x)$ as $k \to 0$, and $K \to \pi/2$ because $\pi/2$ is the quarter period of the sine function.

tor solution is a good approximation to the motion of the pendulum. Recall our condition for an acceptable solution that it differ from the exact solution by less than some tolerance ε. We can then consider any two solutions $\theta(t)$ and $\varphi(t)$ to be nearly-equivalent, written $\theta(t) \sim \varphi(t)$ provided $|\theta(t) - \varphi(t)| \leq \varepsilon$, *i.e.*, provided they are within ε of each other on the solution cylinder. We therefore have established a semantic version of the syntactic consequence relation we established earlier using a linearization argument:

$$\theta(0) = \theta_0, \dot{\theta}(0) = 0, \theta \ll 1 \mathrel{\|\!\!\overset{\sim}{\underset{\ddot{\theta}+g\sin\theta=0}{\models}}} \theta(t) = \theta_0 \cos(\omega t).$$

We have therefore shown now in semantic terms that the simple harmonic oscillator solution is an effectively valid solution to the simple pendulum for small angles, a statement that is true for sufficiently short time scales. This result is illustrated in figure 4.7, which shows how for an initial angle of $\theta_0 = 0.2$ the solutions of the simple pendulum and simple harmonic oscillator are nearly-indistinguishable through 6 s.

The idea behind the condition of synto-semantic stability for an interpreted language is that the syntax and semantics *covary*, so that changes to one are reflected in the other. This property is essential so that methods, such as linearization, that can be applied purely syntactically lead to stable results in the interpretation. Thus, syntactic variations are carried forward to semantic variations for a synto-semantically stable language. Though some care is needed to judge the scope of stability of approximations, we expect interpreted languages to have the property that stable syntactic inferences result in stable semantic ones, *i.e.*, that

$$\Gamma \overset{\sim}{\underset{C}{\vdash}} p \Longrightarrow [\![\Gamma]\!] \mathrel{\|\!\!\overset{\sim}{\underset{[\![C]\!]}{\models}}} [\![p]\!].$$

We may call this property of an effectively valid syntactic inference a *semantically stable consequence*. Conversely, we expect that stable semantic variations will result in stable syntactic ones, so that they can be proved in the language, so that

$$[\![\Gamma]\!] \mathrel{\|\!\!\overset{\sim}{\underset{[\![C]\!]}{\models}}} [\![p]\!] \Longrightarrow \Gamma \overset{\sim}{\underset{C}{\vdash}} p.$$

This property of an effectively valid semantic inference may be called a *syntactically stable consequence*.

The advantage of having a language with both of these properties, where we would say that consequence relations are *synto-semantically stable*, is that we can reason stably using both syntactic and semantic arguments, which is common in mathematical practice. It is this stability property that takes the place of soundness and completeness for interpreted languages. Since it is a property that is possessed or assumed in many scientific languages, we will use a special notation for synto-semantically stable consequences in interpreted languages, namely

$$\Gamma \,|||\!\!\overset{\sim}{\underset{\mathcal{C}}{}}\, p,$$

where we have introduced a third vertical stroke (one for syntax, two for semantics) to indicate that the consequence is both syntactically and semantically effectively valid in the interpreted language. Our first example of such an effectively valid synto-semantic inference is what we established above, *viz.*,

$$\theta(0) = \theta_0, \dot{\theta}(0) = 0, \theta \ll 1 \; |||\!\!\overset{\sim}{\underset{\ddot{\theta}+g\sin\theta=0}{\overline{}}}\; \theta(t) = \theta_0 \cos(\omega t).$$

This can be interpreted as asserting the (effective) commutativity of interpretation and inference.

We have seen in this section how the concept of (syntactic, semantic and synto-semantic) effective validity captures the idea of approximate solutions to differential equations for interpreted languages. The applicability of the concepts of effective validity does not, however, rely on anything specific to differential eqautions. Any problem that can be expressed in terms of some input assumptions and an output solution can be treated in the same way. The case of deductive validity captures any instances where exact solutions are sought, and effective validity captures cases where some standard for approximate solutions is introduced. The analogue of an inference rule in this case is any operation that produces a solution to a problem given appropriate input. Thus, the nature of scientific inference in a problem solving context is seen to be highly analogous to logical deduction. Indeed, in the case of exact solutions, it is formally equivalent to deduction. When we introduce approximation, as is very common in scientific practice, analogue in effective logic is then *nearly deductive* effectively valid inferences.

4.4 Effective Logic and Scientific Problem-Solving

We have now seen how basic scientific problem-solving using approximation can be treated in terms of effectively valid inference. We will see in this section how more complex problem-solving methods can be treated in terms of effective logic. It is very common in science to have a problem that cannot be solved exactly in the originally-posed form. As was mentioned in the previous section, this is generally the case for nonlinear differential equations, which is the typical case for models in applied mathematics that are accurate for a wide range of conditions. In such situations, one generally resorts to some form of approximation. This typically requires some kind of modification of the problem itself, a shift to an analogous problem that is easier to solve. Indeed, this is actually what was done with the use of linearization to solve the simple pendulum problem approximately, we shifted from a nonlinear problem to a linear one, which was easier to solve. This strategy of transforming a difficult problem into an easier one in the search for an approximate solution is a very common one in scientific practice, and underlies strategies of computational complexity reduction in computational science, as was pointed out above. We will see in this section that this method can be understood in terms of the stability of inferential relations for mappings between interpreted languages.

Moving to the next more complicated problem from the simple pendulum takes us to the double pendulum, where we simply add another rod and weight to the simple pendulum. The result is a simple looking system that exhibits surprisingly complex behaviour; indeed, the double pendulum is chaotic for some initial conditions. The double pendulum is a simple example of a nonlinear differential equation for which we do not know a class of special functions that solve it analytically, unlike the simple pendulum that can be solved with Jacobi elliptic functions. This means we need to use other means to describe the behaviour of solutions of the equation. A standard approach here is to use numerical methods and computation to solve the equations approximately. We will see in this section how the approach of solving equations by numerics can be understood in terms of effective logic.

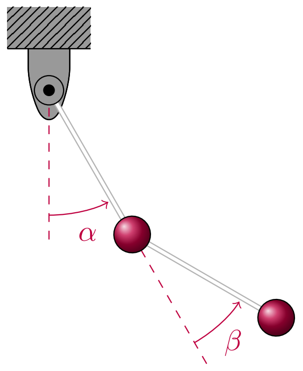


Figure 4.8: The double pendulum model. It has two configurational degrees of freedom, specified by the angles α and β. Together with the corresponding momentum degrees of freedom ℓ_α and ℓ_β of the two weights, the system has a four dimensional phase space.

We begin in this case with the differential equation of motion for the double pendulum, which can be given in terms of Hamilton's equations

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}}, \tag{4.9}$$

with the generalized position $\mathbf{q} = (\alpha, \beta)$ composed of the two angles describing the state of the system (see figure 4.8) and the generalized momentum $\mathbf{p} = (\ell_\alpha, \ell_\beta)$ composed of the angular momenta of the two weights. It can be shown, according to a standard algorithm, that the Hamiltonian for the system is

$$H(\mathbf{q}, \mathbf{p}) = -2\cos\alpha - \cos(\alpha + \beta) + \frac{l_\alpha^2 - 2(1 + \cos\beta) l_\alpha l_\beta + (3 + 2\cos\beta) l_\beta^2}{3 - \cos 2\beta}. \tag{4.10}$$

If we suppose that we are interested in the case where the first weight is held at an initial angle of $\alpha_0 = \pi/2$ (90°), and the second weight is left hanging by gravity, corresponding to $\beta_0 = -\pi/2$, then the inferential problem we are faced with, analogous to the simple pendulum, is

$$\alpha_0 = \pi/2, \beta_0 = -\pi/2 \mathrel{\|\!\!\mid\!\!\approx_{\mathcal{H}}} \alpha(t) = ?, \beta(t) = ?,$$

where $\mathcal{H} = \left\{\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}}, \dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}}, H(\mathbf{q}, \mathbf{p}) = (4.10)\right\}$ specifies the framework constraints for our Hamiltonian system. This notation is intended to express that we seek an effectively valid solution for the given initial conditions that specifies the evolution of the angles α and β over time. Unlike for the simple pendulum, however, we have no way of obtaining a valid (or effectively valid) solution directly, so we must search for an approximate solution by *transforming* the problem.

Since we seek an approximate solution and the system is chaotic for these initial values, we cannot expect fidelity for a very long time given that small errors grow exponentially. We can control the error quite well, however, by using a specialized numerical method that preserves very closely the geometric structure of the problem, in this case the symplectic form on phase space defined by Hamilton's equations. Numerical methods that accomplish such near-preservation of geometric structure are called *geometric numerical methods*, and specifically *symplectic methods* in the case of the symplectic structure of Hamiltonian systems [2].

A simple example of a symplectic method that we can use for this problem is the Störmer-Verlet method, which replaces Hamilton's continuous-time differential equations

(4.9) with a pair of discrete-time difference equations

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2), \quad \mathbf{p}_{n+1} = \mathbf{p}_n - \frac{h}{2}(\mathbf{m}_1 + \mathbf{m}_2),^{19} \tag{4.11}$$

where $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{m}_1$ and $\mathbf{m}_2$ are given by the (semi-implicit) equations

$$\begin{aligned}\mathbf{k}_1 &= \frac{\partial H}{\partial \mathbf{p}}\left(\mathbf{q}_n, \mathbf{p}_n + \frac{h}{2}\mathbf{m}_1\right), \ \mathbf{k}_2 = \frac{\partial H}{\partial \mathbf{p}}\left(\mathbf{q}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2), \ \mathbf{p}_n + \frac{h}{2}\mathbf{m}_1\right), \\ \mathbf{m}_1 &= \frac{\partial H}{\partial \mathbf{q}}\left(\mathbf{q}_n, \mathbf{p}_n + \frac{h}{2}\mathbf{m}_1\right), \ \mathbf{m}_2 = \frac{\partial H}{\partial \mathbf{q}}\left(\mathbf{q}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2), \mathbf{p}_n + \frac{h}{2}\mathbf{m}_1\right),\end{aligned} \tag{4.12}$$

where h is the time-step and for the double pendulum H is given by (4.10) as before. Rather than being continuous curves $\mathbf{q}(t) = (\alpha(t), \beta(t))$, the solutions of the Störmer-Verlet equation are time series $\mathbf{q}_n = (\alpha_n(t_n), \beta_n(t_n))$, where $t_n = nh$. The idea is that the map $\varphi : (\mathbf{q}_n, \mathbf{p}_n) \mapsto (\mathbf{q}_{n+1}, \mathbf{p}_{n+1})$ on phase space that advances the system forward in time is very nearly a symplectic map, meaning that among other things energy is very nearly conserved over time. This gives the numerical method the ability to adequately control the error over extremely long times, which will give us decent performance on this chaotic problem.

To clarify the logic of this situation, observe that we are seeking to find an approximate solution to our problem by mapping the problem to a different problem in a different framework (difference equations) in a way that *nearly preserves* the structure of the original problem; this way, solutions to the new problem can give us approximate solutions to the original problem. Thus, rather than a near-identity transformation of entities or sentences, we are considering a near-identity transformation of a framework. If we obtain a solution to the transformed problem, we obtain it in the synto-semantics of the alternative framework, which here means that we get a discrete solution not a continuous one. Nevertheless, the nature of the near-structure-preservation guarantees that if we carry a solution to the Störmer-Verlet equation back to the framework of Hamilton's equations, we will obtain a sequence of solution points very close to the corresponding solution points of the original equation, *i.e.*, we will have an effectively valid solution to Hamilton's equations for a sequence of times t_n. We can recover an approximate solution for the intervening times using some form of interpolation, which can convert

[19]Using the first equation to illustrate, we can see that writing these equations in a slightly different form, $\frac{\mathbf{q}_{n+1}-\mathbf{q}_n}{h} = \frac{1}{2}(\mathbf{k}_1 + \mathbf{k}_2)$, shows how the time derivative, $\frac{d\mathbf{q}}{dt}$, of Hamilton's equations is approximated by a finite difference and the partial derivative of the Hamiltonian, $\frac{\partial H}{\partial \mathbf{p}}$, is approximated by the average of its value at two special points.

the discrete solution $\mathbf{q}_n = (\alpha_n(t_n), \beta_n(t_n))$ into an approximate continuous solution $\mathbf{q}(t) = (\alpha(t), \beta(t))$.

Since we are now considering mappings between interpreted languages, we need a notation to indicate this. Since we may regard such a mapping as an external effective interpretation of the synto-semantics of interpreted language, we can regard the mapping operation as being somewhat analogous to an (external) interpretation of an uninterpreted language in a model, which uses the double horizontal "models" notation $\models$. Accordingly, we introduce the notation

$$\alpha_0 = \pi/2, \beta_0 = -\pi/2 \,\Vert\!\overset{\sim}{\underset{\mathcal{H}\to\mathcal{N}}{\models}}\, \alpha(t_n) = ?, \beta(t_n) = ?,$$

where $\mathcal{N} = \left\{\mathbf{q}_{n+1} = \mathbf{q}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2), \mathbf{p}_{n+1} = \mathbf{p}_n - \frac{h}{2}(\mathbf{m}_1 + \mathbf{m}_2), H(\mathbf{q}, \mathbf{p}) = (4.10)\right\}$ indicates the constraints now imposed, to denote the mapping of the problem to the framework of the numerical method. The notation $\mathcal{H} \to \mathcal{N}$ indicate that the *source framework*, where the problem was originally posed, is $\mathcal{H}$ and the *target framework*, where the problem we mapped to is posed, is $\mathcal{N}$. Since we are working with interpreted languages it is important to keep track of which languages are being mapped to. Notice that in the mapping we have had to substitute a continuous solution $(\alpha(t), \beta(t))$ with a discrete one $(\alpha(t_n), \beta(t_n))$, since this is what the numerical method can provide.

Now, in our search for an effectively valid solution all would be well here provided we could solve the difference equations of the Störmer-Verlet method, but we do not know functions that solve these equations either. Given that the system exhibits chaotic behaviour this is not surprising. Adding to the difficulty is the fact that two of the equations (those for $\mathbf{m}_1$ and $\mathbf{k}_2$) are implicit, meaning that the variable we are solving for appears on both sides of the equation. For this reason, the typical strategy is to transform the problem again so that approximate solutions to the Störmer-Verlet equations can be found, making the solution of the problem fully algorithmic in the process. This means writing computer code to implement the Störmer-Verlet method, solving the implicit equations approximately, including code for the Hamiltonian (4.10). This is generally done in some high-level programming language, such as C, C++, Fortran or Python, or in a numerical mathematics system such as Matlab or Octave.

To map the problem from the mathematical framework of the numerical method into the framework of a programming language, we must interpret the constraints $\mathcal{N}$ of the numerical method in the synto-semnatics of the programming language. This means that the real-valued quantities of the numerical method are interpreted as floating point quantities, meaning finite precision rational numbers with a well-defined error model. This also means that approximate satisfaction of the constraints is judged in terms of floating point arithmetic. For concreteness, let us suppose that we choose C as our programming language, and we have interpreted the equations in $\mathcal{N}$ in C, giving us a programming framework $\mathcal{P}$. Then we need to write C code to solve our problem algorithmically, which amounts to the construction of code for an inference rule in $\mathcal{P}$, which we suppose we store in a file `stover.c`. For simplicity, we will assume that this program takes initial conditions (`a0` and `b0`) and a time interval (`[0,t]`) as input and outputs vectors (`a` and `b`) of solution values (`a[i]` and `b[i]`) over the given time interval. If we suppose that the software when run returns vectors `v` and `w` for the angles of the two weights, then internally to $\mathcal{P}$, we could write

$$\texttt{a0 = pi/2,b0 = -pi/2} \,|\!|\!|\!\overset{\sim}{\underset{\mathcal{P}}{\vdash}}\, \texttt{a = v,b = w},$$

where `pi` is a machine approximation of π, to express that the software computes what it is supposed to compute, namely that `v` and `w` contain effectively valid values of the state of the double pendulum according to the software version of the dynamics specified in $\mathcal{P}$.

An alert reader will recognize that there is something missing in the story as presented, since the code `stover.c` does not provide us with solutions either. It provides *code* for an inference rule, but not an inference rule itself. Thus, to obtain an inference rule we need to compile it into machine code so that a processor can compute solutions in binary, which are then converted back into floating point numbers to fill the vectors `v` and `w`. Thus, there is actually another synto-semantic transformation required to solve the problem fully. Since it does not serve us to consider this in detail here, we will just treat the compilation and running of the code as happening within the software framework.

Considered as an implementation of our original problem in a software environment, we then we can express the same content as the above displayed expression in terms of the language of our original problem with[20]

$$\alpha_0 = \pi/2, \beta_0 = -\pi/2 \,|\!|\!|\!\overset{\sim}{\underset{\mathcal{H}\to\mathcal{N}\to\mathcal{P}}{\vdash}}\, \alpha(t_i) = v_i, \beta(t_i) = w_i,$$

[20]Note that an alternative notation would be to simply write $\mathcal{H} \to \mathcal{P}$ to indicate the original source framework and the framework of synto-semantic interpretation. We are being fully explicit here for reasons of clarity, but since in general we could end up with graphs of synto-semantic mappings, some simplified notation will eventually be required, and the notation $\mathcal{H} \to \mathcal{P}$ need not lead to confusion when the sequence of mappings is clear.

where v_i and w_i are respectively the i-th components of the vectors $\mathbf{v}$ and $\mathbf{w}$, and $\mathbf{v}$ is mapped to $\mathtt{v}$ and $\mathbf{w}$ is mapped to $\mathtt{w}$ in the external synto-semantics. The truth of this statement expresses that we can obtain an effectively valid solution to our problem *in terms of the synto-semantics of the software system*, or in other words an effectively valid solution to the constraint system $\mathcal{P}$. But this is not what we really care about, since we are interested in a solution to our original problem $\mathcal{H}$. Thus, what we really want to know is whether when we *back-interpret* this solution into the original framework we obtain an effectively valid solution to the original problem.

Since the software implementation is designed to solve the Störmer-Verlet equations accurately, if we wrote the code properly, then the statement

$$\alpha_0 = \pi/2, \beta_0 = -\pi/2 \,|||\!\!\overset{\sim}{\underset{\mathcal{H}\to\mathcal{N}\leftrightarrows\mathcal{P}}{=\!=\!=}}\, \alpha(t_i) = v_i, \beta(t_i) = w_i,$$

which is interpreted in $\mathcal{N}$ will be true, where the $\mathbf{v}$ and $\mathbf{w}$ generated by our software are now interpreted as vectors of real numbers (each floating point value is mapped to its corresponding rational number). The standard of effective validity here is that of approximate solution to the Störmer-Verlet equations for the Hamiltonian H, *i.e.*, $\mathcal{N}$. But what we really want is to be able to back-interpret this solution into the framework of Hamilton's equations and have that statement

$$\alpha_0 = \pi/2, \beta_0 = -\pi/2 \,|||\!\!\overset{\sim}{\underset{\mathcal{H}\leftrightarrows\mathcal{N}\leftrightarrows\mathcal{P}}{\text{———}}}\, \alpha(t_i) = v_i, \beta(t_i) = w_i,$$

now interpreted in $\mathcal{H}$, come out as true, in which case the approximate solution of the Störmer-Verlet equations also furnishes us with approximate values along a solution curve to Hamilton's equations. This is determined by the properties of the numerical method, which if we have chosen our method well should be the case. Indeed, to have this come out true for as wide a range of initial conditions as possible was why we chose the symplectic Störmer-Verlet method in the first place. Notice the single horizontal stroke indicating we are no longer using an external synto-semantics and external standard of effective equivalence, instead we are back in the original framework. If this statement holds, then we have essentially solved our problem, since we can recover an effectively valid solution to Hamilton's equations, over some time interval, by interpolating the discrete set of values v_i, w_i returned by the software. Suppose we wrote code for an appropriate interpolant, yielding cuvres $v(t)$ and $w(t)$, then our ultimate solution would be expressed as

$$\alpha_0 = \pi/2, \beta_0 = -\pi/2 \,|\!|\!\!\approx_{\mathcal{H}}\, \alpha(t) = v(t), \beta(t) = w(t),$$

where we have now dropped the notation indicating the sequence of mappings that led to the effective solution. Recall that this statement will only ever be locally true, locally to some time interval, since eventually the overall error will accumulate and the error will exceed whatever tolerance we choose.

In this example of scientific problem-solving we have seen how the search for approximate solutions leads to mappings between synto-semantic frameworks of scientific problems *in a way that nearly preserves the structure of the problem*, in the sense of nearly-identical assumptions should yield nearly-identical solutions. Although much of the mathematical (geometric) structure of the problem is preserved in the mapping from the Hamiltonian framework $\mathcal{H}$ to the numerical framework $\mathcal{N}$, this structure is only preserved in a coded manner in the translation to the programming framework $\mathcal{P}$, even more so when the code is compiled to machine language. This reveals that what is really essential in the strategy of transforming the problem to find solutions is the preservation of the *inferential structure*, in the sense the the transformation provides an image of the graph of effectively valid inferences in the source framework as a nearly-identical graph of effectively valid inferences in the target, at least locally to some portion of the graph of inferences in the source. The transformation must make generating solutions easier to be useful, but near-preservation of inferential structure is nevertheless essential for the transformation process to produce approximate solutions to the original problem.

We can make this notion of inferential structure-preservation precise in terms of stability properties of the transformation between interpreted languages. As was just pointed out, part of what we require in approximate problem-solving is a transformation of the problem that preserves, at least locally, the inferential structure of the source framework, which ensures that solutions in the target solve a problem with effectively the same structure. Thus, assuming we have an effectively valid inference in the source framework $\mathcal{S}$, $\Gamma \,|\!|\!\!\approx_{\mathcal{S}}\, p$, then it must be the case that when Γ and p are mapped into the synto-semantics of a target framework $\mathcal{T}$ that p is still an effectively valid consequence of Γ, *i.e.*, it must be the case that

$$\Gamma \,|\!|\!\!\approx_{\mathcal{S}}\, p \Rightarrow \Gamma \,|\!|\!\!\approx_{\mathcal{S}\to\mathcal{T}}\, p.$$

Notice that this condition looks formally much like an effective version of the soundness condition in traditional logic. It is not a soundness condition, effective or otherwise, however, since the source language is already interpreted. Thus, the condition on a mapping really has to with the preservation of effective consequence relations. Since this is another kind of stability condition and one that deals with preservation of inferential structure in a mapping to an external synto-semantics, we will call this condition *forward inferential stability.* The term "forward" here indicates the forward direction of the mapping to the target language. Thus, forward inferential stability of a mapping assures us that we land in a target language on a problem having an effectively identical structure to the problem we had in the source.

Since the reason we have been considering mapping into a target language is to facilitate making effectively valid inferences in the source, a successful mapping between languages for this purpose requires that the mapping be invertible, so that we can import solutions from the target back to the source. To be able to do this, it must be the case that an effectively valid inference made in the target maps back to an effectively valid inference in the source, at least locally to those problems in the source framework of interest to us. In our logical notation this is expressed by

$$\Gamma \mathrel{\|\!\overset{\sim}{\models}}_{\mathcal{S}\to\mathcal{T}} p \Rightarrow \Gamma \mathrel{\|\!\overset{\sim}{\models}}_{\mathcal{S}} p,$$

which, in logical terms, expresses that an inference that is externally effectively valid is also internally effectively valid. Though this is akin to the condition of completeness in standard logic, it is not a completeness condition for interpreted languages. Since it does imply the ability to map effectively valid inferences *back* along the original mapping to the target language, we will call this condition *backward inferential stability.* A mapping between interpreted languages that is both forward and backward inferentially stable will be called *inferentially stable*, in which case the relation

$$\Gamma \mathrel{\|\!\overset{\sim}{\models}}_{\mathcal{S}} p \Leftrightarrow \Gamma \mathrel{\|\!\overset{\sim}{\models}}_{\mathcal{S}\to\mathcal{T}} p$$

holds, at least locally.

The condition of inferential stability implies that a target framework presents not only a problem with effectively the same inferential structure, but essentially the *same* problem, so that solving the target problem is essentially the same thing as solving the source problem. Stated another way, the two conditions imply that making inferences in the target framework is essentially the same thing as making inferences in the source, and *vice versa.* Thought of in another way, a forward inferentially stable map is like having an inferential continomorphism (generalized homomorphism) from the source to target,[21] and a backward inferentially stable map is like having an inferential continomorphism from the target to source. Having both together is like having an inferential contisomorphism (generalized isomorphism) from source to target,[22] telling us that inferences in the two languages are effectively equivalent where this condition holds. Thus, when this condition holds, effectively valid reasoning can be done in either language, so that inferences that are easier to make in one language can be mapped over to the other language. Thus, the condition of inferential stability is what allows mapping between languages to be used as a strategy to solve problems approximately. It is important to recognize that, just as for the condition of effective validity, it will only hold locally to some inferential scope, outside of which the mapping will become unstable and the inferences in the two languages will no longer correspond.

What is particularly interesting about the conditions of (forward and backward) inferential stability is that they correspond to conditions on the reliability of approximate reasoning. These conditions must hold in order for approximation methods to yield scientifically meaningful results. Moreover, in many contexts in applied mathematics, particularly in the context of numerical methods, applied mathematicians prove theorems to articulate the conditions under which an approximation method will generate solutions that are nearly-identical to the solutions of the original problem. In numerical methods these are numerical stability theorems, which essentially give conditions under which solutions of the numerical method provide approximate solutions to the original problem.[23] The proof of such a theorem is then actually an backward inferential stability proof. Forward inferential stability for numerical methods is ensured by generating them in terms of some method of approximation of the original problem. In certain cases there is a technical condition corresponding to forward inferential stability that must be met for any potential method, such as the consistency condition for numerical methods for ordinary differential equations.

[21]In categorical terms, this would correspond to some kind of generalization of a functor between categories.

[22]In categorical terms, this would correspond to some kind of generalization of a pair of adjoint functors between categories or a categorical equivalence.

[23]This kind of numerical stability is called *forward stability*, which assures that a method provides an approximate solution. Typically, however, theorems establish that the numerical method provides an exact solution to a slightly modified problem, rather than an approximate solution to the original problem. This alternative stability concept is called *backward stability*. Backward stability results can easily be accommodated by effective logic by adding an equivalence condition to the imposed constraints of a framework, so that the framework is expanded into a family of fixed frameworks, or by leaving any constraints that can be modified as assumptions rather than imposing them globally.

We remark before moving on that effective logic is a local logic without locality being built in to the system, in contrast to other alternative logical systems that explicitly introduce contextuality or local truth (*e.g.*, contextual languages, modal logic, topos theory). The basic notion in effective logic is near-structure-preserving variation, which has the effect of converting properties that are traditionally exact and making them sensitive to error and reliable only under certain conditions, validity being a paradigm example. Rather than being something we build into the structure of the system, then, local properties become a direct consequence of approximation. This matches the approach used in science, since it is the need to solve problems efficiently that often leads to seeking out approximations and it is the nature of approximation to make methods only locally valid, applicable or stable.

4.5 Modeling, Complexity Reduction and Inferential Structure-Preservation

We have now seen how we can understand problem solving methods in science in terms of moving effectively valid inferences between frameworks via near-structure-preserving maps. Moreover, we have seen how this process can be understood in terms of forward inferentially stable maps allowing the movement of inferences from a source framework to a target, and backward inferentially stable maps allowing movement of inferences from the target back to the source. The strict form of validity in traditional logic cannot account for these processes in a direct way because effectively valid inferences, being approximate by nature, are generally not strictly valid and the maps between frameworks are also only structure-preserving in an approximate sense. Effective logic accounts for problem solving strategies in science by showing how a precise (generalized) logical structure can nevertheless obtain in scientific methods that involve approximations.

As we have noted, the approach of seeking inferentially stable mappings between problem-solving frameworks is very common in applied mathematics, where solving a problem in the framework in which it is originally posed proves often to be very difficult, prompting search for nearly-equivalent problems that are easier to solve. Examples of these methods include asymptotic analysis, perturbation theory and numerical methods, which covers quite a large range of mathematical methods. This kind of method also underlies the ubiquitous approach in pure mathematics, *e.g.*, algebraic geometry and algebraic topology, of solving problems in one framework or category by transforming to equivalent problems in another framework or category, though in this case near-identity is based upon exact structure-preservation. It was also mentioned above that this kind of method underlies the modular methods used to accelerate symbolic computations. Thus, effective logic stands to be able to capture the natural reasoning processes of a large portion of science, by capturing the *structure* of the reasoning that scientists use in their own languages. This contrasts distinctly with the traditional approach of reconstructing theories by casting them in a uniform formal language or system.

Moreover, as is shown in [7], the methods of computational science, including both numerical methods and symbolic computation, rely on transformations between problems to reduce the computational complexity of mathematical problems sufficiently so as to make them rapidly computable in practice. The epistemological drive underlying these methods is the need to overcome obstacles to making efficient, reliable inferences given the contextual constraints of scientific practice. We can call such efficient, reliable inference *feasible inference*. These feasible inference methods must preserve the inferential structure of the problem to be able to generate solutions that can potentially correspond to solutions of the original problem, and they must be invertible so that the computed solutions can actually produce a solution to the original problem. Thus, it is seen that strategies of complexity reduction in computational science rely on inferentially stable transformations of mathematical problems in a way that makes their solution computable rapidly.

A consequence of this observation is that if all that is required for an accelerated algorithm is that it reduce computational complexity and preserve inferential structure, then little or none of the mathematical content of the problem *needs* to be preserved in the transformation, provided that solutions to corresponding problems correspond. Thus, despite the fact that it is natural and standard to look for solutions to problems by transformations that nearly-preserve their mathematical structure in some way, there may nevertheless exist transformations with even lower complexity that preserve little or no mathematical structure at all, yet nevertheless deliver efficient, reliable solutions. It was argued above that the successful reduction of problems to machine language is an illustration of this kind of idea.

The notion of inferential stability also provides a new way of thinking about the abstraction processes involved in mathematical modeling. From the perspective of effective logic, a mathematical modeling problem begins with the desire to make effectively valid inferences about the behaviour, or the reasons underlying the behaviour, of some phenomenon. Such inferences are usually formulated in some scientifically-augmented form of natural language.[24] We wish to be able to make inferences about properties or states of a system or phenomenon, sometimes with very tight tolerances on error. Since we typically cannot make these inferences in our scientifically-augmented natural language, we resort to mapping to some other, usually mathematical, language that we expect will assist us in making the desired inferences. We do this so that by making corresponding inferences in the scientific language, we can map the conclusions back to our (scientifically-augmented) natural language to yield descriptions, predictions and explanations of behaviour of the phenomena. This is to say that we require the mapping from the natural language to the scientific language to be inferentially stable. When the mapping has this property, we can rely on inferences made in the target language to be informative about the world.

We can therefore understand the mathematical modeling process as overcoming an inferential obstacle to drawing conclusions about the structure or behaviour of some natural phenomenon, conclusions that are not accessible without the use of scientific theories or mathematics. We use models, then, to facilitate reasoning processes that are not feasible directly. Thus, mathematical modeling can also be seen as a strategy of problem transformation that makes inference feasible. Moreover, we have seen that for this process to be successful and reliable, the mapping from the description of the phenomenon using natural or operational/experimental language to the language of the mathematical model must be inferentially stable, so that the conclusions drawn in the model give reliable conclusions about the phenomenon. Consequently, we may see mathematical modeling procedures as tools for reducing the *inferential complexity*, *i.e.*, the cost of drawing inferences, for description, prediction and control of natural phenomena by transforming between languages. Furthermore, just as for computational complexity reduction strategies in computational science, a key requirement is inferential stability.[25]

With effective logic, therefore, we obtain a picture of the mathematical modeling process that accounts for the kinds of methods used throughout the entire process, including computation, and a picture that is, or can be, fully compatible with the actual methods

[24]Specifying the semantics for such expressions is a notoriously difficult problem, but one faced by any attempt to account for our descriptions of the world. Accordingly, I will not consider this matter here except to point out that a semantics often relies on experiential states, states of experimental apparatus or some canonical, and maximally scientifically neutral, physical model of phenomena.

[25]We note here that the observation above concerning the fact that only the inferential structure, not the content, needs to be preserved in transformations, can be applied the the modeling case. This has interesting philosophical consequences for how we might understand scientific representation, since the effective logic model is consistent with a plurality of aims among scientists, some of whom will be interested in direct descriptions of the structure of some part of the world and others content with empirical adequacy. Any further consideration of these issues is beyond the current scope.

that practicing scientists use. This is the advantage of having a form of description that can map on to scientific language, capturing its basic structure, rather than requiring a mapping of scientific language into a logical language in order to reconstruct it. The task with using effective logic for philosophical purposes, then, is to ensure that it does indeed capture the structure of inference in scientific languages. I have only presented a limited amount of evidence for the representational capacity of effective logic in this chapter. Though a more fully developed argument is reserved for future work, further evidence that effective logic captures the structure of scientific inference in practice is provided in [6, 7]. I should point out, however, that though effective validity can only capture nearly deductive inference within a language, the ability to use inferentially stable mappings between languages allows one to make inferences that do not remotely resemble deductive inference by appealing to radically different languages that nevertheless allow one to complete effectively valid inferences by mapping back to the original, source framework.

4.6 Conclusion

In summary, I have presented a generalized logic based on the concept of effective validity that stands to account for the basic structure of inference in scientific practice, to clarify the structure of reliable methods of computational complexity reduction in computational science, and to provide an account of the mathematical modeling process that views modeling methods as tools of reliable inferential complexity reduction. Such an account emerges naturally from regarding scientific reasoning procedures in terms of inferentially stable mappings between languages.

As it has been presented, this generalized logic functions to capture the basic form of scientific inference as it occurs in real scientific languages. It is opposite in approach to the traditional strategy of representing scientific inference through reconstructions of scientific languages in some formal language. Rather, effective logic employs an epistemological modeling approach in the sense that it captures the structure of inference in particular interpreted languages rather than requiring treatment in some specialized uninterpreted formal language or large class of models. At the same time, it is complementary to traditional rational reconstruction because it is well-suited to a very different problem, *viz.*, mapping the inferential structure of scientific practice. With a more flexible, error sensitive notion of validity, it becomes possible to capture a wider range of scientific inference and has the potential to produce insights into the reliability of scientific languages and to cope with the potential for instability in reasoning involving error.

By extending the traditional notion of valid inference into a context of variation, we open up logic to a treatment of the forms of inference typical in the approximation methods used in mathematical analysis, in contrast to traditional logic which more closely suited to the forms of inference typical in the exact methods of abstract algebra. At the same time, it opens up logic to an accurate treatment of the forms of inference in the mathematical modeling process and potentially to scientific inference more broadly. We have seen how the introduction of a context of variation can lead to different kinds of mathematical questions, such as the stability of consequence relations, or even mathematical proofs, under certain (near-identity) transformations of the syntax. Nothing here is strictly new, since there already exist forms of each of these things within traditional logic, and traditional logic can surely illuminate all of these things in its own terms. The difference with effective logic is that we move toward a natural language for approximate inference, which stands to introduce fresh and illuminating perspectives on old problems while also suggesting new kinds of questions and directions of inquiry.

4.7 Bibliography

[1] R.W. Batterman. *The devil in the details: Asymptotic reasoning in explanation, reduction, and emergence.* Oxford University Press, USA, 2002.

[2] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[3] W.L. Harper. *Isaac Newton's Scientific Method.* Oxford University Press, 2011.

[4] Carl Hempel. Aspects of scientific explanation. 1965.

[5] Derek F Lawden. *Elliptic functions and applications.* Springer New York, 1989.

[6] Robert HC Moir. *Structures in real theory application: A study in feasible epistemology.* PhD thesis, The University of Western Ontario, 2013.

[7] Robert HC Moir. Feasible computation: Methodological contributions of computational science. In *Physical Perspectives on Computation, Computational Perspectives on Physics.* Cambridge University Press, 2018.

[8] Paul Oppenheim and Hilary Putnam. Unity of science as a working hypothesis. 1958.

[9] Frederick Suppe. *The structure of scientific theories.* University of Illinois Press, 1974.

[10] Patrick Suppes. *Models of data*. Springer, 1969.

[11] Patrick Suppes. *Representation and invariance of scientific structures*. CSLI Publications Stanford, 2002.

[12] Bas C van Fraassen. On the extension of beth's semantics of physical theories. *Philosophy of science*, 37(3):325–339, 1970.

[13] Bas C van Fraassen. Representation: The problem for structuralism. *Philosophy of Science*, 73(5):536–547, 2006.

[14] B.C. van Fraassen. *The scientific image*. Oxford University Press, USA, 1980.

[15] Vladimir Voevodsky et al. Homotopy type theory: Univalent foundations of mathematics. *Institute for Advanced Study (Princeton), The Univalent Foundations Program*, 2013.

[16] M. Wilson. *Wandering significance: An essay on conceptual behavior*. Oxford University Press, USA, 2006.

[17] W.C. Wimsatt. *Re-engineering philosophy for limited beings: piecewise approximations to reality*. Harvard Univ Press, 2007.

CHAPTER 5

Feasible Computation: Methodological Contributions from Computational Science

Feasible Computation: Methodological Contributions from Computational Science[1]

5.1 Introduction

The focus of the standard model of computability is effective calculability, *viz.*, a mechanical method for computing values of a function on the natural numbers or solving a symbolic problem. The Church-Turing thesis is then that this informal concept is identical to that of computability by recursive functions. It is this notion of computability that underlies the standard definition of a formal system as a recursive axiomatic system, *viz.*, a system of words whose formulas (*e.g.*, theorems) and proofs are effectively decidable. Indeed, this proof-theoretic conception of an axiomatic system was viewed by Gödel as the only permissable interpretation of "formal system" [17]. Thus, this formal understanding of mathematical theories represents mathematical proof as a form of computation. Viewed in relation to the common view in philosophy that scientific inference can be faithfully represented as logical inference, this common view regards scientific inference as computational.

Whether computability is modeled in terms of recursive functions on the natural numbers or in terms of manipulations of symbols or words, we can regard computation as involving a map between an input and an output, so that computation in general is essentially equivalent to function evaluation. Since mathematical problems can be regarded in the same way, as maps between certain input data and output solutions, problem solving can also be regarded in terms of function evaluation. Given, futher, the connection between inference and formal systems, scientific inference can also be regarded as function evaluation. As a result, underlying computation, problem solving and inference is always a form of function evaluation. Consequently, we will move between these perspectives throughout the paper where it is useful to do so, keeping in mind that computing values of a function underlies each of the notions.

It is traditional to think of computation (and the associated concepts of problem solving and inference) in exact terms, so that a computational problem is concerned with computing the exact value of a function. In contexts where it is known that exact solutions to a problem exist, the question becomes one of whether there are effective (in

[1]A version of this paper will be published in the volume *Physical Perspectives on Computation, Computational Perspectives on Physics* by Cambridge University Press, available in 2018.

the sense of finitely computable by an algorithm) methods to solve them, if so whether efficient algorithms exist (in the sense of computable in polynomial time/space), the complexity of available algorithms, the minimum complexity among these, *etc.*

A consideration of computation in scientific practice raises somewhat different complexity concerns, and what counts as an acceptable complexity can vary depending on the context, including mathematical, software, hardware, energy and financial constraints. Thus, there is a subtler condition on computational complexity that has to do with the ability to compute solutions to a problem in a manner that is fitting to the constraints imposed by the context in which those solutions are to be used in practice. We will call an algorithm that has this property *feasible* and problems for which such algorithms exist *feasibly computable*.[2] A problem is *feasibly uncomputable* when no feasible algorithm exists. Feasibility does not just have to do with finding adequately low complexity algorithms, since it is also an epistemological concept, having to do with methods we have epistemic access to. A feasibly uncomputable problem might become feasibly computable in the future if new analytic methods, better algorithms or technology are developed. For example, the Berlekamp-Zassenhaus algorithm for factoring polynomials over the integers was not feasible for computing factorizations when it was first developed in the sixties, but became feasible with subsequent advances in computing technology.

The field of computational science is concerned with generating feasible algorithms to solve mathematical problems, usually those that are important in scientific applications. An important difference between such feasible algorithms from traditional algorithms considered in computability theory, is that in general they involve various forms of approximation. We will see that there is a common strategy in computational science that can take a problem that is not feasibly computable, or not sufficiently feasible in the sense that the computational complexity is too high, and then generate a (more) feasible algorithm to a slightly modified problem. This strategy to produce feasible algorithms itself has an algorithmic structure, involving a recursive process of complexity reducing transformations of the problem into a feasibly computable problem, followed by a recursive back-interpretation of the solution, a process that can itself require problem solving. We will see how one, more general, version of this strategy underlies numerical comput-

[2]Note that computing solutions in a manner fitting to the constraints imposed by practice often involves introducing forms of error into a problem. Accordingly, feasible computation requires that algorithms be reliable, in the sense of providing robustly accurate solutions or information about a phenomenon, given the kinds of variation of the problem that are contextually relevant. The focus of this paper is primarily the complexity advantages from transforming problems, but it is important to appreciate that to provide feasible solutions to problems, these transformations must also be reliable.

ing, which uses approximations, and how a more restricted version underlies symbolic computing, which is exact. The nature of this feasible computation strategy in these two branches of computational science has some consequences for computability theory, which we consider in the final section.

Before we introduce this feasible computing strategy, we will consider its roots in the history of science. Though the exact form of the method arguably has its roots in pure mathematics, we trace the more general form of the method, which allows approximate evaluation, to the techniques developed by physicists to overcome the computational limitations of the mathematics used to formulate theories and models of natural phenomena. Approximation methods were developed by physicists to get information out of theories and models even when they could not be solved exactly under realistic assumptions. The need to obtain quantitative results therefore drove the development of methods that ultimately yielded precise and reliable numerical approximations. We find in this process reasons why the feasible computing strategy is likely to be replicated widely in scientific practice, even outside of the mathematical sciences, specifically because the motivation for the method is epistemological: since scientists are constantly confronted with the computational limitations of their own conceptual tools, they have to find reformulations of problems that make inference feasible.

It emerges, therefore, that feasible computing is a fundamental part of much of scientific inference, as well as at the core of advanced algorithms for solving problems in computational science. Moreover, by revealing an algorithmic strategy of converting feasibly uncomputable problems into feasibly computable ones, we show how computational science extends beyond the traditional theory of computation and is revealing new aspects of computability, aspects of fundamental importance to scientific inference and practical scientific computation.

5.2 Approximate Problem Solving in Physics

The ability to calculate is a fundamental part of the scientific process as a result of the need to work out the consequences of our theories for real world phenomena. Much of the time there is a tension between minimizing the complexity of theoretical inference and minimizing the complexity of application of a theory. A simple example of this contrast is the situation in logic, where for (meta)theoretical purposes, we work with the most limited set of axioms and inference rules (often just *modus ponens*) as possible, but for the purpose of doing proofs within a system we want as rich system of inference rules and theorems as possible to simplify the proofs.

In the context of scientific practice, this need to minimize the complexity of theories can lead to a gulf between theory and the ability to describe the phenomena. An extreme example of this occurred in the history of fluid mechanics. Although the standard equations of fluid motion were developed in the eighteenth and early nineteenth centuries, by Euler and Navier, based on work by many others, the ability to draw consequences from these theories for realistic fluids was extraordinarily limited. This led to statements like d'Alembert's paradox, which stated that perfect fluids in steady motion exerted no force on fully-immersed bodies. It took an extensive development over almost two centuries, involving both theoretical innovations and deep investigations into physical fluid behaviour, to resolve this gap. The theoretical development was driven by attempts to apply the theory (see [7], Ch. 8), and this required novel use of approximations in the construction of the theory itself. Examples of this are modifications of the equations of motion (*e.g.*, adding higher order terms), allowing discontinuities (*e.g.*, shock waves), and asymptotic behaviour (*e.g.*, Prandtl's boundary layer theory). Thus, approximation is not only important in terms of getting numbers out of known theories or models, but is part of theoretical inference and the process of theory development itself.

Determining the consequences of a theory for the behaviour of the phenomena it describes requires generating solutions of the equations in realistic situations. Though exact solutions are extremely valuable, they are only available in very rare cases, and typically only for highly idealized or controlled situations. Since it is a matter of great priority to be able to know what a theory says about the phenomena, it becomes essential to use approximation methods to be able to gain information about the consequences of theoretical models. Given that applying a theory generally requires the construction of models, which introduces forms of systematic error, it is usually justifiable to solve modeling problems using approximation methods provided they introduce less severe forms of error than those introduced in model construction. For these reasons, the history of physics is full of strategies for effective approximation. Three important general classes of approximation methods, namely numerical methods, asymptotic methods and perturbation expansions, all have origins in the need to complete inferences that cannot be completed exactly. Moreover, they involve a strategy of *modifying the problem* into one where inference becomes feasible.

To illustrate the nature of these methods as simply as possible, we will consider how they emerge out of series expansions of functions. This actually captures the nature of these methods very well and is indeed how they emerged, since the method to develop an expression into an infinite series occurred very early. A convenient starting point is Taylor's theorem, announced by Taylor in 1712, though it appeared in a manuscript of Gregory in 1671, which was developed out of Newton's forward difference interpolation formula [18, p. 332]. Taylor's theorem allows one to expand a (suitably differentiable) function $f(x)$ about a point a into a power series

$$f(a+h) = f(a) + f'(a)h + \frac{f''(a)}{2!}h^2 + \frac{f'''(a)}{3!}h^3 + \cdots,$$

where $h = x - a$. This is useful theoretically, since it can allow one to compute integrals of functions that are not known by expanding the integrand into a series and integrating term by term and summing the result. But because h^n goes rapidly to zero as $x \to a$, it also provides a very useful algorithm for approximating the value of $f(x)$ near a, *i.e.*, when $|x - a| \ll 1$. Thus, by truncating the series up to a certain order (power of h) we can obtain local approximations of $f(x)$. In so doing, we substitute the function $f(x)$ with a nearby function equal to the truncated series.

It is precisely this sort of strategy that underlies the use of numerical methods to solve differential equations. The simplest numerical method, attributed to work of Euler in 1768-9 [10, p. 141], is motivated simply on the basis of a Taylor expansion of the solution truncated after first order. Consider a differential equation with an initial condition (called an initial value problem) of the form

$$y'(t) = f(y,t), \quad y(0) = y_0, \tag{5.1}$$

where we interpret the independent variable t as the time. We can "take a time step" by approximating the value of the solution at $t = h$, for $0 < h \ll 1$, in terms of known quantities by truncating the Taylor expansion of $y(t)$ about the point $t = 0$

$$y(h) = y(0) + y'(0)h = y_0 + f(y_0, 0)h,$$

where we have substituted equation (5.1) for $y(0)$ and $y'(0)$. Since the initial time is $t_0 = 0$, the values of t and y after the first step are $t_1 = h$, $y_1 = y(h)$, and we can iterate this procedure, computing the value of $y_{n+1} = y(t_{n+1}) = y((n+1)h)$ in terms of the recurrence

$$y_{n+1} = y_n + f(y_n, t_n)h, \tag{5.2}$$

which is Euler's method for a constant step size h. If we imagine connecting the points (t_k, y_k) by straight lines, then we generate an approximate solution to (5.1) as a polygonal arc.

There are two important features to note about this procedure. One is that, under mild conditions on the function f in a region around $(0, y_0)$, as $h \to 0$ the polygonal arc converges to the solution of (5.1), as was shown by Cauchy prior to 1840. Thus, numerical methods provided an early tool for demonstrating the local existence of solutions to differential equations. The second is that it provides a very simple way of estimating the behaviour described by a differential equation, even if the nonlinearity of the function f makes analytic solution impossible. To effectively control the error, however, the size of h needs to be very small, making calculation by hand tedious. Numerical methods were nevertheless sometimes used, particularly to avoid tedious perturbation methods [16, 405]. Although prior to the development of numerical methods for differential equations, Clairaut, aided by two associates, took months in 1757-58 to perform effectively the first large scale numerical integration to predict the return of what thereafter became known as Halley's comet [16, 305]. The subsequent nineteenth (and early twentieth) century development of numerical methods for differential equations by Adams, Heun, Runge and Kutta, were motivated by the desire to have more computationally efficient methods for applications in celestial mechanics, thermodynamics and ballistics calculations [10, p. 286].

The structure of this kind of method is important for us to note. Given the difficulty in solving differential equations analytically, the need to obtain information about the consequences of differential equations for the behaviour of real world phenomena motivated the development of accurate and efficient computational procedures (numerical methods) to compute approximate solutions. The computational procedure involves a modification of the problem, which here requires moving from a differential equation (5.1) to a difference equation (5.2), for which solutions could be computed (or approximated)[3] feasibly. The (approximate) solution of this modified problem then provides an approximation of the solution of the differential equation, giving insight into its consequences for behaviour. Thus, an inferential obstacle at the level of theory is overcome by clever use of approximation.

Even before Brook Taylor proved the famous theorem bearing his name, scientists developed functions into series expansions, many of which were worked out by Newton.

[3]As we will see in the next section, the difference equations of numerical methods often cannot be computed exactly but give way to an iterative strategy for approximate solution.

The methods for expanding into series emerged out of the theory of finite differences and interpolation, themselves developed for the purposes of computing numerical tables for logarithmic and trigonometric tables [10]. Since such methods provide means for generating *infinite* series expansions, which can convert mathematical problems involving transcendental functions to equivalent calculations on polynomials, infinite series were an important tool from the inception of the differential calculus. Indeed, Newton relied heavily on such methods.

A number of difficulties arise from this approach, however, principal among them is the question of when, and in what sense, can one identify a function with its power series expansion. For example, Newton (1665) and Mercator (1668) [12, p. 354] both obtained the series

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots,$$

which was observed to have an infinite value for $x = 2$, even though the function value is $\log(3)$ [13, p. 437]. It was therefore recognized early on that series expansions needed to be treated with care. Responses to the risks of divergent series were varied. Some, such as d'Alembert, singled out convergent series and were suspicious of the use of non-convergent series, and the first tests of convergence were introduced in the eighteenth century, by Maclaurin, d'Alembert and others. Others, however, and Euler in particular, recognized that nonconvergent series could be useful as representations of functions, so that expressions such as

$$\tfrac{1}{2} = 1 - 1 + 1 - 1 + \cdots$$

could be not only meaningful but useful in algebraic manipulations even though the sum of the series lacked an arithmetic meaning. Nonconvergent series were not necessarily devoid of arithmetic content, however, and specifically divergent series often proved useful for computing accurate approximate values of functions.

In a work 1754/55 work on divergent series (see [13], pp. 450-1) Euler considered the differential equation

$$t^2 y' + y = t,$$

which has the solution

$$y = e^{1/t} \int_0^t \frac{e^{-1/x}}{x} dx. \tag{5.3}$$

Euler observed that the divergent series

$$y = t - (1!)t^2 + (2!)t^3 - (3!)t^4 + \cdots$$

also satisfies the differential equation *and* that the sum of a small number of terms of this series gave a good approximation to the value of the integral, even though the series diverges. This peculiar property has to do with a particular relationship between partial sums of the series and the value of the integral. It turns out that this relationship can be understood in terms of approximation, which explains why divergent series can nevertheless be useful for approximation.

This behaviour can be understood in terms of the remainder between the function being expanded into a series and partial sums of the expansion. Taylor's theorem, as it is presented now, includes not only a prescription for computing a series expansion of a function from its derivatives, but also an expression for the remainder R_n when the series is truncated after n terms, *i.e.*,

$$f(a+h) = f(a) + f'(a)h + \frac{f''(a)}{2!}h^2 + \cdots + \frac{f^{(n)}(a)}{n!}h^n + R_n,$$

where the expression

$$R_n = f^{(n+1)}(a+\theta h)\frac{h^{n+1}}{(n+1)!}, \quad \theta \in (0,1),$$

due to Lagrange is sometimes given. What one may observe from this, is that when as h goes to zero, the remainder term R_n vanishes faster than the order n term of the truncated series. More formally, we may observe that if $T_n = \frac{f^{(n)}(a)}{n!}h^n$ is the order n term of the expansion, then $\lim_{h\to 0} R_n/T_n = 0$. Stated specifically in terms of h^n, the powers used to expand the function, $\lim_{h\to 0} R_n/h^n = 0$. This latter property is written as $R_n = o(h^n)$ as $h \to 0$, which can be read as saying R_n is *asymptotically dominated by*, or *infinitesimal compared to*, h^n as $h \to 0$.[4] The significance of this is that by knowing that the last term in an approximation dominates the remainder, we can be assured that sufficiently close to $h = 0$ the partial sum will provide a good approximation. The important point is that this is the case *whether or not the series converges.*

To illustrate this, consider the case of the integral of $\int_1^x e^t/t\,dt$. Through repeated use of integration by parts, this integral can be developed into a series as

$$\int_1^x \frac{e^t}{t}dt = e^x\left(\frac{1}{x} + \frac{1}{x^2} + \frac{2!}{x^3} + \frac{3!}{x^4} + \cdots\right).$$

[4]Strictly speaking the order notation should be written as $R_n \in o(h^n)$ since $o(h^n)$ is a set of functions, but I follow the standard abuse of notation by writing it as an identity.

There are two important features of this series for us to note. First, for no value of x does the series in the brackets converge. Thus, there is no hope to compute an arithmetic sum of the series to compute the value of the function this way. Nevertheless, the series has the property that the remainder R_n after n terms is $o(1/x^n)$ as $x \to \infty$; thus, by taking sufficiently large x, partial sums of the series can provide good approximations of the value of the integral. Series that have this property that the remainder is asymptotically infinitesimal compared to the last term of the truncated series are called *asymptotic series*. They generally have the property that for a given value of x there is an optimal number of terms to take to obtain the best approximation before the approximation begins to get worse. They also tend to be very advantageous for the purposes of computation because often only a few terms are needed to get very accurate approximations.

To illustrate, by keeping 4 terms of this series, the approximation is correct to 3 decimal places by $x = 12$, and to 4 decimal places by $x = 20$, getting better the larger x becomes. On the other hand if we fix x, say at $x = 12$, then though keeping more terms initially yields a better approximation, giving a result correct to 5 decimal places by keeping 9 terms, after that adding more terms makes the result worse. By 13 terms the approximation is only correct to 3 decimal places again, and by 30 terms the approximation is not correct to any decimal place.

A major advantage of asymptotic series, given their tendency to converge rapidly taking only a few terms, is that provided one is considering phenomena in an appropriate asymptotic regime so that truncated series are accurate, one can transform an unknown or difficult to manipulate function into functions that are known or more easily manipulated, an effective reduction in computational complexity. In the series considered above, we can usefully approximate a non-elementary function, essentially the exponential integral $\mathrm{Ei}(x)$, by elementary functions. Thus, by strategic use of truncated asymptotic series, we can complete inferences that are not otherwise possible if we were to restrict to exact values or solutions, or can complete inferences more easily. This is therefore another case of using approximation to render inferences (more) feasible, and it is once again the difficulty in obtaining exact solutions that motivates it.

The strategy of using asymptotic approximations can also be understood in terms of a modification of the problem. In the case of approximation of integrals, we can see that the result does not differentiate to the integrand we are trying to integrate (*e.g.*, $e^x/x \neq \frac{d}{dx}e^x(1/x + \cdots + (n-1)!/x^n)$ for any n).[5] Thus, we have actually solved a slightly

[5]This approach to analyzing approximation error in terms of modified problems underlies the method of backward error analysis in numerical computing (see [5]). Although backward error analysis is relevant

different problem, but it is close enough in the appropriate asymptotic regime to give us useful information. Although subtle in this case, the character of asymptotic methods as a modification of the problem becomes more explicit when they are used to solve mathematical problems, such as differential equations. At least as far back as Euler, but more fully developed in the work of nineteenth century mathematicians such as Jacobi, Liouville and Stokes, divergent series expansions were used to solve differential equations approximately, in some cases explicitly involving an asymptotically valid modification of the differential equation [14, pp. 1100*ff.*]. In this latter case, the equation itself is considered in an asymptotic regime where it is more easily solved, so that the problem is modified directly. The solution thus obtained then carries information about the solutions to the original problem, and the behaviour of phenomena accurately described by it, in the same asymptotic regime. Once again, an inferential obstacle at the level of theory is overcome by a clever use of approximation.

As was stated earlier, exact solutions are extremely valuable when they are available, but not only because they do not introduce error. They are also important for enabling the computation of approximate solutions to problems that differ by a small amount, or the approximate characterization of phenomena that differ only slightly from phenomena that can be described exactly. In this case we have a problem that can be regarded as a small perturbation of a problem that can be solved exactly. As such, the known exact solution can be used to compute approximate solutions to the perturbed problem. The mathematical techniques developed to make this approach work became perturbation theory, and have their origins in astronomy.

Although Newton introduced in the second edition of the *Principia* the idea of computing corrections to the two-body problem of the Moon's orbit around the Earth due to the effect of gravity of the Sun [16, p. 277], the analytic methods of perturbation theory find their roots in Euler. In his work on treating three-body problems as a small perturbation of Keplerian two-body motion, he introduced the technique of giving solutions in terms of trigonometric series, *i.e.*, series in powers of $g\cos\theta$ for small g, and even introduced methods for accelerating their convergence.[6] These methods were developed initially for specific three-body problems by Clairaut, d'Alembert and Euler himself, and later developed into a more general approach, beginning with Laplace and Lagrange. The perturbative approach became a standard method for constructing analytic theories

to our discussion here, consideration of it is beyond our scope.

[6]Such methods can sometimes convert divergent series into convergent ones, revealing more clearly how a divergent series can still represent a function in an algebraic sense.

of planetary and satellite motion, as well as for computing numerical tables for use in astronomical study and navigation.

Much later these methods were developed into a more general method for solving equations that could be regarded as a small perturbation of a problem with a known solution, and is widely used in contemporary physics. The basic idea of the method is the following. Suppose we have a differential equation of the form

$$y' = f(y, \varepsilon)$$

such that when $\varepsilon = 0$ the solution $Y_0(t)$ is known. The strategy is then to expand the solution when $|\varepsilon| > 0$ in powers of ε so that

$$y(t) = Y_0(t) + Y_1(t)\varepsilon + Y_2(t)\varepsilon^2 + \cdots, \tag{5.4}$$

which is usually truncated after a certain number of terms, thereby determining the order of the perturbations one is considering (second order if we truncate after the ε^2 term). By substituting this equation into both sides of the differential equation, possibly also developing $f(y, \varepsilon)$ into a power series in ε, and collecting terms with the same power of ε onto one side of the equation so that the other side is zero, one obtains a series of terms multiplied by successively higher powers of ε. Then the coefficient of each power of ε is a differential equation, each of which must be set to zero (since the other side of the equation is zero), yielding a sequence of (successively more tedious to solve) differential equations, beginning with the one we already have a solution for. Solving these equations for $Y_1(t)$, $Y_2(t)$, *etc.*, then provides first, second, and so on, corrections to the exact solution according to (5.4). Thus, provided ε is sufficiently smaller than 1, one can obtain a good approximate solution to the perturbed equation. This process as described rarely works in a straightforward way, and can itself involve laborious calculations, but it gives a sense of the method (see also [2]).

Just as for numerical methods and asymptotic expansions, perturbation theory involves developing functions into series, which are typically asymptotic, as was recognized early on but only properly emphasized and treated after Poincaré [16, p. 422], [14, p. 1103*ff.*]. And as in those cases, perturbation methods work by solving a modified problem. Rather than directly approximating or replacing the problem with a nearby one, the approach here is to attempt to stretch an exactly solvable problem into the problem one wishes to solve, or an approximation to it, since the problem one wishes to solve is too difficult to solve directly. The perturbation solutions, particularly as rapidly converging asymptotic series, then convey information about the solutions to the insoluble problem, possibly in some asymptotic regime. Once again, an inferential obstacle at the level of theory is overcome by a clever use of approximation to make inference feasible.

It is evident even from this limited consideration of approximation methods in physics, that approximation is about far more than simply calculating numbers or determining behaviour in particular situations. Approximation methods are part of the theoretical process and play an important role in the development of theory. This is very clear in the case of perturbation methods in the history of astronomy, which led to better developed theories of planetary and lunar motion, as well as the discovery of Neptune. Even numerical methods allowed early demonstrations of the existence of solutions to differential equations. This role for approximation methods has only grown, with asymptotic methods becoming essential tools of analysis throughout physics, applied mathematics and computer science. Perturbation methods also have wide application, underlying the phenomenally successful perturbative approach to quantum field theory and modern fluid mechanics, as well as being extended to many other classes of mathematical problem and finding applications throughout applied mathematics and computer science [1]. And numerical methods, far from simply being tools for calculating values, have developed into sophisticated tools of scientific inference, such as the tools of geometric numerical methods [11], and new branches of science, such as computational fluid mechanics (see [20]).

We have also seen in this section how approximation methods in physics enable one to extract information from theoretical models when exact solutions are not available. There is a common pattern of using approximation to modify the original problem one wished to solve into one from which information, or solutions, can be more feasibly attained, thereby giving approximate information about behaviour of solutions to the original problem. As will be made clear in the next section, this is the kernel of an algorithmic process of feasible problem solving that underlies the success of methods in scientific computing for rapidly solving difficult mathematical problems.

5.3 Feasible Computation: Algorithmic Problem Solving

The ability to solve mathematical problems is essential to scientific inference. In the previous section we saw how approximation methods have become an essential tool for making scientific inference feasible given the rarity of exact solutions, and how such methods involve a modification of the problem into one that is solvable. The strategy of modifying a problem to make solution feasible is not restricted to approximation methods, however, and underlies the methods of analytic geometry, introduced by Fermat and Descartes in the seventeenth century [12, p. 302*ff.*], Galois theory and algebraic geometry developed in the nineteenth century, as well as algebraic topology and the modern algebraic geometry developed in the twentieth century, which all solve problems in their original form by converting them into equivalent algebraic ones.[7] Though such methods are exact, they nonetheless involve transforming a problem into a more feasible one, so that solutions obtained not only carry information about the original problem but actually yield exact solutions. In such cases, the problems are equivalent, so that reasoning can be performed in whichever context is preferable.[8]

Whether or not approximation is used to solve an infeasible problem, it is generally necessary to make more than one transformation of the problem to reach a feasibly solvable problem. Such a strategy underlies the common method of reducing a mathematical problem to another class of problem that is simpler or regarded as already solved. A nice example of this approach occurs in the analytic solution of partial differential equations (PDE), which can be "solved" by reducing them to a system of ordinary differential equations (ODE), reducing an infinite dimensional problem to a finite dimensional one, a reduction in complexity. Van Dyke [19] points out that such solutions are generally regarded as "exact" even when the ODE must be solved numerically. This is therefore an example where an exact transformation of a problem (PDE to ODE) is combined with approximation methods (numerics) to render solution feasible. To see that this sort of scenario is very general, notice that even for uncontroversial cases of exact, closed form solutions, numerical methods are generally required to evaluate the functions used to express solutions in closed form. Consider for example the difference in feasibility of information for the solution of the Hydrogen atom problem in quantum mechanics as expressed in terms of spherical harmonics and (associated) Laguerre polynomials versus

[7]Note that the motivation for the original development of these theories was generalization and proof of important theorems, so the problem that is made more feasible by changing to an equivalent or more general formulation is not necessarily a computational one. It is nevertheless an inferential problem that is made more feasible, which as was pointed out in the introduction, can be regarded as a form of computation.

[8]Noting such a situation in his laying the foundations of the calculus of variations, Euler wrote: "It is thus possible to reduce problems of the theory of curves to problems belonging to pure analysis. And conversely, every problem of this kind proposed in pure analysis can be considered and solved as a problem in the theory of curves" [18, p. 399]. Euler preferred a geometric approach, but as a result of Lagrange's analytic formulation that proceeded directly from Euler's work, the analytic approach has become standard.

computed plots of the 3 dimensional probability distributions.

The process of making problem solving feasible, therefore, involves in general a *sequence* of transformations of the problem, each of which makes solution more feasible, reducing the complexity of computing a solution. Feasibility requires that this process terminate after a small number of steps where one actually obtains a solution to one of the simplified problems. Now, since the purpose of transforming the problem is to solve the original problem, the final stage of the process is to back-interpret the computed solution through the sequence of simplified problems so that it yields a solution or an approximate solution to the original problem, or a problem sufficiently close to it. This recursive process of problem simplification to a feasibly solvable problem, followed by back-interpretation of the result is the characteristic pattern of what I call feasible computation.

We have seen several examples now of this pattern of feasible computation, although they have generally focused on a single step of the iterative process or have left the sequence of steps implicit. To make the structural pattern clear, consider the following illustrative but simple example of the process. This is the use of logarithmic tables to render arithmetic calculations feasible. This method of calculation was developed in the seventeenth century by John Napier, Joost Bürgi and Henry Briggs, and used logarithms to convert multiplication, division, exponentiation and root extraction problems, respectively, into addition, subtraction, multiplication and division [10]. This approach to simplifying arithmetic underlay the use of the slide rule for arithmetical calculations until the development of inexpensive pocket scientific calculators supplanted its use in the seventies. This is an interesting example because an exact equivalence of problems is the basis of the method, but it is nevertheless a tool for accurate approximation calculation because of how the tables were constructed and used.

The transformation of an arithmetic problem is based the basic property of the logarithm that it converts products of its arguments into sums of their logarithms, *i.e.*,

$$\log(xy) = \log(x) + \log(y)$$

from which it follows that

$$\log\left(\frac{x}{y}\right) = \log(x) - \log(y), \quad \log(x^n) = n\log(x), \quad \log(\sqrt[n]{x}) = \frac{\log(x)}{n}. \tag{5.5}$$

Thus, one makes multiplication or division of quantities more feasible by computing their logarithms, adding or subtracting the result, and then finding the quantity whose logarithm equals the sum or difference. If one had infinite precision, then this would be an exact calculation. Logarithmic tables, however, can only have so many values in them, and it was extraordinarily tedious to compute them. So much so, that only certain values were computed directly, and the intermediate values were computed by interpolation. In fact, methods of interpolation developed a great deal through the need to compute more accurate tables [10]. Thus, one in fact replaced an arithmetic problem with an approximately equivalent one. Further approximation is involved in the use of the tables, since the result one computes may not equal an element of the table exactly, so one finds an approximate intermediate result.

We see from this process all of the features of the general pattern of feasible computation. First of all, there is the mapping of a problem that is difficult to solve directly, computing products or quotients, to an approximately equivalent problem of summing or subtracting their logarithmic table values, and then back-interpreting the result to the approximate product or quotient of the quantities. We even find the iterative pattern when a table is used for exponentiation and root extraction. In this case, the problem maps to a product or a quotient of quantities according to (5.5), which can then reduce to a sum or difference by applying the table a second time, followed by two back-interpretations to yield the result. Thus, the use of logarithmic tables provides an early example of the pattern of feasible computation.

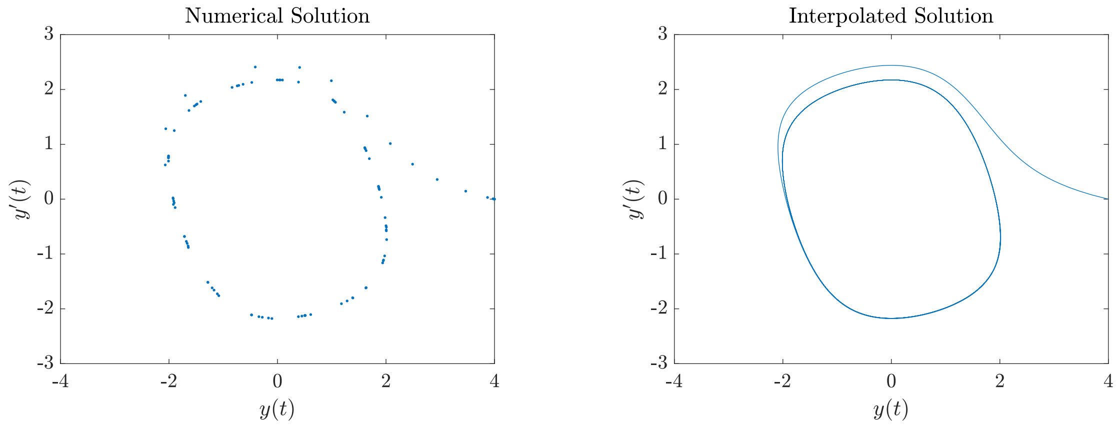
Logarithmic tables are such a nice example because the pattern of transforming the problem iteratively, computing of the solution, and back-interpreting the result is so clear. Moreover, the motivation of the method, *viz.*, to reduce the complexity of arithmetical calculations, typically for the purposes of applying theoretical models to solve practical problems, such as astronomical prediction and navigation, is so clearly tied to feasibility. In most cases the feasible computation pattern is implicit, but nevertheless present, in strategies to simplify problem solving. It does also not always involve back-interpretation to the original problem, if some simpler problem or model suits the scientific purposes at hand. A nice example of this is post-Newtonian celestial mechanics, which uses corrections to Newtonian gravity from general relativity (GR) to model the motion of bodies in weak gravitational fields. This is a simplification from general relativity, but the results are not mapped back to the framework of GR since it is easier to model things in terms of vector mechanics on a fixed background space, rather than using the more complex tensor framework needed for relativity.

We can see at this point that features of the feasible computation strategy are present wherever a problem is modified to simplify its process of solution. In the examples considered so far, with the exception of logarithmic tables, the full iterative aspect of repeated problem simplifications is typically not performed by a single individual. Most of the cases we have considered, one person is doing one kind of reduction, leaving any further reduction to be performed by someone else, or different levels of simplification are divided between theoretical and practical work. This situation has changed dramatically as a result of the development of advanced computing machines, which have made it possible to offload tedious arithmetical calculations to microprocessors, allowing scientists to focus on scientific inference. It has also led to new branches of science, in particular computational science, a multidisciplinary field concerned with developing algorithms to solve difficult mathematical problems as efficiently, accurately and reliably as possible, using a combination of exact and approximate methods. An important point for us is that computational science uses the recursive feasible computation process to construct algorithms that can solve mathematical problems *automatically*.

Modern scientific computing uses the feasible computation strategy to great effect by reducing difficult mathematical problems to combinations of simpler problems that can be solved rapidly and accurately with well-understood error behaviour.[9] A large proportion of mathematical problems can be reduced to problems in linear algebra and polynomial arithmetic. Breaking down problems in this way then allows highly optimized low level algorithms to be applied to these simplified problems, so that the results can be combined (back-interpreted) to generate fast, accurate solutions to the original problem. One important such package is the basic linear algebra subprograms (BLAS), an optimized package for low level linear algebra computations orignally written in Fortran in the seventies, which still operates (sometimes in a further optimized form) under the hood of most systems that do numerical calculations, including the mathematical computing environments MATLAB, MAPLE and MATHEMATICA.

To make it more clear how advanced algorithms for solving mathematical problems do indeed follow the feasible computing pattern, let us consider an example of how one would go about solving an ordinary differential equation in MATLAB. For concreteness,

[9]It is worth noting in passing that users of most computational software still need to do their own error analysis, and cannot blithely accept a computer generated result as correct if the software provides no explicit guarantees on accuracy or reliability.



(a) Sequence of phase space values produced by `ode45`. (b) Interpolant of the numerical skeleton from (a).

Figure 5.7: Solution of the van der Pol equation (5.6) for $\mu = 1$, $y_0 = 0$, $y_0' = 4$ for $t \in [0, 40]$ using Matlab's `ode45` routine.

suppose that we wish to solve the van der Pol equation

$$y'' - \mu(1 - y^2)y' + y = 0, \quad y(0) = y_0, y'(0) = y_0', \tag{5.6}$$

for the case $\mu = 1$. Due to the nonlinearity (the $\mu y^2 y'$ term) we do not have analytic solutions for this problem, which forces us to use an approximate method. For examining arbitrary initial conditions we need to simulate the solution, hence we seek a numerical solution. Solving this equation in Matlab is made extremely simple. With a standard reformulation of the problem as a coupled pair of differential equations, we can write four lines of code (easily reduced to two) to solve the equation:

```
f = @(t,y)([y(1)-y(1)^3/3-y(2);y(1)]);
tf = 40;
init = [4;0];
sol = ode45(f,[0,tf],init);
```

For these values of `tf` and `init`, the final time and initial condition,[10] this code executes in less than one hundredth of a second. With two more lines of code, within another hundredth of a second, we can interpolate the sequence of values y_n produced by the numerical method, the results appearing in figure 5.7. With this simple procedure, we find a stable limit cycle solution of the oscillator. Imagine trying to do the same calculation by hand using numerical methods and tables of logarithms.

This simple procedure masks the recursive feasible computation structure, which is

[10]Note that `init` is a vector, with the first component being y_0 and the second y_0'.

all packaged into the Matlab routine `ode45`. This procedure hides a sequence of three transformations of the problem. As we saw in the previous section, the first step is to replace the differential equation with a difference equation, to obtain approximate values y_n of the solution at times $t_n = t_{n-1} + h_n$, where the time step h_n can now vary from step to step.[11] But these difference equations are also rarely solvable analytically, and can include implicit equations that require iterative methods to solve. Thus, we take a second problem simplification step, which involves a move to an implementation of the numerical method in the code of a high level programming language. This transformation has some obvious aspects, such as translating from a mathematical to a programming language, and some subtler aspects, such as replacing the continuum of real numbers with a finite bounded system of floating point numbers, a simulation of the real numbers that a computer can calculate with. Thus, it really is a transformation of the problem involving approximation, not simply a reformulation. But this step does not produce a solution either, since the code needs to be run, which requires a third transformation of the problem into machine code, which can be run on a microprocessor.[12] It is only when the code is run that a solution is computed, when the solution finally is made feasible. The computed solution is a sequence of binary strings, which is then back-interpreted into decimal numbers, which can be interpreted as specifying points in phase space. When we interpolate the skeleton of values, we then obtain a continuous curve that is our approximate solution to the original equation (see figure 5.7).

We see, therefore, that running six lines of Matlab code involves a sequence of problem simplification stages that systematically make solution of the problem feasible, followed by a back-interpretation of the result through the sequence of transformations to obtain an approximate solution to the original problem, completing the inference. The approximation or translation at each step increases the feasibility of solution. We first replace a solution that varies continuously (differential equations), with one that varies discretely (difference equations), which in principle can be computed step by step, an increase in feasibility. Since these equations are rarely solvable in practice, we replace the numerical method with a fully algorithmic procedure written in code, which in principle can be computed by inputing the data for the problem and running the algorithm, a

[11] The routine `ode45` actually uses a pair of numerical methods that work together to approximate the error and take a step of the integration, as well as sophisticated step size control to control error and take as few steps as possible.

[12] Matlab actually uses a combination of compiled kernel code and interpreting of the code the user enters to produce machine code, so a more detailed analysis of problem transformation is possible. The net result, however, is a transformation from high level code to machine code.

further increase in feasibility. Since such computations are extraordinarily tedious, and error prone if performed by a human, we translate the algorithm into machine language so that a microprocessor can execute it, which then makes the solution feasible in the conditions we encounter in scientific practice.

A similar structure appears in any case of the use of numerical methods to solve problems on computing machines, whether we are solving a single linear equation or nonlinear partial differential equations for heterogeneous materials. Thus, the pattern of feasible computation is actually fundamental to scientific computing. It is not restricted to numerical computing, however, appearing also in exact, symbolic computation. Given that pure mathematics uses transformations of problems to equivalent ones to increase feasibility, as pointed out above, it may not be surprising that the feasible computing pattern appears in exact computation. What may be surprising, however, is that exact algorithms can involve forms of error and approximation.

In exact computation, algorithms generally come down to arithmetic of integers, polynomials and matrices, avoiding the floating point numbers used in numerical algorithms, which use approximate arithmetic. Doing arithmetic in the domains in which calculations are formulated, however, can be very computationally expensive and can lead to the problem of intermediate expression swell, where the problem and solution might be fairly simple but the intermediate expressions can be large, requiring large amounts of time and space to compute. Thus, the standard approach is to use one of a number of reduction methods to project the problem into a finite domain, computing the result there and then mapping back the result to solve the original problem. This is precisely the feasible computation pattern, which is essential in computer algebra for reducing the time and space complexity of algorithms.

Even the simplest problems, such as arithmetic over the integers $\mathbb{Z}$, can use reduction methods. Intermediate expression swell can be avoided by reducing to equivalent operations over finite sets of integers modulo m, *i.e.*, $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$, which can be represented as $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$. Since the results stay within this finite set, memory use is tightly controlled, reducing complexity. For integers of size n, the algebraic complexity of integer arithmetic is $O(n^2)$, but this can also be reduced by moving to modular domains and further reduction.[13] For sufficiently large integers, fast Fourier transform methods can yield asymptotically fast integer arithmetic ($O(n^{1+\varepsilon})$, for any $\varepsilon > 0$) by encoding integers as polynomials with coefficients in $\mathbb{Z}_p$, where p is a well-chosen prime number,[14] comput-

[13]The 'size' n here is measured in terms of the number of bits in a binary representation.

[14]The nth primitive roots of unity in the complex numbers are the numbers $e^{2\pi i/n}$, but in modular

ing the product of the polynomials based on evaluations of the polynomials at primitive roots of unity and an interpolation of the result, and finally decoding the resulting polynomial to the integer solution. This process itself is seen to follow the feasible computing strategy. Further complexity gains can be made by reducing to modular domains $\mathbb{Z}_p$, where p is a machine word prime and arithmetic operations can be performed on the microprocessor in effectively a single clock cycle, which reduces operations to $O(1)$.

Methods that reduce to and perform operations in finite domains $\mathbb{Z}_m$ are called *modular methods*. These methods come in two general classes, which function in different ways. One works by computing multiple modular images of the input, performs the required calculation in each of these images, and then uses a result called the Chinese remainder theorem to combine (back-interpret) the results to obtain the solution to the original problem. The other works by computing a single modular image, performs the required calculation, and then "lifts" (back-interprets) the result, by computing a sequence of approximations using a process called the *Hensel construction*, to obtain the solution. Interestingly, the Hensel lifting process is based on Newton's method for numerical rootfinding. The single modular image becomes the initial guess of the iterative process. This initial guess can be regarded as the zeroth order approximation in a power series (in powers of a prime number p) expansion of the solution. The iteration therefore computes increasingly higher order approximations until the exact solution is reached. Modular methods can be used for many problems, including fast polynomial arithmetic, factorization, integration and solution of linear difference and differential equations [9].

Although the sense of 'approximation' here is analogical and does not involve error in the manner that numerical or asymptotic approximations do, some uses of the Hensel construction nevertheless introduce error, so that there is actually a loss of information in the use of the feasible computation process, just as in the case of numerical computation. A sophisticated algorithm developed by Dahan *et al.* [6] for solving nonlinear systems of polynomials makes complexity gains by using a *probabilistic* modular method. The algorithm considers the case where the solution is a set of points, each described as a special set of polynomials called a triangular set. By accepting a small probability that multiple solutions can be mapped to the same modular image, so that when the result is lifted only one of the corresponding solutions is computed, the complexity can be reduced dramatically, rendering the solution of a large number of nonlinear systems feasible. This failure probability can actually be chosen, increasing the complexity as it

fields $\mathbb{Z}_p$ they are distinct elements ω of the field that have the property $\omega^n = 1$. Only for certain values of p will there exist such distinct elements.

is reduced, much like an error tolerance in numerical computation, showing that some instances of the feasible computation pattern in exact computation, share a surprising degree of structure. This algorithm is implemented for solving nonlinear systems in Maple.

In an analogous manner to the implementation of numerical methods, the implementation of symbolic algorithms involves stages of transformation of the problem so that the algorithms can ultimately be reduced to machine language and the feasibly computed results back-interpreted to solve the original problem. Also analogously to numerical methods, which so often reduce to floating point polynomial and linear algebra computations, symbolic methods often reduce to multiprecision integer polynomial and matrix algebra. In recognition of this, there are now efforts to produce optimized low level tools for symbolic computation. One such project, on which the author is a developer, is the basic polynomial algebra subprograms (bpas), which aims to be a symbolic analogue to blas (see [3]).

We now see how the advent of high performance computing machines has allowed the feasible computing strategy to be employed to great effect to solve mathematical problems with high efficiency and reliability. The feasible computing strategy is used in both the symbolic and numerical branches of scientific computing, involving in both cases the reduction of "high level" problems to combinations of optimized low level algorithms with high efficiency implementations, that allow solutions feasibly computed by microprocessors to be back-interpreted to high level solutions. Thus, a strategy that has its roots in the history of physics and pure mathematics now allows computing machines to greatly expand the range of mathematical problems that are feasibly computable, hence greatly expanding the range of feasible scientific inference.

5.4 Consequences for Computability Theory

The contrast between numerical and symbolic computing offers a way of bringing out the implications of feasible computing in computational science for the theory of computation. One consequence is brought out in the common structure between numerical and symbolic computing. In both cases we find the main structure of transformations that increase the feasibility of solution of problems. As such instances of the feasible computing strategy are "higher order" algorithms, in the sense that they are not simply concerned with finding optimal algorithms for a particular problem, they seek to find optimal *transformations* between problems such that the overall algorithm has minimal computational complexity. When we consider that the full feasible computation process involves implementation of mathematical algorithms in a software environment and running of machine code, the process also involves transformation between *kinds* of problems, not simply mappings between problems of a single sort. If we regard problems as points in a space, then feasible computing reduces complexity by strategically moving between points within the space, and between spaces for the implementation process. Given the importance of the feasible computing structure in the solution of mathematical problems, the development of a theory of such higher order computation stands to be of significant value.

A second consequence, is one that is brought out by the difference in structure between numerical and symbolic computing. The main difference between numerical and symbolic computing versions of the feasible computing strategy is that the numerical version is more general. This is because approximations break the exact structure preserved in symbolic computing, but if we restrict the strategy by reducing possible approximations to zero we get the symbolic version. The theory of computation is geared toward the exact case, where we can talk about the space $\mathcal{A}_p$ of all algorithms to solve a particular problem p and the minimal complexity in that space. The symbolic feasible computing strategy provides an effective means of *finding* minimal complexity algorithms, or approximations to them, but the algorithms generated stay within $\mathcal{A}_p$. In contrast, the numerical version involves moving to a different problem p' with its own space $\mathcal{A}_{p'}$ of algorithms, which can have a different minimal complexity. Thus, we can make complexity gains by allowing moves to nearby problems, provided the nearby problem provides essentially the same information as the original one, at least for the purposes for which the algorithm is being used. It is not clear how generally this is merely a strategy that allows us feasible *access* to lower complexity algorithms, and how generally the spaces $\mathcal{A}_p$ and $\mathcal{A}_{p'}$ really have different lower complexity bounds. Nevertheless, this leads to a different way of thinking about computational complexity in computation.

A thought experiment can help to clarify the nature of the issue here. Suppose that the problem p that we wish to solve is only computable in exponential time, but that, in some neighbourhood of p within a natural space of problems, the problems p' generically have polynomial time complexity. Does it still make sense to consider p to have exponential complexity if the problems p' have no discernible differences in the information contained in their solutions? For this matter to be of serious concern, the lower complexity bounds of $\mathcal{A}_p$ and $\mathcal{A}_{p'}$ must really be different. Even if they are not, because there exist equally low complexity algorithms we do not know about, an epistemological version of the same problem results from the use of feasible computing.

A final consideration for computability theory concerns the larger pattern of feasible computation in scientific inference processes. We traced in some detail the roots of the general, approximate feasible computation process to approximation methods in physics. We also saw in a more limited way that the exact version of the process has roots in pure mathematics. The result is that the feasible computation pattern is found extremely broadly, whenever a mathematical problem is not solvable in its original form, whether the solution process sought is purely analytical or involves scientific computation. Since the method has this epistemological drive, the need to overcome an inferential obstacle to draw feasible consequences, it is likely a very general process in scientific inference in general, possibly involving weaker forms of the strategy of modifying problems to compute feasible solutions in fields that are less mathematized. Given that computability theory and the analysis of algorithms rely heavily on asymptotic methods, they are certainly using a form of the feasible computation pattern. Given that machine learning algorithms can generate reliable but approximate solutions to various problems, it is likely they are using a form of the feasible computation pattern as well. If the feasible computation pattern is found more broadly in the practice of science, then an extension of computability theory to treat higher order approximate computation presents the possibility of extending the theory of computation, and possibly even applying advanced algorithms through developments in computational science, to a theory of the methodology and epistemology of scientific practice.

5.5 Bibliography

[1] Konstantin E Avrachenkov, Jerzy A Filar, and Phil G Howlett. *Analytic perturbation theory and its applications.* SIAM, 2013.

[2] Richard Ernest Bellman. Perturbation techniques in mathematics, physics, and engineering. 1964.

[3] Changbo Chen, Svyatoslav Covanov, Farnam Mansouri, Marc Moreno Maza, Ning Xie, and Yuzhen Xie. Basic polynomial algebra subprograms. *ACM Communications in Computer Algebra*, 48(3/4):197–201, 2015.

[4] Robert M Corless. Error backward. In *Chaotic numerics: an International Workshop on the Approximation and Computation of Complicated Dynamical Behavior, Deakin University, Geelong, Australia, July 12-16, 1993*, volume 172, page 31. American Mathematical Society, 1994.

[5] Robert M Corless and Nicolas Fillion. *A graduate introduction to numerical methods*. Springer, 2013.

[6] Xavier Dahan, Marc Moreno Maza, Eric Schost, Wenyuan Wu, and Yuzhen Xie. Lifting techniques for triangular decompositions. In *Proceedings of the 2005 international symposium on Symbolic and algebraic computation*, pages 108–115. ACM, 2005.

[7] Olivier Darrigol. *Worlds of flow: A history of hydrodynamics from the Bernoullis to Prandtl*. Oxford, 2005.

[8] Anindya De, Piyush P Kurur, Chandan Saha, and Ramprasad Saptharishi. Fast integer multiplication using modular arithmetic. *SIAM Journal on Computing*, 42(2):685–699, 2013.

[9] Jürgen Gerhard. *Modular algorithms in symbolic summation and symbolic integration*, volume 3218. Springer, 2004.

[10] Herman H Goldstine. *A History of Numerical Analysis from the 16th through the 19th Century*. Springer-Verlag, 1977.

[11] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[12] Morris Kline. *Mathematical Thought From Ancient to Modern Times: Volume 1*, volume 1. OUP USA, 1990.

[13] Morris Kline. *Mathematical Thought From Ancient to Modern Times: Volume 2*, volume 2. OUP USA, 1990.

[14] Morris Kline. *Mathematical Thought From Ancient to Modern Times: Volume 3*, volume 3. OUP USA, 1990.

[15] Richard L Liboff. *Introductory quantum mechanics*. Addison-Wesley, 2003.

[16] Chris M Linton. *From Eudoxus to Einstein: a history of mathematical astronomy*. Cambridge Univ Pr, 2004.

[17] Wilfred Sieg. Formal systems, properties of. In R. A. Wilson and F. Keil, editors, *The MIT Encyclopedia of the Cognitive Sciences*. 1999.

[18] Dirk J Struik. *A source book in mathematics, 1200-1800*. Harvard University Press, 1969.

[19] Milton Van Dyke. Perturbation methods in fluid mechanics. *NASA STI/Recon Technical Report A*, 75, 1975.

[20] Pieter Wesseling. *Principles of computational fluid dynamics*, volume 29. Springer Science & Business Media, 2009.

CHAPTER 6

Conclusion: Feasible Computation in the Mathematical Sciences

Conclusion: Feasible Computation in the Mathematical Sciences

6.1 Concluding Remarks

A central theme throughout this dissertation is the overcoming of obstacles to making reliable inferences in computational science. In chapter 5 we introduced the term *feasible computation* to identify the notion of reliable and efficient computation subject to the contextual demands experienced in practice. In these terms, then, a central drive in computational science is overcoming obstacles to feasible computation.

In the context of symbolic computing, one of the obstacles to feasible computing is conceptual: since symbolic computing is generally regarded as exact computation, it is a common belief that no error is introduced. As pointed out above, Corless and Jeffrey have worked throughout their careers to point out that failure to faithfully respect the continuity of analytic objects that are treated algebraically leads to error in symbolic computing, and underlies their introduction of the unwinding number in [5] to restore continuity at branch cut crossings. It is precisely the elimination of this kind of error in symbolic integration that motivates chapter 2. It was shown that the introduction of a second kind of unwinding number to restore continuity at branch point crossings can restore continuity for integrals containing the natural logarithm and arctangent functions, which are central in the symbolic integration problem on account of Liouville's theorem. This second unwinding number is called the *radial unwinding number*, to distinguish it from the *angular unwinding number* that restores continuity at branch cut crossings. It was then demonstrated that the unwinding numbers needed to restore continuity for the integrals of rational functions are computable, and explicit formulae are given for their evaluation in the case of ordinary rational functions and rational functions of a transformed variable.

A related obstacle to feasible computing in the integration problem is how to conceptualize complex functions. The multivaluedness of many elementary complex functions is a source of confusion to all students who are introduced to complex function theory, and it has proved to be a challenging problem in the development of computer algebra systems. Although multivaluedness of complex functions has well-developed theoretical treatments in terms of Riemann surfaces (correcting discontinuity in the domain) and branch cuts (correcting discontinuity in the codomain), both approaches present significant challenges in symbolic computing. Another contribution of chapter 2 is to point out that by employing a hybrid approach that treats the codomain as a two-dimensional complex manifold (or pair of two-dimensional real manifolds constituting the real and imaginary parts) we can define an antiderivative for meromorphic functions. The angular and radial unwinding numbers then make continuous paths on the codomain manifold(s) computable. This has the benefit of allowing contour integrals to be computed by computing the algebraic antiderivative and then evaluating the antiderivative (corrected for continuity) along the contour, rather than having to integrate the parameterization of the contour.

Another obstacle to feasible computing in symbolic computation is practical: since so much of symbolic computing is designed for interactive computer algebra systems, few open software tools are available for symbolic computing that are suitable for integration into large scientific computing projects. In our view, one of the reasons for this limitation is the common view that symbolic computation is exact computation. Insofar as symbolic computation is needed for computational modeling, which always introduces forms of error in the modeling process, the introduction of error in computation is acceptable provided it is small compared to other sources of error in the modeling problem [10]. In chapter 3 we show that the combination of efficient symbolic computation (in BPAS [3]) with multiprecision numerics (in MPSolve [1]) allows tolerance proportionality for the symbolic-numeric integration of rational functions. Among other possible applications, this software can be used to compute high precision integrals of rational approximations, even near singularities of the integrand.

We provide two algorithms that preserve the structural information in the exactly specified integrand, one that approximates the integrand by computing the partial fraction decomposition (PFD) and then integrates, and the other that computes the integral exactly, using the Lazard-Rioboo-Trager (LRT) algorithm, and approximates the algebraic numbers in the result. It is shown that the PFD-based approach does not present significant costs in terms of efficiency or stability, providing the overall superior algorithm. The LRT-based approach has the advantage of retaining the exact integral for further symbolic computation, while providing efficient and stable access to a seminumerical integral.

A related obstacle to feasible computing on this problem is the availability of knowledge of the reliability of the result. There are general methods of backward error analysis (see [4]) that can provide clear information concerning the reliability of a computation in numerical computing. In some cases, however, this analysis is not done on account of the error analysis interfering with the efficiency of the algorithm. We address this issue for the symbolic-numeric integration of rational functions by designing the error analysis in such a way that it can be computed in parallel during the main integration algorithm. In this way, the reliability information becomes part of the computation and interferes minimally, if at all, with efficiency. We provide some experimental data to show that the PFD-based approach accomplishes this task without extending the runtime, and that the LRT-based approach does so except on non-generic type problems that can be integrated quickly without generating large coefficients in the polynomials that define the algebraic numbers in the result.

Both chapters 2 and 3 demonstrate the need for new ways of thinking about error in computational science, particularly in symbolic computation. Chapter 4 contributes to this by identifying the basic logical structure of feasible computing. This is so because inference can be regarded as a form of computation and *vice versa*, as is pointed out in the introduction of chapter 5. It is shown how the basic idea of stable inference can be made precise in terms of a generalized concept of logical validity, which we call *effective validity*. This concept can also be seen as a generalization of the concept of continuous function, since it encapsulates the idea that stability requires that micro-local variation of the input of a function must lead to micro-local variation of the output. The formulation is intended to be as general as possible, so as to include not only scientific inference but eventually informal inference in general. A consequence of this generality, is that the concept of stable inference can apply to both numerical and symbolic computing. Indeed, the connection between stability and continuity identified in the concept of effective validity makes it clear why discontinuity is a form of error in symbolic computing, while subsuming numerical error under the same concept of instability.

The languages we work with in computational science, whether mathematical or programming languages, are generally interpreted, *i.e.*, where we have a definite syntax and a definite intended semantics. Where error can be introduced syntactically or semantically, the languages must be stable under syntactic and semantic variation. Languages where the syntax and semantics covary with perturbations of the syntax or semantics are defined in chapter 4 to be syntacto-semantically stable. Underlying the feasible computation pattern identified in chapter 5, then, are transformations between syntacto-semantically stable languages that make desired information more feasibly accessible. In chapter 5, the feasible computation pattern was identified as a recursive strategy of transformations of a problem that increase feasibility, reaching a problem that is feasibly computable, followed by back-interpretation of the result. In terms of effective logic, the transformations of the problem leading to a feasibly computable one must be forward inferentially stable; to be able to back-interpret the result, the transformations must also be backward inferentially stable. Thus, together, chapters 4 and 5 show that the strategy in computational science of efficiently solving problems in a reliable way involves the search for inferentially stable transformations of mathematical problems that reduce computational complexity. Chapter 4 then makes the point that nothing about the logical structure of this requires that mathematical or conceptual content be preserved in these transformations, only that the inferential structure is preserved, so that a feasibly computed solution can be back-interpreted to an effectively valid solution to the original problem.

In chapter 5 two forms of the feasible computation strategy are pointed out, one involving transformations of problems without relevant loss of information, being the form employed in pure mathematics and exact computation, and another where there is information loss, the form employed in applied mathematics and approximate computation. The logical distinction between these two kinds of transformation, one involving exact structure-preservation (homomorphism and isomorphism) and the other involving approximate structure-preservation (conceived in terms of new concepts called *continomorphism* and *contisomorphism*) is pointed out in chapter 4. Since the feasible computation form with information loss is the more general of the two, it is the focus of chapter 5. The historical roots of this strategy in methods of getting approximate, but reliable, information out of physical theories is examined. We then point out that computational science constructs feasible algorithms by chaining together stable transformations of problems that reduce computational complexity (inferentially stable transformations), providing numerous successful examples of this strategy. It is then argued that where accessible nearby problems have different complexity than the original problem, in such a way that error does not significantly affect the result, this should affect how we understand the computational complexity of the original problem. In this way, considering a computational modeling context changes how we understand not only error but also complexity.

It is seen, therefore, that chapters 4 and 5 elucidate the epistemology of computational science in highly entangled ways. Together they reveal a computational strategy that is replicated surprisingly widely in science. Chapter 5 reveals that the epistemological motive of the strategy is to overcome obstacles to inference that are encountered in scientific practice, *viz.*, when we encounter an inferential obstacle, we transform the problem in some way to make the information we seek accessible, either exactly or approximately. Given the genericity of this epistemological motive, it is expected that the feasible computation strategy, and inferentially stable transformation of languages, will be found very generally in scientific inference. It is hoped that this insight will help to cut through the apparent diversity in scientific thought to reveal deep underlying symmetries in method and ideas. Restricted to the context of scientific computation, it is hoped that a clarification of the underlying logic of successful strategies for feasible computation will lead to the development of new and better algorithms, and software design that promotes reliable belief-formation and supports reliable decision-making.

6.2 Future Work

There are a number of directions in which the research presented in this dissertation can be extended and developed. The natural extension of the unwinding number approach to continuous symbolic integration is to extend the radial unwinding number to branch points of algebraic and restricted classes of transcendental functions, and to then consider the computability of jump conditions in terms of symbolic integration algorithms for algebraic and transcendental functions, including the methods in [2]. As is mentioned in the conclusion of chapter 2, extending the theory to algebraic functions will require a treatment in terms of Puiseux series rather than Laurent series, since algebraic functions generally do not have Laurent series expansions at their branch points. A treatment in terms of Puiseux series then also opens up line and contour integration over algebraic curves of positive genus, *viz.*, those curves that do not have rational parameterizations.

The work in chapter 3 on multiprecision symbolic-numeric integration on exact input extends naturally in two different directions. One is to consider symbolic integration algorithms for algebraic and transcendental functions. As described in [2], the core of the transcendental integration algorithms relies on residue reduction, which produces similar formal sums over the roots of an irreducible polynomial. Thus, there is room to extend the approach of chapter 3 to transcendental function integration. A research question raised here is whether the structured PFD approach, which was shown to have a number of advantages over the LRT approach, can be stably employed to integrate rational functions in the process of integration of integrands in transcendental differential extension fields of $\mathbb{Q}[x]$. Assuming efficient algorithms of this type can be constructed, another research question is whether the error analysis can be designed in a similar way to be executable concurrently with the main integration. Extending the approach to transcendental integration, together with an efficient implementation in BPAS, would open up multiprecision symbolic-numeric integration to a much wider array of problems encountered in applied mathematics.

A second direction in which the work in chapter 3 can be extended is to consider rational functions with approximate input rather than exact input. This is the assumed input form for the original hybrid integration methods for rational functions developed by Noda and Miyahiro [13, 14]. With the existence now of sophisticated tools for approximate polynomial algebra, such as [17], the extension of our approach to approximate input could lead to robust and effective methods for approximate integration. A potential strategy would be to search for the nearby most singular problem, which will be a nearby problem with additional structure, and employ the structured PFD-based integration algorithm presented in chapter 3. This approach puts the problem onto the perjorative manifold, which guards against ill-conditioning as described by Kahan [9], and the structured integration ensures that the problem stays (exactly or approximately, depending on the integration method) on the pejorative manifold.

In chapter 4 the concept of effective validity is presented as a logical model of inferential phenomena, but the formulation is intended to allow mathematically precise definitions to be constructed. A future direction of this work is then to consider rigorous formulations of effective logic using formal methods that are likely to make it applicable widely in areas of computational science and applied mathematics. One possibility is a formulation in terms of normed or metric spaces, treating micro-locality in terms of the traditional notion of a metric, or the metric induced by a norm. This approach should be well-suited to numerical computing and symbolic analysis. Another possibility is a formulation in terms of the semantics of programming languages, well-suited to computer and computational science more generally.[1] Particularly treated in terms of categorical semantics, this approach allows nearness to be treated in precise structural terms in a way that coheres with a type-theoretic syntax, given the deep connection between category theory and type theory in categorical logic [6].

There are then interesting connections to homotopy type theory (HoTT) (see [16] for a summary of the foundations of HoTT), indicated in footnote 9 chapter 4. Due to the fact that HoTT relies on exact equivalence (isomorphism) between objects, described in above in terms of micro-local equivalence extending to global equivalence, indicates that

[1]The author thanks Ursula Martin for this suggestion.

HoTT does not provide a faithful representation of general approximate near-preservation of structure. Nevertheless, there is a striking similarity between the concept of identity in HoTT, which treats isomorphic objects in terms of a single identity type, and objects in effective logic subject to near-identity transformations that are micro-locally identical. That near-identity transformations generate (isomorphic) equivalence relations only locally and relative to a type defined by the structure preserved over micro-local variation, there is a distinct sense in which the notion of identity in effective logic is locally similar to the notion of identity in HoTT. It seems at least possible to approximate the concept of contisomorphism in terms of (local) identity types in HoTT, where the type can change depending on the centre of variation and the type is defined implicitly by what structure nearby objects share, which may not be specifiable explicitly. A potential future research project, then, is to explore this possible connection between effective logic and HoTT.

Another future area of investigation for effective logic is to extend the basic conceptual framework to uninterpreted languages. This project would move toward tools for the analysis of stability properties of languages and their models, including the identification of stability concepts for theorems involving variations across mathematical contexts, as illustrated above through the example of the fundamental theorem of calculus. In addition, there is also the potential for modeling the structure of the use of mathematical languages in science where the syntax and methods are extended without a well-defined intended interpretation.

A prime example of this latter phenomenon is in quantum field theory (QFT), and advanced theoretical physics in general. Here there are numerous examples of the syntactic variation of a mathematical theory, term or concept with a clear semantics into contexts where there is no clear semantics, yet the overall strategy is phenomenally successful empirically. Examples of such moves beyond existing semantic ground include the strategies to construct quantum field theories from their classical analogues, the introduction of cutoffs, the many versions of Wick's theorem, *etc.* (see [15] for details). The fact that QFT relies heavily on the theory of Lie groups and Lie algebras, like many advanced theories in physics, means there are also explicit examples of (infinitesimal) near-identity transformations in the construction of the theory. For example, because relativistic quantum field theories must be representations of the Poincaré group, as a result the Hamiltonian, momentum, rotation and boost operators generate the Lie algebra of the Poincaré group, thus providing nice examples of infinitesimal near-identity operators. QFT therefore provides an interesting case where the concepts of effective logic are likely to clarify the logical structure of the theory and its methods.

Finally, there are a number of different ways in which the work of chapter 5 can be extended. Among these include a more exhaustive study of the feasible computation strategy in applied mathematics and computational science as well as the the extension to inference and computation methods in other areas of science. The existence of the pattern throughout methods of proof in pure mathematics, identified at the beginning of section 5.3 needs to be thoroughly investigated. Methods in modern homotopy theory, such as [7], provide a particularly striking example of the exact feasible computation strategy in mathematical proof, as has been shown in unpublished work. Aside from pure mathematics, the existence of the approximate feasible computation strategy pattern exists in computer science on account of the common use of asymptotic analysis and as a result of the use of heuristic algorithms, as is discussed briefly in [12], the publication version of chapter 5. Forms of the pattern in data handling in applied science have also been identified in [11] and unpublished work.

Aside from pure, applied and computational mathematics, the natural epistemological motive underlying the feasible computation strategy is likely to make forms of it fairly ubiquitous throughout science. Thus, another area of future work is to search for examples of the strategy in a variety of different areas of scientific inquiry. Aside from demonstrating the existence of the strategy in more general contexts, this work will also involve a study of the forms of variation of the inference strategy and what this tells us about the particular sciences in which the given form of the strategy is used.

A last area where the work of chapter 5 could be extended is to investigate the possibility of a precise generalized notion of computational complexity that takes into account the ability of approximation to change the complexity class of a problem. The ability of numerical methods to make practically uncomputable problems “computable with lightning speed”, to quote Trefethen, is one of the celebrated features of contemporary scientific computing. The successful employment of heuristic algorithms and machine learning in increasingly many contexts shows the ability of forms of approximation to reduce computational complexity in ways that make feasible solutions to problems that would otherwise be infeasible. Since there is a connection between complexity reducing transformations of problems and inferentially stable transformations of languages in effective logic, the concepts of effective logic may be useful in the interest of developing a generalized theory of computational complexity.

6.3 Bibliography

[1] Dario A Bini and Leonardo Robol. Solving secular and polynomial equations: A multiprecision algorithm. *Journal of Computational and Applied Mathematics*, 272:276–292, 2014.

[2] Manuel Bronstein. *Symbolic integration I: Transcendental Functions*, volume 1. Springer, 2005.

[3] Changbo Chen, Svyatoslav Covanov, Farnam Mansouri, Marc Moreno Maza, Ning Xie, and Yuzhen Xie. Basic polynomial algebra subprograms. *ACM Communications in Computer Algebra*, 48(3/4):197–201, 2015.

[4] Robert M Corless and Nicolas Fillion. *A graduate introduction to numerical methods*. Springer, 2013.

[5] Robert M Corless and David J Jeffrey. The unwinding number. *ACM SIGSAM Bulletin*, 30(2):28–35, 1996.

[6] R.L. Crole. *Categories for types*. Cambridge Univ Pr, 1993.

[7] Paul G Goerss and John F Jardine. *Simplicial homotopy theory*. Springer Science & Business Media, 2009.

[8] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[9] William Kahan. Conserving confluence curbs ill-condition. Technical report, DTIC Document, 1972. http://www.dtic.mil/get-tr-doc/pdf?AD=AD0766916, Accessed: 2017-12-19.

[10] Robert HC Moir. Reconsidering backward error analysis for ordinary differential equations. M.sc. thesis, The University of Western Ontario, 2010.

[11] Robert HC Moir. *Structures in real theory application: A study in feasible epistemology*. PhD thesis, The University of Western Ontario, 2013.

[12] Robert HC Moir. Feasible computation: Methodological contributions of computational science. In *Physical Perspectives on Computation, Computational Perspectives on Physics*. Cambridge University Press, 2018.

[13] Matu-Tarow Noda and Eiichi Miyahiro. On the symbolic/numeric hybrid integration. In *Proceedings of the international symposium on Symbolic and algebraic computation*, page 304. ACM, 1990.

[14] Matu-Tarow Noda and Eiichi Miyahiro. A hybrid approach for the integration of a rational function. *Journal of computational and applied mathematics*, 40(3):259–268, 1992.

[15] Michael E Peskin and Daniel V Schroeder. *An Introduction to Quantum Field Theory*. Westview Press, 1995.

[16] Vladimir Voevodsky et al. Homotopy type theory: Univalent foundations of mathematics. *Institute for Advanced Study (Princeton), The Univalent Foundations Program*, 2013.

[17] Zhonggang Zeng. Apatools: a software toolbox for approximate polynomial algebra. In *Software for Algebraic Geometry*, pages 149–167. Springer, 2008.

Curriculum Vitae

Robert Hugh Caldwell Moir

Education:

PhD, Applied Mathematics (with Scientific Computing)
Western University, London ON, Canada
2013–2017

PhD, Philosophy
Western University, London ON, Canada
2004–2013

MSc, Applied Mathematics
Western University, London ON, Canada
2009–2010

MA, Philosophy
Western University, London ON, Canada
2003–2004

BA, Philosophy and Mathematics (First Class Joint Honours)
McGill University, Montréal QC, Canada
2001–2003

BSc, Physics, (minor Chemistry) (First Class Honours)
McGill University, Montréal QC, Canada
1995–2001

Honours and Awards:

Government of Ontario, Western University
Queen Elizabeth II Graduate Scholarship in Science and Technology
2015–2016

University of Pittsburgh
Visiting Scholar
2011–2012

Social Sciences and Humanities Research Council of Canada
Doctoral Fellowship
2007–2009

Related Work Experience:

Instructor
Department of Philosophy
Western University, London ON, Canada
2010–2013

Teaching Assistant
Department of Applied Mathematics
Western University, London ON, Canada
2009–2010, 2013-2017

Publications:

Moir, RHC (forthcoming 2018). "Feasible Computation: Methodological Contributions from Computational Science." In: Cuffaro, M and Fletcher, S (Eds.), Physical Perspectives on Computation, Computational Perspectives on Physics, Cambridge University Press.

Chen, C, Covanov, S, Mansouri, F, Moir, RHC, Moreno Maza, M, Xie, N and Xie, Y (2016). "The basic polynomial algebra subprograms." ACM Communications in Computer Algebra, v. 50, no. 3, pp. 97-100.

Moir, RHC, Corless, RM, Jeffrey, DJ, (2014). "Unwinding Paths on the Riemann Sphere for Continuous Integrals of Rational Functions." In: Elias, J, Fernández-Sánchez, J, and Sombra, M (Eds.), Proceedings de Encuentro de Álgebra Computacional y Aplicaciones (EACA) XIV (EACA trans: Meeting on Computer Algebra and Applications), Barcelona, June 2014, pp. 139-142.

Programming Language Experience:

C, C++, Java

CilkPlus, MPI, CUDA

Maple, Matlab, Octave