TOWARDS MORE EFFICIENT INTERVAL ANALYSIS: CORNER FORMS AND A REMAINDER INTERVAL NEWTON METHOD

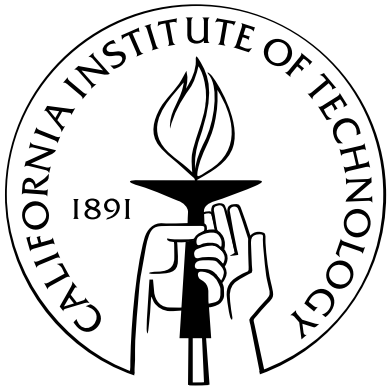
Thesis by

Marcel Gavriliu

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2005

(Defended May 26, 2005)

Acknowledgements

This work was supported by National Science Foundation grants #ASC-89-20219 and #ACI-9982273; the Office of the Director of Defense Research and Engineering, and the Air Force Office of Scientific Research (F49620-96-1-0471), as part of the MURI program; and the JPL/NASA Manifold Approximation project.

Abstract

In this thesis we present two new advancements in verified scientific computing using interval analysis:

1. **The Corner Taylor Form (CTF) interval extension.** The CTF is the first interval extension for multivariate polynomials that guarantees smaller excess width than the natural extension on any input interval, large or small. To help with the proofs we introduce the concept of *Posynomial Decomposition (PD)*. Using PD we develop simple and elegant proofs showing the CTF is isotonic and has quadratic or better (local) inclusion convergence order. We provide methods for computing the exact local order of convergence as well as the magnitude of excess width reduction the CTF produces over the natural extension.

2. **The Remainder Interval Newton (RIN) method.** RIN methods use first order Taylor Models (instead of the mean value theorem) to linearize (systems of) equations. We show that this linearization has many advantages which make RIN methods significantly more efficient than conventional Interval Newton (IN). In particular, for single multivariate equations, we show that RIN requires only order of the square root as many solution regions as IN does for the same problem. Therefore, RIN realizes same order savings in both time and memory for a significant overall improvement.

We also present a novel application of the two contributions to computer graphics: *Beam Tracing Implicit Surfaces.*

Table of Contents

List of Figures

Chapter 1

Introduction and Motivation

Over the last decade we have observed an increasing trend towards replacing costly real-life tests and experiments with computer simulations. From automobile crash tests to spacecraft trajectory planning to DOE's Advanced Simulation and Computing project (formerly known as ASCI) decisions that strategically affect our day to day lives are made based on the results of computer simulations. This emerging trend prompts the need for reliable and efficient self-verified computing methods that can guarantee prediction of results one hundred percent.

Traditional numerical computing uses IEEE floating point arithmetic (IEEE 754 standard). This floating point standard is widely supported in hardware; highly optimized math libraries (such as Intel's Performance Libraries) are readily available. Unfortunately, floating point procedures are not sufficient to guarantee correct results in all cases, as is demonstrated by a classic example by Rump, which we briefly review in the next section. Rump provides a simple rational expression designed so that evaluation using floating point fails to produce the correct result even when the number of digits of precision is doubled, and doubled again. This simple example shows that no algorithm using floating point alone can be relied on to make strategic decisions without risk.

Interval analysis was formally introduced by R. E. Moore in the 1960's, see [Moore 1962, Moore 1966]. It provides a natural framework for self-verified numerical computing with its ability to correctly and automatically account for errors from many sources, including rounding errors due to limited precision of the floating point representation of real numbers, approximation errors due to algebraic manipulation of formulas, and measurement error in the initial data.

1.1 Benefits of Interval Computations

Although not new, interval analysis has not found the widespread acceptance its creators had hoped for. The common belief is that there are faster, more straightforward methods that can account for rounding and other types of errors. For example, it is common practice to compute results independently in both single and double precision floating point and compare the digits of the two results. If the significant digits agree up to a certain precision then the matching digits are considered correct. However, it is relatively easy to design examples where the above method breaks. One such case is the classic example by Rump who, in 1988, published an expression for which numerical evaluation with floating point arithmetic gave erroneous and misleading results. When evaluating Rump's expression with increasing numbers of digits the results seemed stable as they agreed in their first few significant digits. However, as it turns out, all the digits were incorrect and, although the computed answer was relatively far from zero, it failed to even capture the correct sign. Rump's example is not reproducible on modern IEEE 754 computers. Fortunately, the following expression due to Walster and Loh, reproduced from [Hansen and Walster 2003], produces a similar outcome, this time using IEEE 754 floating point arithmetic:

$$f(x,y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

For $x = 77,617$ and $y = 33,096$ one would obtain the following results:

$$\begin{aligned} \text{32 bits:} \quad f(x,y) &= 1.172604 \\ \text{64 bits:} \quad f(x,y) &= 1.1726039400531786 \\ \text{128 bits:} \quad f(x,y) &= 1.172603940053178618588349045201838 \end{aligned}$$

In spite of their agreement in the first digits all three results are wrong. The correct answer is:

$$f(x,y) = -0.827396059946...$$

Evaluation using even the simplest form of interval analysis (natural extension) produces a wide interval that contains the correct value above. While not directly providing a better (point) answer, interval evaluation alerts us to the numerical instability of the expression and suggests that higher-accuracy methods need to be employed if the correct answer is to be computed.

Several real world examples of disasters caused by numerical instability of floating point are documented by Douglas N. Arnold on his website at:

`http://www.ima.umn.edu/~arnold/disasters/`,

as well as in [Hansen and Walster 2003]. All these disasters could have been easily avoided if interval analysis were used for validation.

Another common complaint is that interval methods are too slow to be useful in practice. While it is true that computing with intervals is inherently slower than computing with floating point numbers we have to make certain that we are comparing apples with apples. Often times, interval methods are the only ones capable of reliably solving the problem at hand. This is the case, for example, when solving general nonlinear global optimization problems. Another example is the computation of global solution sets of underdetermined systems of nonlinear equations. In such cases there is no competing floating point method—interval analysis is the fastest method available.

In the cases where a competing (non error bounding) floating point method does exist, methods using interval analysis will naturally be slower. The slowdown factor depends on many factors and varies greatly. Properly optimized interval algorithms are generally no more than one order of magnitude slower than their float counterparts. This is often a reasonable price to pay for the guaranteed error bounds produced by interval analysis. As interest in intervals grows so will the degree of refinement of implementation and the gap will continue to narrow.

Interval researchers have long argued that Moore's Law (computer performance doubles every 18 months) will make interval analysis practical. However, absolute speed is not always the correct benchmark. Rather, it is the speed differential between interval and floating point methods that keeps potential users away. Recent evidence seems to suggest we are already at

the limits of Moore's Law and we cannot count on CPU speeds doubling every 18 months. As a result, the intrinsic efficiency of the algorithms used becomes increasingly more important.

Use of the classic natural extension coupled with simple spatial subdivision is slow and produces unusable results for all but the simplest of problems, as can be seen in the examples in chapter 4. Such meager performance can be enough to convince people that all of interval analysis is inefficient and should be avoided. Fortunately, this is not the case with state of the art methods such as higher order interval extensions (Centered and Taylor Forms, Bernstein Forms, Taylor Models, etc.) coupled with quadratically convergent Interval Newton—very sharp bounds can be computed in reasonably fast times at the expense of rather complicated implementation costs.

The methods introduced in this thesis further improve the efficiency of interval methods. Corner Taylor Forms are the first interval extensions to guarantee smaller excess width than the natural extension when evaluated on large intervals while preserving the quadratic convergence properties of the Taylor Form. Remainder Interval Newton improves over classic Interval Newton with a special subdivision algorithm that extends its applicability to non-square systems while helping improve efficiency by a factor of the square root (fewer steps).

1.2 Thesis Overview

In this section we give a structural overview of the thesis.

The thesis has two parts. The first part is comprised of chapters 2 and 3. Its main objective is to provide an easy to read overview of the previous state of the art in interval analysis and to provide some of the motivation for the contributions we present in the second part of the thesis. Therefore, we have omitted all the proofs and instead concentrated on the relationships between the many concepts we discuss.

Chapter 2 reviews the most basic concepts of interval analysis. We purposely leave out the more advanced methods which will be discussed later in the thesis. The chapter begins with a short overview of notation in section 2.1; some new notation is introduced here. Intervals and interval arithmetic are defined in section 2.2 followed by a discussion of interval valued

functions and natural interval extensions in section 2.3. Next we take a look at various solution methods that use interval analysis. Section 2.4 reviews the basic divide and conquer algorithm for solving nonlinear systems of equations. Finally, section 2.5 presents a simple branch and bound algorithm for solving general nonlinear optimization problems.

Chapter 3 extends the concepts introduced in the previous chapter and presents the most important state of the art methods in use in interval analysis today. We begin with a discussion of Taylor Forms and Taylor Modes in section 3.1. In section 3.2 we review some of the higher order types of inclusion functions such as the Centered (Slope) Form and the Bernstein Form. Finally, section 3.3 discusses some of the most important variants of Interval Newton for solving systems of nonlinear equations.

Part two of this thesis details our contributions to the state of the art. We give proofs of all the new results as well as some new, more elegant, proofs of results that are already known.

In chapter 4 we introduce our first contribution, the *Corner Taylor Form* interval extension. The Corner Taylor Form is a special case of the more general Taylor Form interval extension. The majority of the previous research on inclusion functions was concerned primarily with achieving minimal excess width on small input intervals. Unfortunately, the resulting inclusion functions often had worse excess width than the natural extension on larger input intervals. Many times the excess width was so large as to render the results useless. In contrast, the Corner Taylor Form's excess width is guaranteed to always be smaller than the excess width of the corresponding natural extension, for all input intervals, large or small. The Corner Taylor Form has equal excess width to the natural extension if and only if the natural extension has zero excess width. Proofs of these properties for Corner Taylor Forms with interval valued coefficients are detailed in section 4.8. For the special case of Corner Taylor Forms with real valued coefficients we prove some extended properties in section 4.9. Here we develop a constructive proof and a closed form polynomial expression for the magnitude of the reduction in the excess width as a function of the width of the input interval. These formulas can be used in practice to estimate when the Corner Taylor Form would yield significant benefits—if the benefit is not large enough one could use the less accurate but more efficient natural extension. In addition, the Corner Taylor

Form has many of the desirable properties of the more general Taylor Form interval extension. In section 4.8.2 we prove isotonicity (the interval analytic equivalent monotonicity). Finally, in section 4.9.2 we develop a novel proof showing the excess width of the Corner Taylor Form has at least quadratic order of convergence or better. In particular, we show how to compute the order of convergence in closed form as a function of the expression of the polynomial and the input interval under investigation. These formulas can be used once again to make real-time decisions about which inclusion functions to use. The proofs are facilitated by a new, surprisingly simple and powerful tool called *Posynomial Decomposition*. Posynomial Decomposition (PD) is described in section 4.5. The chapter concludes with several simple examples.

Chapter 5 details the second contribution of this thesis. We present a new method for solving systems of nonlinear equations called the *Remainder Interval Newton* method (RIN for short). In place of the commonly used mean value theorem, RIN employs a first order Taylor expansion with interval remainder terms (a.k.a. first order Taylor Models) to linearize the system of equations, see section 5.1.1. Another important component of the RIN method for underdetermined systems is a new subdivision method designed to maximize the benefits of the linearization process, see section 5.1.3. This subdivision scheme allows efficient pruning of regions where solutions are known not to exist. It also provides the option to enclose the solution set with solution-aligned polyhedral regions thereby producing a significant reduction in the number of regions needed to cover a (non point) set of solutions. The new subdivision method further improves the convergence order of the RIN method by a factor of the square root of the original number of steps. In practice we observe several orders of magnitude reduction in the total number of steps as well as the number of solution regions returned, see section 5.5. In section 5.2 we discuss the RIN algorithm for solving square systems of nonlinear equations, with examples shown in section 5.5.

We conclude the thesis with an application of the two contributions to a problem in computer graphics: rendering of implicit surfaces, see chapter 6. We are particularly interested in robustly rendering “difficult” implicit surfaces, with very high curvature and fine hair like features which would be impossible to render using other methods.

Chapter 2

Review of Interval Analysis

In this chapter we review some of the fundamental definitions and properties of interval analysis that will be used throughout this thesis.

For more in depth discussion of topics related to interval analysis we refer the reader to the books by Moore [Moore 1979], Alefeld and Hertzberger [Alefeld and Herzberger 1983], Hansen [Hansen 1992] and more recently with Walster [Hansen and Walster 2003], Neumaier [Neumaier 1990], and Jaulin et al. [Jaulin et al. 2001].

2.1 A Note About Notation

In this thesis we introduce a new system of notation for interval quantities which we feel is clearer and more convenient than other notation systems currently found in the interval analysis literature. The reason for introducing new notation is our desire for a system that interferes as little as possible with current *de facto* mathematical notation yet is simple, uncluttered and clearly identifies any interval quantities present in formulas.

We looked for notation that can be easily employed in handwritten text, thus ruling out the use of typographical enhancements—such as bold letters representing intervals—that have been previously proposed.

The most commonly used interval notation dates back to Moore and the early days of interval analysis. It uses capitalized letters to represent interval quantities, i.e. x is a real, while X is an interval. Unfortunately, this notation interferes with usual matrix and set notation. It can

cause a lot of confusion in formulas where both intervals and matrices (or sets) are present simultaneously.

Another system identifies interval quantities by enclosing them in square brackets, i.e. x is a real and $[x]$ is an interval. While this notation avoids confusion with matrix and set notation and is handwriting friendly, we feel it adds unnecessary bulk to formulas and can be confused for "just" parentheses.

After many experiments we came to the solution of using simultaneous overbars and underbars to represent interval quantities. This notation is consistent with the current use of overbars and underbars to respectively represent the upper and lower bounds of intervals. Since an interval is composed of both its upper and lower bounds, it was only natural to visually merge the two into the symbolic representation of the interval.

2.1.1 Interval Notation

In this thesis, real numbers are denoted by lowercase letters, e.g. $x, a, u \in \mathbb{R}$. The set of all closed real intervals is denoted by $\mathbb{IR}$. Members of $\mathbb{IR}$ are denoted by lowercase letters with over and underbars, e.g. $\overline{\underline{x}} = [x_0, x_1]$, $\overline{\underline{a}}$, $\overline{\underline{u}} \in \mathbb{IR}$. For emphasis, vectors are denoted by bold letters, e.g. $\mathbf{x}$, $\mathbf{a}, \mathbf{u} \in \mathbb{R}^n$ are vectors of reals, and $\overline{\underline{\mathbf{x}}}$, $\overline{\underline{\mathbf{a}}}$, $\overline{\underline{\mathbf{u}}} \in \mathbb{IR}^n$ are vectors of intervals. The components of a vector are marked with subscripts, e.g. $\mathbf{x}_i$, $\overline{\underline{\mathbf{y}}}_j$. Matrices are denoted with capital letters as usual, while interval matrices have over and underbars: $\mathrm{A} \in \mathbb{R}^{m \times n}$, $\overline{\underline{\mathrm{B}}} \in \mathbb{IR}^{m \times n}$.

Real valued functions are denoted with small caps, e.g. $f(x)$, $g(\overline{\underline{u}})$, $h : \mathbb{R}^n \to \mathbb{R}$, while interval valued functions have over and underbars, e.g. $\overline{\underline{f}}(x)$ or $\overline{\underline{g}}(\overline{\underline{u}})$, $\overline{\underline{h}} : \mathbb{IR}^n \to \mathbb{IR}$. Inclusion functions are denoted by calligraphic caps operating on the original function, e.g. $\mathcal{I}(f)(\overline{\underline{x}})$. In general, $\mathcal{I}(f)$ is a generic inclusion function of f. Specific types of inclusion functions are denoted with special characters, such as $\mathcal{N}(f)$ for the natural extension of f.

In summary:

Real scalar:	x,
Scalar interval (older notations: $X, [x], \mathbf{x}$):	$\overline{\underline{x}}$,
Real vector:	$\mathbf{x}$,
Vector interval:	$\overline{\underline{\mathbf{x}}}$,
i-th component of real vector:	$\mathbf{x}_i$,
i-th component of interval vector:	$\overline{\underline{\mathbf{x}}}_i$,
Real matrix:	M,
Interval matrix:	$\overline{\underline{M}}$,
(i,j)-th component of real matrix:	M_{ij},
(i,j)-th component of interval matrix:	$\overline{\underline{M}}_{ij}$,
Real valued function:	$f(x)$,
Interval valued function:	$\overline{\underline{f}}(x)$,
Generic inclusion function of f:	$\mathcal{I}(f)$,
Generic inclusion function of f (evaluated on $\overline{\underline{x}}$):	$\mathcal{I}(f)(\overline{\underline{x}})$,
Natural extension of f:	$\mathcal{N}(f)$,
Natural extension of f (evaluated on $\overline{\underline{x}}$):	$\mathcal{N}(f)(\overline{\underline{x}})$.

More types of inclusion functions and interval extensions are discussed in the following chapters. To keep the list above short we will introduce their notations at the time of their first appearance in the text.

2.1.2 Other Notation

In the following chapters we need to clearly distinguish not only between different functions but also between different expressions of the same function. For the sake of clarity we have chosen a somewhat verbose notation which uses common English names for the expressions in question.

A generic expression of a function f will be denoted by:

Generic expression: $Expression(f)$.

Note that the expression is itself a function. Evaluation of the expression at a point *x* is written:

Evaluation of a generic expression: $Expression(f)(x)$.

When a function has a natural (or canonical) expression we denote it by:

Natural (canonic) expression: $Natural(f)(x)$.

As an example, we list the names of some common expressions of polynomials (in evaluation form) below:

Horner factoring:	$Horner(f)(x)$,
Taylor expansion:	$Taylor(f,c)(x)$,
MacLaurin expansion:	$MacLaurin(f)(x)$,
Horner-Taylor expansion:	$Horner\text{-}Taylor(f,c)(x)$,
Bernstein expansion:	$Bernstein(f)(x)$,
Chebyshev expansion:	$Chebyshev(f)(x)$.

To facilitate writing expressions of multivariate functions we use the following notation:

$$
\begin{array}{lrcl}
\text{Multi-index:} & \mathbf{i} & \in & \mathbb{N}^n, \\
\text{Vector factorial:} & \mathbf{i}! & = & \prod_{k=1}^{n} i_k!, \\
\text{Vector binomial coefficients:} & \begin{pmatrix} \mathbf{n} \\ \mathbf{i} \end{pmatrix} & = & \prod_{k=0}^{n} \begin{pmatrix} n_k \\ i_k \end{pmatrix}, \\
\text{Vector power:} & \mathbf{x}^{\mathbf{i}} & = & \prod_{k=1}^{n} x_k^{i_k}, \\
\text{Vector partial derivative:} & p^{(\mathbf{i})}(\mathbf{x}) & = & \dfrac{\partial^{i_1+\ldots+i_n}}{\partial^{i_1} x_1 \ldots \partial^{i_n} x_n} p(\mathbf{x}).
\end{array}
$$

2.2 Intervals and Interval Arithmetic

A real valued interval represents the closed set of real numbers contained between a lower bound $\underline{x}$ and an upper bound $\overline{x}$:

$$\text{Interval: } \overline{\underline{x}} = [\underline{x}, \overline{x}] = \{x \mid \underline{x} \leq x \leq \overline{x}\}.$$

Intervals with $\underline{x} = \overline{x}$ are called *thin*, *point* or *degenerate intervals*, while intervals with $\underline{x} < \overline{x}$ are called *thick* or *proper intervals*.

If $0 \leq \underline{x}$ the interval is called a *positive interval*, and we write $\overline{\underline{x}} > 0$. Conversely, if $\overline{x} \leq 0$ we call the interval a *negative interval*, and write $\overline{\underline{x}} < 0$. Positive or negative intervals are the two types of *sign coherent intervals*. If $\underline{x} = 0$ or $\overline{x} = 0$ we call the interval a *zero-bound interval*. A zero-bound positive interval is called a *zero-positive interval*. Similarly, a zero-bound negative interval is called a *zero-negative interval*.

$$
\begin{array}{ll}
\text{Positive interval:} & \overline{\underline{x}} > 0 \text{ iff } \underline{x} > 0, \\
\text{Negative interval:} & \overline{\underline{x}} < 0 \text{ iff } \overline{x} < 0, \\
\text{Zero-positive interval:} & \overline{\underline{x}} \geq 0 \text{ iff } \underline{x} = 0, \\
\text{Zero-negative interval:} & \overline{\underline{x}} \leq 0 \text{ iff } \overline{x} = 0.
\end{array}
$$

A metric (distance function) on $\mathbb{IR}$ can be defined as follows:

$$\text{Metric (distance): } \mathrm{q}\left(\underline{\overline{x}}, \underline{\overline{y}}\right) = \sup\left(|\ \underline{x} - \underline{y}\ |, |\ \overline{x} - \overline{y}\ |\right).$$

It is straightforward to verify that the above operator satisfies all the requirements of a metric. Therefore, one can define convergent series of intervals in the usual fashion:

$$\text{Convergent sequence: } \lim_{i\to\infty} \langle\, \underline{\overline{x}}_i \rangle = \underline{\overline{x}} \text{ iff } \lim_{i\to\infty} \mathrm{q}(\,\underline{\overline{x}}_i, x) = 0.$$

Some common unary operators on a real interval $\underline{\overline{x}}$ are defined as follows:

$$\begin{aligned}
\text{Midpoint:} \qquad \mathrm{m}(\,\underline{\overline{x}}\,) &= \frac{\underline{x} + \overline{x}}{2}, \\
\text{Corner:} \qquad \mathrm{c}(\,\underline{\overline{x}}\,) &= \begin{cases} 0 & , \text{ if } 0 \in \underline{\overline{x}} \\ \underline{x} & , \text{ if } \underline{x} > 0 \\ \overline{x} & , \text{ if } \overline{x} < 0 \end{cases} \\
\text{Width:} \qquad \mathrm{w}(\,\underline{\overline{x}}\,) &= \overline{x} - \underline{x}, \\
\text{Radius:} \qquad \mathrm{rad}(\,\underline{\overline{x}}\,) &= \frac{\overline{x} - \underline{x}}{2}, \\
\text{Absolute Value:} \qquad |\,\underline{\overline{x}}\,| &= \{|y| \mid \underline{x} \le y \le \overline{x}\}, \\
\text{Mignitude:} \qquad \mathrm{mig}(\,\underline{\overline{x}}\,) &= \min|\,\underline{\overline{x}}\,|, \\
\text{Magnitude:} \qquad \mathrm{mag}(\,\underline{\overline{x}}\,) &= \max|\,\underline{\overline{x}}\,|, \\
\text{Sign:} \qquad \mathrm{sgn}(\,\underline{\overline{x}}\,) &= \begin{cases} -1 & , \text{ if } \overline{x} \le 0 \\ 0 & , \text{ if } \underline{x} < 0 < \overline{x} \\ +1 & , \text{ if } \underline{x} \ge 0 \end{cases}, \\
\text{Interior:} \qquad \mathrm{int}(\,\underline{\overline{x}}\,) &= (\,\underline{x}, \overline{x}\,).
\end{aligned}$$

If S is a set of real numbers we denote its one-dimensional convex hull by $[[\,S\,]]$. Obviously:

$$\text{Interval Hull:} \quad [[\,S\,]] \;=\; [\inf(S), \sup(S)],$$

is a closed interval.

The corresponding vector operators are defined component wise. For example, the vector corner operator is:

$$\text{Vector Corner:} \quad (\mathrm{c}(\underline{\bar{\mathbf{x}}}))_k \;=\; \mathrm{c}(\underline{\bar{x}}_k).$$

We do not define the sign of an interval vector.

The usual unary and binary arithmetic operations are defined as follows:

$$\begin{aligned}
&\text{Negation:} && -\underline{\bar{x}} &&=\; [-\bar{x}, -\underline{x}],\\
&\text{Addition:} && \underline{\bar{x}} + \underline{\bar{y}} &&=\; \left[\underline{x} + \underline{y}, \bar{x} + \bar{y}\right],\\
&\text{Subtraction:} && \underline{\bar{x}} - \underline{\bar{y}} &&=\; \left[\underline{x} - \bar{y}, \bar{x} - \underline{y}\right],\\
&\text{Multiplication:} && \underline{\bar{x}}\;\underline{\bar{y}} &&=\; \left[\left[\,\underline{x}\,\bar{y}, \bar{x}\,\underline{y}, \underline{x}\,\bar{y}, \bar{x}\,\bar{y}\,\right]\right],\\
&\text{Division:} && \frac{\underline{\bar{x}}}{\underline{\bar{y}}} &&=\; \left[\left[\frac{\underline{x}}{\bar{y}}, \frac{\bar{x}}{\underline{y}}, \frac{\underline{x}}{\bar{y}}, \frac{\bar{x}}{\bar{y}}\right]\right].
\end{aligned}$$

Note that for the above definition of division to work $\underline{\bar{y}}$ must not contain zero. It is possible to extend the definition of division to include cases where zero is a member of $\underline{\bar{y}}$; examples can be found in the literature.

We also define:

$$\text{Positive Integral Power:} \quad \underline{\bar{x}}^{\,n} \;=\; \begin{cases} [\underline{x}^{\,n}, \bar{x}^{\,n}], \text{ if } n \text{ is odd or } \underline{\bar{x}} \text{ is positive} \\ |\underline{\bar{x}}|^{n}, \text{ if } n \text{ is even} \end{cases},$$

$$\text{Negative Integral Power:} \quad \underline{\bar{x}}^{\,-n} \;=\; \frac{1}{\underline{\bar{x}}^{\,n}}.$$

We departed slightly from the usual practice of treating positive integral powers as repeated

multiplications in favor of the above definition. The above integral power operator avoids the excess width introduced by repeated multiplication, as illustrated in the following example:

$$[-2,3]^2 \;=\; [0,9] \;=\; \{y = x^2 \mid x \in [-2,3]\},$$

$$[-2,3]\,[-2,3] \;=\; [-6,9] \;=\; \{y = x_1 x_2 \mid x_1, x_2 \in [-2,3]\}$$

Note that the power operator (top) returns the exact range of values while the same expression evaluated through repeated multiplication (bottom) can suffer from significant overestimation. The overestimation is due to the fact that interval multiplication as defined above assumes there is no correlation between the two factors being multiplied together.

Interval addition and multiplication exhibit the following algebraic properties:[1]

Commutativity:	$\underline{x} + \underline{y}$	$=$	$\underline{y} + \underline{x}$,	$\underline{x}\,\underline{y}$	$=$	$\underline{y}\,\underline{x}$,
Associativity:	$(\underline{x} + \underline{y}) + \underline{z}$	$=$	$\underline{x} + (\underline{y} + \underline{z})$,	$(\underline{x}\,\underline{y})\,\underline{z}$	$=$	$\underline{x}\,(\underline{y}\,\underline{z})$,
Neutral Element:	$0 + \underline{x}$	$=$	$\underline{x}$,	$1 \cdot \underline{x}$	$=$	$\underline{x}$.

Unfortunately, multiplication of intervals is not distributive over addition as it is with real numbers. A subdistributive law holds:

$$\text{Subdistributivity:} \quad \underline{x}\,\left(\underline{y} + \underline{z}\right) \;\subseteq\; \underline{x}\,\underline{y} + \underline{x}\,\underline{z}\,.$$

[1] These properties do not hold in the presence of rounding error.

It is useful to identify the special cases where distributivity does hold:

$$\text{Distributivity holds:} \begin{cases} \underline{x}\,(\underline{y} \pm \underline{z}) = \underline{x}\,\underline{y} \pm \underline{x}\,\underline{z} \text{ if } \underline{x} = \overline{x} \text{ (thin factor)}, \\ \underline{x}\,(\underline{y} + \underline{z}) = \underline{x}\,\underline{y} + \underline{x}\,\underline{z} \text{ if } \underline{y} \geq 0 \text{ and } \underline{z} \geq 0 \text{ (non-negative terms)}, \\ \underline{x}\,(\underline{y} + \underline{z}) = \underline{x}\,\underline{y} + \underline{x}\,\underline{z} \text{ if } \underline{y} \leq 0 \text{ and } \underline{z} \leq 0 \text{ (non-positive terms)}, \\ \underline{x}\,(\underline{y} - \underline{z}) = \underline{x}\,\underline{y} - \underline{x}\,\underline{z} \text{ if } \underline{y} \geq 0 \text{ and } \underline{z} \leq 0 \text{ (non-negative terms variation)}, \\ \underline{x}\,(\underline{y} - \underline{z}) = \underline{x}\,\underline{y} - \underline{x}\,\underline{z} \text{ if } \underline{y} \leq 0 \text{ and } \underline{z} \geq 0 \text{ (non-positive terms variation)}, \\ \underline{x}\,(\underline{y} \pm \underline{z}) = \underline{x}\,\underline{y} \pm \underline{x}\,\underline{z} \text{ if } \underline{x} \geq 0,\ \underline{y} = 0 \text{ and } \overline{z} = 0 \\ \qquad\qquad \text{(positive factor, zero-straddling terms)}, \\ \underline{x}\,(\underline{y} \pm \underline{z}) = \underline{x}\,\underline{y} \pm \underline{x}\,\underline{z} \text{ if } \underline{x} \leq 0,\ \underline{y} = 0 \text{ and } \overline{z} = 0 \\ \qquad\qquad \text{(negative factor, zero-straddling terms)}. \end{cases}$$

The proofs are straightforward and can be found in most texts on interval analysis, such as [Neumaier 1990].

2.3 Inclusion Functions and Interval Extensions

In this section we review some of the concepts associated with interval valued functions and their use for computing bounds for the range of real valued functions over interval domains.

2.3.1 Interval Valued Functions and the Range Inclusion Function

An interval valued function is a mapping from a set S to the intervals $\mathbb{IR}$:

$$\text{Interval valued function: } \overline{f} : S \to \mathbb{IR}.$$

An example of an interval valued function on the reals is the *floating point bound function*:

$$\text{FP-bounds: } \underline{\overline{FP}} : \mathbb{R} \to \mathbb{IR},\ \underline{\overline{FP}}\ (x) = [x_*, x^*]\,,$$

where x_* is the largest floating point number smaller than or equal to x, and x^* is the smallest floating point number greater than or equal to x.

Another example of an interval valued function, this time on intervals, is the *range inclusion function* associated with a continuous real valued function $f : \mathbb{R} \to \mathbb{R}$ and denoted by $\mathcal{R}(f) : \mathbb{IR} \to \mathbb{IR}$:

$$\text{Range (continuous case): } \mathcal{R}(f)(\underline{\overline{x}}) = \{y \mid (y = f(x)) \wedge (x \in \underline{\overline{x}})\}\,.$$

The range can be defined for discontinuous functions as well, using the interval hull of the range set, as follows:

$$\text{Range (discontinuous case): } \mathcal{R}(f)(\underline{\overline{x}}) = \left[\!\left[\ \{y \mid (y = f(x)) \wedge (x \in \underline{\overline{x}})\}\ \right]\!\right].$$

An interval valued function is *continuous* if for any convergent sequence $\langle \underline{\overline{x}}_i \rangle$ (or $\langle x_i \rangle$ if the domain is $\mathbb{R}$) the sequence of intervals $\langle \underline{\overline{f}}\ (\underline{\overline{x}}_i) \rangle$ (or $\langle \underline{\overline{f}}\ (x_i) \rangle$) converges as well.

2.3.2 Inclusion of the Range of Real Valued Functions

Using the range inclusion function defined in the previous section it is straightforward to design algorithms for solving many important problems, such global nonlinear optimization. Unfortunately, the range inclusion function is often not computable.

It turns out that computing a superset of the range inclusion function is usually sufficient, provided it has certain characteristics. We call these types of interval valued functions *inclusion functions*. As the name suggests, the value of an inclusion function over some interval $\underline{\overline{\mathbf{x}}}$ in the domain is an interval in the codomain that includes the range of the function over the interval $\underline{\overline{\mathbf{x}}}$:

$$\text{Inclusion function: } \mathcal{I}(f)(\underline{\overline{\mathbf{x}}}) \supseteq \mathcal{R}(f)(\underline{\overline{\mathbf{x}}}) \text{ for } \forall\, \underline{\overline{\mathbf{x}}} \subseteq D,$$

or, equivalently:

$$\text{Inclusion Property: } f(\mathbf{x}) \in \mathcal{I}(f)(\overline{\underline{\mathbf{x}}}) \text{ for } \forall\, \overline{\underline{\mathbf{x}}} \subseteq D \text{ and } \forall \mathbf{x} \in \overline{\underline{\mathbf{x}}},$$

where D is the domain of f. An important class of inclusion functions are the *inclusion isotone* inclusion functions which satisfy the following property:

$$\text{Inclusion Isotonicity: } \mathcal{I}(f)(\overline{\underline{\mathbf{x}}}) \subseteq \mathcal{I}(f)\left(\overline{\underline{\mathbf{y}}}\right) \text{ for } \forall\, \overline{\underline{\mathbf{x}}} \subseteq \overline{\underline{\mathbf{y}}} \subseteq D.$$

With the above definitions many different classes of inclusion functions can be defined for any given function f, the range being the "tightest" and most useful, while the inclusion function that always returns $[-\infty, \infty]$ is the "widest" but also not useful at all. We define a metric, called *excess width*, for measuring how close a given inclusion function is to the range:

$$\text{Excess width: } \Delta\mathrm{W}(\mathcal{I}(f))(\overline{\underline{\mathbf{x}}}) = \mathrm{w}(\mathcal{I}(f)(\overline{\underline{\mathbf{x}}})) - \mathrm{w}(\mathcal{R}(f)(\overline{\underline{\mathbf{x}}})).$$

Obviously, the excess width is a non-negative real valued function with interval arguments; the smaller the excess width is over some interval $\overline{\underline{\mathbf{x}}}$ the better the inclusion function over $\overline{\underline{\mathbf{x}}}$ is. Another measure of overestimation is the *excess width ratio*, which gives information about the excess width relative to the size of the arguments:

$$\text{Excess width ratio: } \Delta\mathrm{WR}(\mathcal{I}(f))(\overline{\underline{\mathbf{x}}}) = \frac{\Delta\mathrm{W}(\mathcal{I}(f))(\overline{\underline{\mathbf{x}}})}{\mathrm{w}(\overline{\underline{\mathbf{x}}})}.$$

Yet another measure for overestimation of the range is the *excess width range ratio*, which gives information about the excess width relative to the size of the range itself:

$$\text{Excess width range ratio: } \Delta\mathrm{WRR}(\mathcal{I}(f))(\overline{\underline{\mathbf{x}}}) = \frac{\Delta\mathrm{W}(\mathcal{I}(f))(\overline{\underline{\mathbf{x}}})}{\mathrm{w}(\mathcal{R}(f)(\overline{\underline{\mathbf{x}}}))}.$$

It is often useful to know the behavior of the overestimation of an inclusion function when evaluated on a sequence of intervals converging to a point. Of course, we prefer inclusion func-

tions whose excess width goes to zero in this case and in fact some authors require this property in the definition of the inclusion function. If this is the case, we define the *inclusion order* of an inclusion function to be the order of convergence of the excess width when evaluated on a sequence of intervals convergent to a point:

$$\text{Inclusion order: } \mathcal{O}\left(\mathcal{I}(f)\right) = \mathcal{O}\left(\Delta \mathrm{W}(\mathcal{I}(f))(\underline{\overline{\mathbf{x}}}_i)\right)\Big|_{\underline{\overline{\mathbf{x}}}_i \to \mathbf{x}}.$$

2.3.3 Interval Extensions

We come to the question of obtaining an expression of an inclusion function from some given expression of a function f. If the expression of f is formed from compositions of the four basic arithmetic operations $\{+, -, \times, \div\}$ and integer powers, this turns out to be easy: replace all occurrences of real variables x_i with interval variables $\underline{\overline{x}}_i$ and all the real operations with the corresponding interval operations and what you get is the expression of an inclusion function. This is the statement of the *fundamental theorem of interval analysis*. The process of obtaining the expression of an inclusion function from the expression of the original function is called *interval extension*, see [Moore 1979]. Given some expression we write the interval extension simply as an evaluation with interval coefficients:

$$\text{Interval extension: } \mathit{Expression}(f)(\underline{\overline{x}}) \text{ is the interval extension of } \mathit{Expression}(f)(x).$$

The above definitions cover interval extensions of rational functions. However, interval extensions can be generalized to include other types of primitives. First, we define some composition rules:

$$\text{Function composition 1: } \mathit{Expression}(f)(\mathit{Expression}(g)(\underline{\overline{x}})) \text{ is an inclusion function of } f \circ g.$$

If r is a rational function and f is a function for which an inclusion function $\mathcal{I}(f)$ is known we

can obtain an inclusion function of $r \circ f$ and $f \circ r$ as follows:

Function composition 2: $Expression(r)(\mathcal{I}(g)(\underline{\overline{x}}))$ is an inclusion function of $r \circ g$,

Function composition 3: $\mathcal{I}(g)(Expression(r)(\underline{\overline{x}}))$ is an inclusion function of $g \circ r$.

We can summarize the composition rules above into the following rule:

Function composition: $\mathcal{I}(f)(\mathcal{I}(g)(\underline{\overline{x}}))$ is an inclusion function of $f \circ g$.

The proofs are straightforward. Note that the composition rules also allow the computation of inclusion functions of algorithmically (recursively) defined functions.

Inclusion functions for many elementary functions can be easily defined and are detailed in the literature. These include square, cubic and higher roots, trigonometric and inverse trigonometric functions, logarithms and exponentials, etc. Thus, using the composition rules, inclusion functions for a large variety of complicated functions can be easily obtained.

Most functions have a standard expression—an expression that is most often used. The inclusion functions generated from these expressions are called *natural inclusion functions*, or *natural extensions*. For example, given a multivariate polynomial p, the natural extension [2] is defined as:

$$\text{Natural extension for polynomials: } \mathcal{N}(p)(\underline{\overline{\mathbf{x}}}) = MacLaurin(p)(\underline{\overline{\mathbf{x}}}).$$

The natural extension of $\cos((x-1)(x-3))$ is:

$$\mathcal{N}(\cos((x-1)(x-3)))(\underline{\overline{x}}) = \cos\left(\underline{\overline{x}}^{\,2} - 4\,\underline{\overline{x}} + 3\right).$$

In general, inclusion functions obtained through interval extension from different expressions of the same function f are *not the same*, even when evaluated with infinite precision. This is unlike the real case where different expressions evaluate to the same value of $f(x)$. One of the

[2] other authors choose to define the natural extension of polynomials as $Horner(p)(\underline{\overline{\mathbf{x}}})$. This definition yields tighter inclusion functions when powers are treated as repeated multiplication, see chapter 3.

main interests of interval analysis is in finding expressions of functions that generate "better" inclusion functions, see chapters 3 and 4.

2.4 Inclusion of the Solution Set of Nonlinear Systems of Equations

In this section we review some basic algorithms for robustly computing roots of nonlinear equations of the form:

$$\mathbf{f}(\mathbf{x}) = 0 \tag{2.1}$$

using interval analysis. We purposely postpone any discussion of Interval Newton until chapter 3, so only basic divide and conquer type algorithms are presented here.

First we formalize the problem and the requirements on the solutions. The bold face of $\mathbf{f}$ in $\mathbf{f}(\mathbf{x}) = 0$ means $\mathbf{f}$ is a vector valued function:

$$\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m.$$

Let S be the set of all the roots of equation (2.1) inside some domain $\underline{\overline{\mathbf{d}}} \subset \mathbb{R}^n$:

$$S = \{\mathbf{x} \in \underline{\overline{\mathbf{d}}} \mid \mathbf{f}(\mathbf{x}) = 0\}.$$

We are interested in finding a finite interval covering of S. A finite interval covering is a finite set of interval vectors $\overline{\underline{S}}_\varepsilon = \{\underline{\overline{\mathbf{s}}}_i\}$ with disjoint interiors such that:

1. $S \subseteq \overline{\underline{S}}_\varepsilon$, i.e. given $\forall \mathbf{x} \in S$ then $\exists i$, $\underline{\overline{\mathbf{s}}}_i \in \overline{\underline{S}}_\varepsilon$ and $\mathbf{x} \in \underline{\overline{\mathbf{s}}}_i$,
2. $\text{int}(\underline{\overline{\mathbf{s}}}_i) \cap \text{int}(\underline{\overline{\mathbf{s}}}_j) = \emptyset$ for $\forall i, j, (i \neq j)$,
3. $\forall i, \text{w}(\underline{\overline{\mathbf{s}}}_i) \leq \varepsilon$, with $\varepsilon \in \mathbb{R}$.

The intervals $\underline{\overline{\mathbf{s}}}_i$ are called solution intervals. The first rule above ensures that we do not miss any solutions. The second rule ensures that intervals do not cover the same area twice. Finally, the last rule is a weak measure of the accuracy of the interval covering with respect to S.

Note however, that there might be *superfluous solution intervals* in $\overline{\underline{S}}_\varepsilon$ that contain no solutions. We mark superfluous solution intervals by $\overline{\underline{\mathbf{s}}}_i^\circ$ and their subset by $\overline{\underline{S}}_\varepsilon^\circ \subseteq \overline{\underline{S}}_\varepsilon$. Solution intervals that contain at least one solution are called *proper solution intervals* and are marked by $\overline{\underline{\mathbf{s}}}_i^\bullet$. The *proper solution covering* is denoted by $\overline{\underline{S}}_\varepsilon^\bullet \subseteq \overline{\underline{S}}_\varepsilon$.

We could impose an additional rule that prevents such superfluous solution intervals from ever appearing in the finite interval covering:

4. $\forall i, \overline{\underline{\mathbf{s}}}_i \cap S \neq \emptyset$.

However, interval coverings that satisfy these stronger requirements are not as easily computable. Therefore, we only require the first three rules here. In practice, the number of superfluous solution intervals (at some fixed value of ε) is directly related to the accuracy of the inclusion function used to generate them.

The *quality of a finite interval covering* is defined as the ratio between the total area of the proper solution intervals and the total area of all the intervals in the covering:

$$\text{Quality of a finite interval covering: } Q\left(\overline{\underline{S}}_\varepsilon\right) = \frac{\mathcal{A}\left(\overline{\underline{S}}_\varepsilon^\bullet\right)}{\mathcal{A}\left(\overline{\underline{S}}_\varepsilon\right)}.$$

The quality is a real number between 0 and 1, with 0 being the worst and 1 being the best.

From now on we will drop the explicit use of the term “finite” and simply write “interval covering” instead.

2.4.1 A Basic Divide and Conquer Algorithm

The simplest algorithm that can compute finite interval coverings like the one specified in the previous section is of the divide and conquer type and is shown in figure 2.1. The algorithm works by adaptively subdividing the domain and eliminating intervals that are guaranteed not to contain any solutions.

The subdivision step can be implemented in several different ways. The intervals can be bisected into equal halves along one of its dimensions. This dimension can be chosen to minimize function variation, or simply be the longest edge. Bisection can also be biased to produce unequal

```
SimpleSolve (
  in f:    function whose solutions we are seeking
  in d:    domain interval
  in ε:    maximum size of a solution interval
  out S_ε: [empty] interval covering of the solutions of f in d
)
{
  create stack: [empty] stack of subintervals to be examined;
  put d on stack;
  while ( stack is not empty ) do
  {
    pop x from stack;
    // estimate the range of f on x
    compute y = J(f)(x);
    if ( 0 ∈ y )
    // there could be solutions in x
    {
      if (w(x) < ε)
        append x to S_ε;
      else
      {
        Subdivide ( in x, out x_1, out x_2 );
        put x_1 on stack;
        put x_2 on stack;
      }
    }
  }
  return; // search is exhausted
}
```

Figure 2.1: A simple divide and conquer algorithm for solving nonlinear systems of equations using interval analysis. The values between square brackets listed next to variable declarations represent initial values. Using a FIFO queue instead of the LIFO stack usually increases storage requirements.

intervals. Multisection is also an option; for example, one could bisect each dimension resulting in 2^n new subintervals at each step.

Any inclusion function $\mathcal{J}(\mathbf{f})$ can be used to bound the range. If the inclusion function used has nonzero excess width the algorithm may not eliminate all such regions and will return an interval covering that contains a certain amount of superfluous solution intervals. The larger the

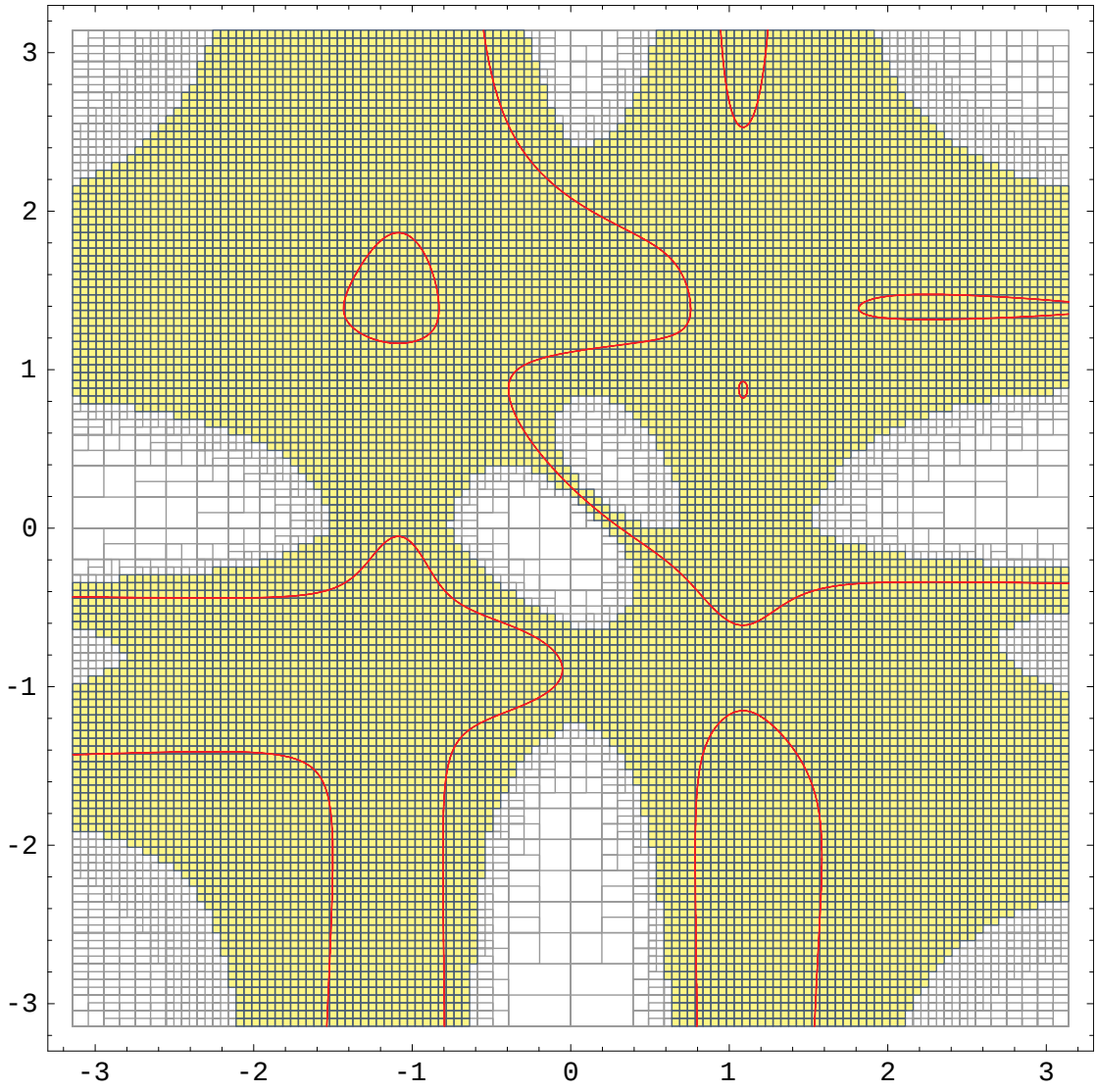


Figure 2.2: Plot of an interval covering produced by the divide and conquer algorithm in figure 2.1, using the natural inclusion function and $\varepsilon = 2^{-4}$. The interval covering contains 12,407 solution intervals and has a quality factor of only 0.0571. The algorithm performed a total of 29,105 iterations which took 84.281 seconds (Mathematica 5, P4-2.2GHz). The time/quality cost is 1,474.858 seconds. Compare this with figure 2.3.

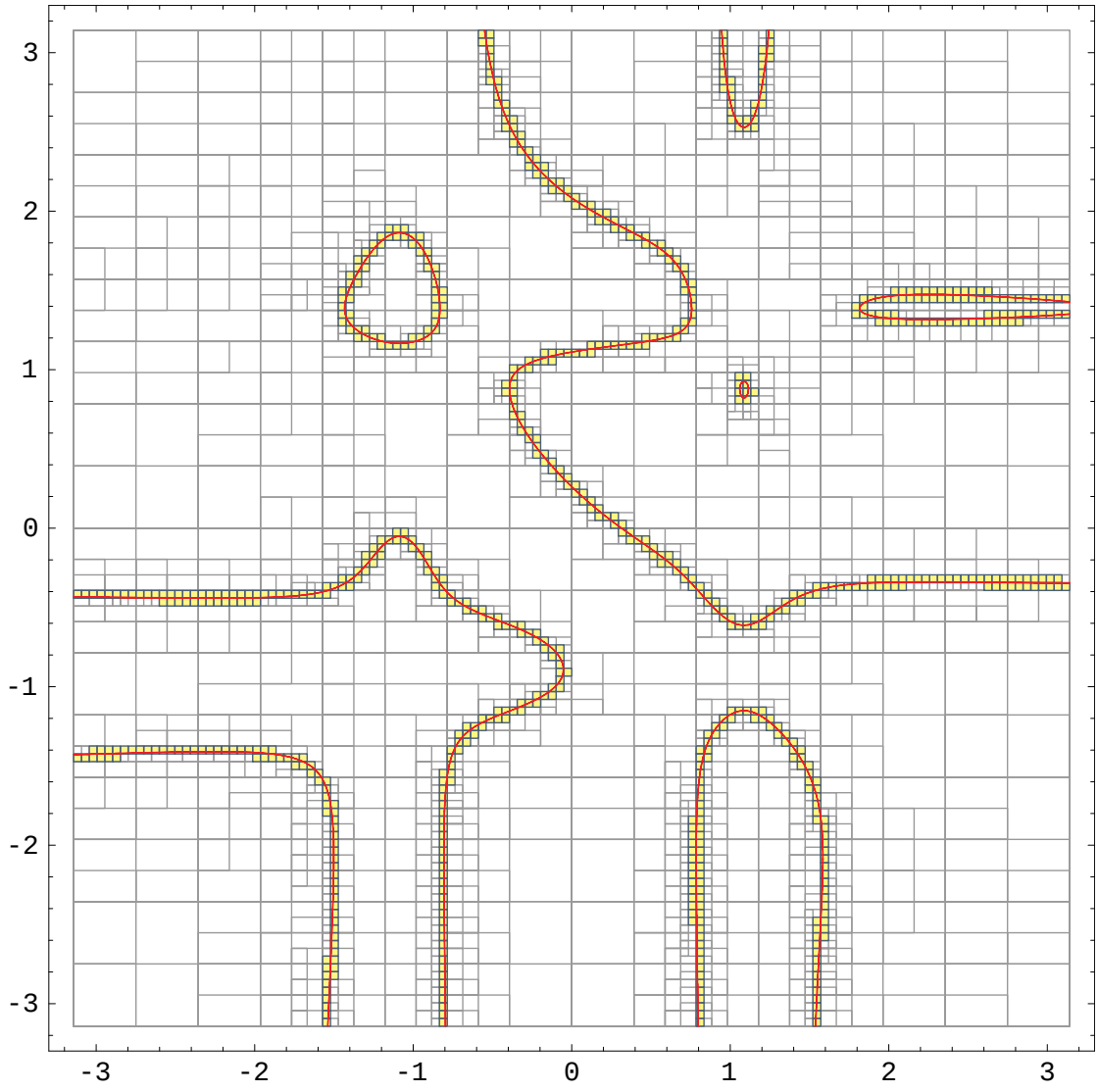


Figure 2.3: Plot of an interval covering produced by the divide and conquer algorithm in figure 2.1, using a Midpoint Taylor Form inclusion function and $\varepsilon = 2^{-4}$. The interval covering contains 788 solution intervals and has a quality factor of 0.8997. The algorithm performed a total of 4,951 iterations which took 63.016 seconds (Mathematica 5, P4-2.2GHz). The time/quality cost is 70.038 seconds. Compare this with figure 2.2.

excess width of the inclusion function is, the worse the quality of the interval covering will be and more superfluous solution intervals will be produced.

In addition, the total running time of the algorithm depends on the amount of excess width as well since large amounts of extra time can be spent processing and subdividing superfluous intervals.

This can be best seen by comparing figure 2.2 and figure 2.3. The result in figure 2.2 is computed using the natural extension and has a quality factor of only 0.0571, much too low to be of any practical use. The result in figure 2.2 is computed using the Midpoint Taylor Form interval extension, see section 3.2.2. It has a much better quality factor of 0.8997.

Although running times are similar, a meaningful comparison of the two approaches could not be made without considering the quality of the interval covering produced. We propose a new measure called the *Time/Quality cost*:

$$\text{Time/Quality cost: TQCost}\left(\overline{\underline{S}}_{\varepsilon}\right) = \frac{time}{Q\left(\overline{\underline{S}}_{\varepsilon}\right)}.$$

The Time/Quality cost of the interval covering in figure 2.2 is 1,474.858 seconds, while that of the interval covering in figure 2.3 is 70.038 seconds.

Within this framework one quickly discovers that naive use of natural extensions routinely produces interval coverings of very bad quality, so bad that they are virtually useless. Cases like these are often responsible for the poor reputation interval analysis still has to date. They show how important it is to use inclusion functions with the smallest excess width available.

In chapters 4 and 5, we make extensive use of the sensitivity of the divide and conquer (and derived algorithms) to the type of inclusion function used to illustrate and measure the qualitative differences between several different types of inclusion functions.

2.5 Inclusion of the Solution Set of Nonlinear Optimization Problems

One of the main appeals of interval analysis is its ability to solve very general nonlinear optimization problems which would be impossible to solve using most other methods. The basic interval algorithm for optimization is due to Moore and Skelboe and is sketched in figure 2.4.

Like we did in the previous section we first formalize the class of problems we are trying to solve. Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a multivariate vector valued function and S be the set of all the global minima of $\mathbf{f}$ inside the domain $\overline{\underline{\mathbf{d}}} \subset \mathbb{R}^n$:

$$S = \{\mathbf{x} \in \overline{\underline{\mathbf{d}}} \mid \forall \mathbf{y} \in \overline{\underline{\mathbf{d}}}, \mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x})\}.$$

Once again, we are interested in finding a finite interval covering of S as defined in section 2.4. Superfluous and proper solution intervals as well as the quality of an interval covering are defined as before. The same considerations about the relationship between the inclusion function used and the quality of the interval covering and the execution efficiency apply to the Moore-Skelboe algorithm.

2.5.1 The Moore-Skelboe Optimization Algorithm

The pseudocode of the constrained optimization algorithm is shown in figure 2.4. The algorithm maintains a dynamic estimate of the upper bound to the global (constrained or unconstrained) minimum value of the function. The upper bound is used to eliminate regions that cannot contain a global minimum as well as regions that were previously thought to contain a global minimum but eventually become obsolete. This latest process is done by the **PurgeObsoleteMinima** routine.

For unconstrained minimization the routine **SatisfiesConstraints** always returns “true”. For constrained minimization problems, the routine returns “true” if there is at least one point inside $\overline{\underline{\mathbf{x}}}$ where all constrains are satisfied or “false” otherwise. Writing such a routine for all types of constraints is not easy. Fortunately such routines can be designed for many interesting problems.

```
MooreSkelboeOptimize (
  in f:    function whose minima we are seeking
  in d̲:    domain interval
  in ε:    maximum size of a solution interval
  out S̲_ε: [empty] interval covering of the minima of f over d̲
)
{
  create stack: [empty] stack of subintervals to be examined;
  real supMin: [∞] current estimate of the upper bound of min_{d̲}(f);
  put d̲ on stack;
  while ( stack is not empty ) do
  {
    pop x̲ from stack;
    // estimate the range of f on x̲
    compute ȳ = 𝕀(f)(x̲);
    if ( ( supMin ≥ y̲ ) and SatisfiesConstraints ( x̲ ) )
    // there could be a global minimum in x̲
    {
      // update global minimum
      set supMin = min(supMin, ȳ);
      // eliminate solution intervals s̲_i with inf(𝕀(f)(s̲_i)) > supMin
      PurgeObsoleteMinima ( in supMin, in/out S̲_ε );
      if ( w(x̲) < ε )
        append x̲ to S̲_ε;
      else
      {
        Subdivide ( in x̲, out x̲_1, out x̲_2 );
        put x̲_1 on stack;
        put x̲_2 on stack;
      }
    }
  }
  return; // search is exhausted
}
```

Figure 2.4: A simple branch and bound nonlinear optimization algorithm using interval analysis, after Moore and Skelboe. The subroutines used in the algorithm are briefly described in section 2.5.1.

2.6 Inclusion of the Solution Set of Systems of Differential and Integral Equations using Interval Picard Iterations

In this section, we very briefly describe the interval version of Picard's algorithm for computing Taylor form enclosures for the solution of certain differential and/or integral equations. For more detail refer to [Berz and Hofstätter 1998].

Interval Picard iteration is not the only method for solving ODEs using interval analysis. Other methods—such as interval versions of Euler's method—have been studied.

2.6.1 Definitions

Norm: $\| f(t)-g(t) \| = \max_t |f(t)-g(t)|$

Lipschitz condition: A function $f(t)$ satisfies a Lipschitz condition on $[a,b]$ if there is a positive real number $L_f^{[a,b]}$ such that for all $t_1, t_2 \in [a,b]$:

$$|f(t_1)-f(t_2)| \leq L_f^{[a,b]} |t_1 - t_2|.$$

Cauchy remainder: If $f(t)$ can be expanded in an Taylor series then there exist $\xi_t \in [t_0, t]$ such that:

$$\begin{aligned} f(t) &= \sum_{i=0}^{\infty} \frac{1}{i!} \frac{d^i f}{dt^i}(t_0)(t-t_0)^i \\ &= \sum_{i=0}^{n} \frac{1}{i!} \frac{d^i f}{dt^i}(t_0)(t-t_0)^i + \frac{1}{(n+1)!} \frac{d^{n+1} f}{dt^{n+1}}(\xi_t)(t-t_0)^{n+1}. \end{aligned}$$

The expression:

$$CR(f, t_0, n) = \frac{1}{(n+1)!} \frac{d^{n+1} f}{dt^{n+1}}(\xi_t)(t-t_0)^{n+1},$$

is called the n^{th} order Cauchy remainder of f at t_0.

Interval Lagrange remainder: ξ_t is not usually computable in closed form. However, the following is always true:

$$\begin{aligned} ILR(f, t_0, n) &= \frac{1}{(n+1)!} \frac{d^{n+1} f}{dt^{n+1}}([t_0, t])(t-t_0)^{n+1}, \\ f(t) &\in \sum_{i=0}^{n} \frac{1}{i!} \frac{d^i f}{dt^i}(t_0)(t-t_0)^i + ILR(f, t_0, n). \end{aligned}$$

The expression $ILR(f, t_0, n)$ is called the n^{th} order interval Lagrange remainder of f at t_0.

2.6.2 Interval Picard Iteration

Consider the following type of differential equation:

$$y' = F(y).$$

which can be rewritten in integral form as:

$$y = y(t_0) + \int_{t_0}^{t} F(y(x))dx.$$

We further assume that F is a polynomial and therefore Lipschitz everywhere (bounded derivative) and that we can produce an initial polynomial enclosure y_0^* for y over the interval $[t_0, t_0 + 1/L_F]$, i.e.

$$y(t) \in y_0^*(t), \forall t \in [t_0, t_0 + 1/L_F].$$

The Picard iteration step

$$y_{n+1}^*(t) = y(t_0) + \int_{t_0}^{t} F(y_n^*(x))dx,$$

defines successively better polynomial enclosures for y (i.e. it is a contraction around the solution y of the differential equation). The integration step can be performed symbolically since all functions involved are polynomials.

The proof that the above Picard iteration step is a contraction is as follows:

$$\begin{aligned}
\| y^*_{n+1}(t) - y(t) \| &= \max_t |y^*_{n+1}(t) - y(t)| \\
&= \max_t \left| y(t_0) + \int_{t_0}^t F(y^*_n(x))dx - y(t_0) - \int_{t_0}^t F(y(x))dx \right| \\
&= \max_t \left| \int_{t_0}^t [F(y^*_n(x)) - F(y(x))]\, dx \right| \\
&\leq \max_t \int_{t_0}^t |F(y^*_n(x)) - F(y(x))|\, dx \\
&\leq \max_t \int_{t_0}^t L_F\, |y^*_n(x) - y(x)|\, dx \\
&\leq \max_t [(t - t_0) L_F\, |y^*_n(x) - y(x)|] \\
&\leq (t_0 + 1/L_F - t_0) L_F \max_t |y^*_n(x) - y(x)| \\
&= \max_t |y^*_n(x) - y(x)| \\
&= \| y^*_n(x) - y(x) \| .
\end{aligned}$$

If F is not a polynomial, but it can be expanded in a Taylor series then we use the following method. Let F^*_n be the n^th degree Taylor series with interval Lagrange remainder of F, i.e.

$$F(x) \in F^*_n(x) = \sum_{i=0}^{n} \frac{1}{i!} \frac{d^i F}{dx^i}(x_0)(x - x_0)^i + ILR(F, x_0, n).$$

Then the Picard iteration step is as follows:

$$y^*_{n+1}(t) = y(t_0) + \int_{t_0}^t F^*_n(y^*_n(x))dx.$$

2.6.3 An Example

Let us look at an example of Picard iteration that could be used to solve the differential equation:

$$y' = y^2, y(0) = 1$$

Let $y^*_0 = [1,2]$. $F(x) = x^2$ and $F'(x) = 2x$. The Lipschitz constant of F on the interval $[1,2]$ is:

$$L_F = maxF'([1,2]) = max([1,4]) = 4.$$

Therefore, we can expect a Picard iteration to be a contraction for values of t in the interval $[0,1/4]$. The first two such Picard iterations are as follows:

$$\begin{aligned} y_1^*(t) &= 1+\textstyle\int_0^t [1,2]^2 dx \\ &= 1+[1,4]t, \end{aligned}$$

$$\begin{aligned} y_2^*(t) &= 1+\textstyle\int_0^t (1+[1,2]x)^2 dx \\ &= 1+x+[1,4]x^2+[1/3,16/3]x^3. \end{aligned}$$

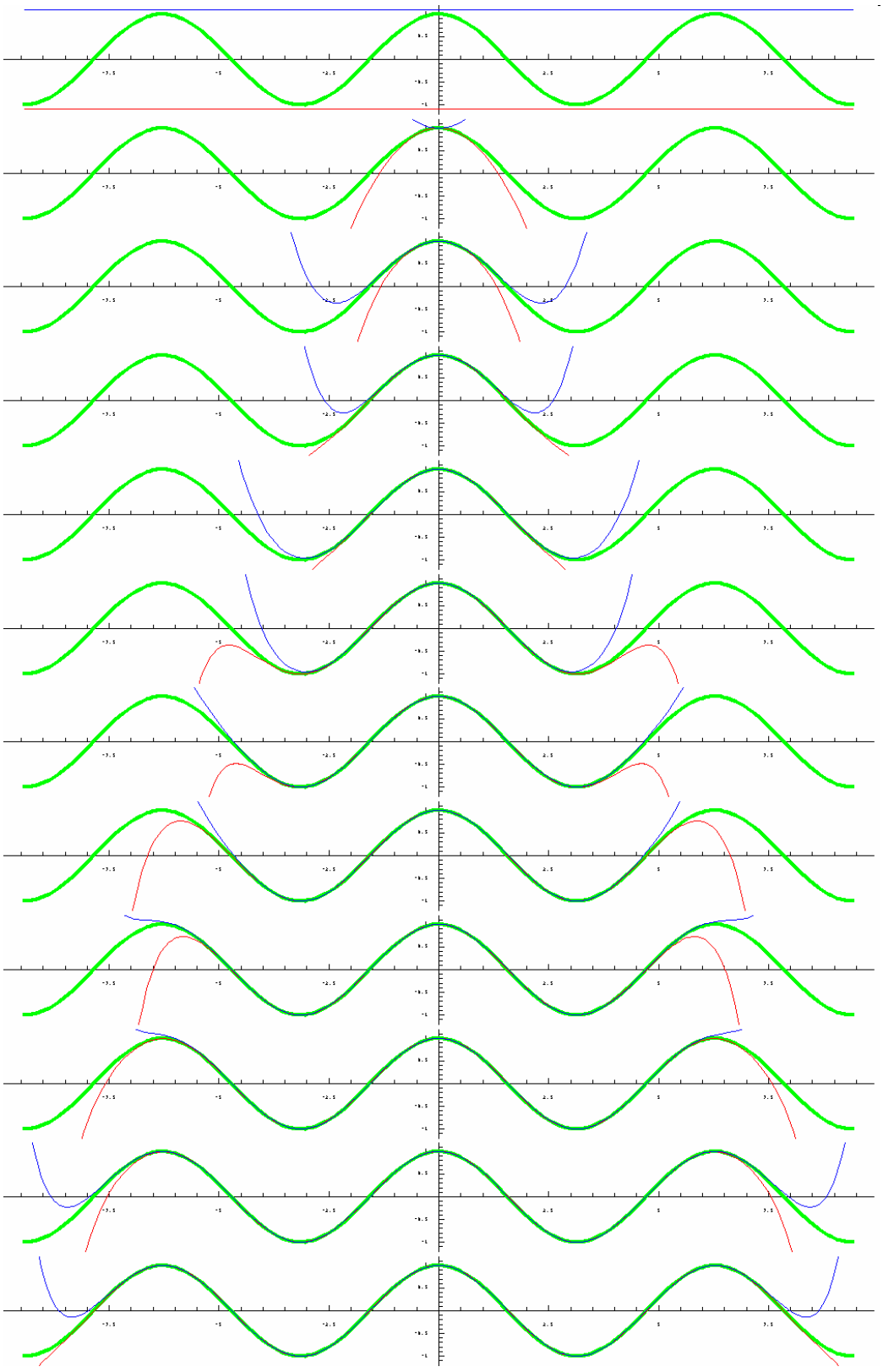


Figure 2.5: An example of contracting Taylor Models generated using interval Picard iteration. The degree of the Taylor Models increases from top to bottom.

Chapter 3

Related Previous Work

In this chapter we review some important state of the art interval algorithms. We begin our discussion with a short review of the concepts of Taylor Forms and Taylor Models, see Section 3.1. Next, in Section 3.2 we review the most important methods for the robust inclusion of the range of multivariate functions. Finally, in Section 3.3 we review some of the state of the art Interval Newton methods for the inclusion of the global set of solutions of nonlinear systems of equations.

3.1 Taylor Forms and Taylor Models

We begin our review with a discussion of Taylor Forms because they are the principal tool used to convert nonlinear functions into polynomials, for which all the later methods are developed in the following sections.

Through the following arguments $f(\mathbf{x})$ is a generic real-valued multivariate nonlinear function while $p(\mathbf{x})$ and $q(\mathbf{x})$ are real-valued multivariate polynomials. $\overline{\underline{p}}(\mathbf{x})$ and $\overline{\underline{q}}(\mathbf{x})$ are interval-valued multivariate polynomials, i.e. they are polynomials with at least one interval coefficient.

Definition 3.1.1 (Taylor Form). *An interval-valued polynomial* $\overline{\underline{p}}(\mathbf{x})$ *is a Taylor Form of* $f(\mathbf{x})$ *over the interval box* $\overline{\underline{\mathbf{d}}}$ *if for* $\forall \mathbf{x} \in \overline{\underline{\mathbf{d}}}$ *we have:*

$$f(\mathbf{x}) \in \overline{\underline{p}}(\mathbf{x}).$$

In other words, a Taylor Form is an interval-valued polynomial that bounds the values of the function f at every point in some given region $\overline{\underline{\mathbf{d}}}$. For example, a Taylor Form of $cos(x)$ over the entire real line is:

$$cos(x) \in 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{[-1,1]}{720}x^6.$$

If all the coefficients of the Taylor Form $\overline{\underline{p}}\,(\mathbf{x})$ are real except for the constant term which is an interval we have a *Taylor Model*. For example, a Taylor Model of $cos(x)$ over the interval $x \in [-1,1]$ is:

$$cos(x) \in \left[\frac{719}{720}, \frac{721}{720}\right] - \frac{1}{2}x^2 + \frac{1}{24}x^4.$$

Taylor Models are the sum of a polynomial and an interval called the *interval remainder bound*. We write:

$$f(\mathbf{x}) \in p(\mathbf{x}) + \overline{\underline{r}}\,,$$

where $\overline{\underline{r}}$ is the interval remainder bound.

Taylor Forms and Taylor Models of analytic functions can be obtained from their Taylor expansions, hence the names. For example, for a one dimensional function $f(x)$ we obtain the n^{th} Taylor Form over the interval $\overline{\underline{d}}$:

$$f(x) \in Taylor^{<n>}(f,c)(x) + \frac{(x-c)^{n+1}}{(n+1)!}\left[\mathcal{I}\left(\frac{d^{n+1}f}{dx^{n+1}}\right)\left(\overline{\underline{d}}\right)\right],$$

where $c \in \overline{\underline{d}}$ and $\mathcal{I}\left(\frac{d^{n+1}f}{dx^{n+1}}\right)\left(\overline{\underline{d}}\right)$ is an inclusion function of the n+1st derivative of f evaluated on $\overline{\underline{d}}$. The corresponding Taylor Model can be obtained by replacing the real variable x with the interval $\overline{\underline{d}}$ in the $(x-c)^{n+1}$ term:

$$f(x) \in Taylor^{<n>}(f,c)(x) + \frac{(\overline{\underline{d}}-c)^{n+1}}{(n+1)!}\left[\mathcal{I}\left(\frac{d^{n+1}f}{dx^{n+1}}\right)\left(\overline{\underline{d}}\right)\right].$$

The interval remainder is:

$$\overline{\underline{r}} = \frac{(\overline{\underline{d}}-c)^{n+1}}{(n+1)!}\left[\mathcal{I}\left(\frac{d^{n+1}f}{dx^{n+1}}\right)\left(\overline{\underline{d}}\right)\right].$$

The power of the Taylor Model approximation is that the width of the interval remainder term decreases with order n+1:

$$\mathrm{w}(\,\overline{\underline{r}}\,) = O\left(\frac{\mathrm{w}(\,\overline{\underline{\mathbf{d}}}\,)^{n+1}}{(n+1)!}\right).$$

Thus, Taylor Models can be an efficient way of converting nonlinear functions to polynomials for which interval extensions with higher orders of convergence that the natural extension are presented in Section 3.2. We explain some important points about interval extensions and Taylor Models next.

3.1.1 Taylor Form Interval Extensions

It is important to note that Taylor Forms as defined in the previous section are not inclusion functions because they are defined over $\mathbb{R}^n$ and not over $\mathbb{IR}^n$. However, inclusion functions can be easily generated from a Taylor Form through interval extension. We call this process the *Taylor Form interval extension.*

For example, a Taylor Form interval extension of $cos(x)$ over the entire real line is:

$$\mathcal{R}(cos)(\,\overline{\underline{x}}\,) \in 1 - \frac{1}{2}\,\overline{\underline{x}}^{\,2} + \frac{1}{24}\,\overline{\underline{x}}^{\,4} - \frac{[-1,1]}{720}\,\overline{\underline{x}}^{\,6}.$$

Similarly, a Taylor Model interval extension of $cos(x)$ over the interval $\overline{\underline{x}} \subseteq [-1,1]$ is:

$$\mathcal{R}(cos)(\,\overline{\underline{x}}\,) \in \left[\frac{719}{720}, \frac{721}{720}\right] - \frac{1}{2}\,\overline{\underline{x}}^{\,2} + \frac{1}{24}\,\overline{\underline{x}}^{\,4}.$$

The differences between a Taylor Form (Model) and a Taylor Form (Model) interval extension are often not made clear in the literature. For example, it is often stated that Taylor Models have inclusion orders of arbitrary high orders. In general this is only true for point wise evaluation as was shown at the end of the previous section. In order for a Taylor Model interval extension to be an inclusion function of order $n+1$ one would need to be able to compute the range of the polynomial part with inclusion order greater than or equal to $n+1$. In general, this

is not possible (without a priori knowledge about the nature of the function or running global optimization) even for univariate polynomials.

A Taylor Model interval extension is of the form:

$$\mathcal{R}(f)(\overline{\underline{\mathbf{x}}}) \in \mathcal{I}\left(Taylor^{<N>}(f,\mathbf{c})\right)\left(\overline{\underline{\mathbf{d}}}\right) + \overline{\underline{r}},$$

and the inclusion order is equal to the inclusion order of $\mathcal{I}\left(Taylor^{<N>}(f,\mathbf{c})\right)$.

3.1.2 Taylor Form Chronology

Taylor Forms date as far back as the original texts by Moore, see [Moore 1962, Moore 1966, Moore 1979]. Univariate Taylor forms were studies under the name ultra arithmetic or functoid by Kaucher, Miranker, Rivlin, Epstein and others in the 1980s, see [Epstein et al. 1981, Epstein et al. 1982, Kaucher and Miranker 1983a, Kaucher and Miranker 1983b, Kaucher and Miranker 1984a, Miranker 1983]. The multivariate case was studied in depth by Eckmann et al., see [Eckmann et al. 1986], and by Kaucher in [Kaucher and Miranker 1984b]. The methods were used extensively in various computer assisted proofs. A comprehensive comparison of Taylor Forms interval extensions and other types of inclusion functions can be found in Stahl's thesis [Stahl 1996]. For a more detailed chronology on Taylor Forms and related topics we refer the reader to the paper by Neumaier [Neumaier 2002].

Taylor Models were introduced by Berz et al. as a means of bounding the solutions of differential algebraic equations, see [Berz and Hofstätter 1998]. Berz and his collaborators have written many papers on the subject. A complete repository is available on the web at:

`http://bt.pa.msu.edu/pub/`.

More recently, Taylor Model interval extensions using Bernstein expansions were studied by Nataraj and Kotecha, see [Nataraj and Kotecha 2002].

3.2 Methods for the Robust Inclusion of the Range of Multivariate Functions

In this section we review some of the most important types of interval extensions to date. In light of our discussion of Taylor Models as a way to treat all analytic functions as polynomials we will restrict our presentation to bounding the range of the latter.

3.2.1 Horner Forms

In its pure form, the Horner Form interval extension for polynomials is only an enhancement of the natural extension.[1]

The Horner Form of a univariate polynomial is:

$$p(x) = a_0 + x^{i_1}(a_1 + x^{i_2}(a_2 + x^{i_3}(\ldots + x^{i_k}(a_k + x^{i_{k+1}}a_{k+1})))),$$

and the Horner Form interval extension is:

$$\mathcal{R}(p)(\underline{\overline{x}}) \subseteq \mathcal{H}(p)(\underline{\overline{x}}) = a_0 + \underline{\overline{x}}^{i_1}(a_1 + \underline{\overline{x}}^{i_2}(a_2 + \underline{\overline{x}}^{i_3}(\ldots + \underline{\overline{x}}^{i_k}(a_k + \underline{\overline{x}}^{i_{k+1}}a_{k+1})))).$$

The Horner Form achieves two things:

1. It provides an efficient way of evaluating both the polynomial and the inclusion function (in fact, Horner Forms are the cheapest interval extensions to compute and evaluate), and
2. It generates somewhat tighter bounds than the natural extension.

The narrower bounds produced are due to the subdistributivity of interval multiplication. Note however, that subdistributivity does not apply if powers of intervals are computed directly.[2] When powers are computed directly, power series evaluation may produce tighter bounds than the Horner scheme—more research is needed to determine exactly when this happens. For ex-

[1] Remainder: in this thesis we call the interval extension of the power series expression of a polynomial (MacLaurin form) the natural extension. Some authors call the Horner Form interval extension the natural extension.

[2] Instead of using repeated multiplication.

ample, Horner evaluation of x^2+3x on the interval $[-1,1]$ produces the range $[-4,4]$ while the natural extension with powers evaluated directly produces the tighter range $[-3,4]$:

$$[-1,1]([-1,1]+3) = [-1,1]\cdot[2,4] = [-4,4]$$
$$[-1,1]^2+3[-1,1] = [0,1]+[-3,3] = [-3,4]$$

Things are slightly more complicated in the multivariate case, as there are many different ways in which Horner Forms that can be computed. In general, one would choose an ordering of the variables and write Horner Forms with respect to each variable successively until the whole expression is suitably factored. The author is not aware of any publications that provide an optimal ordering of variables.

3.2.1.1 Summary of Properties

The Horner Form interval extension has the following properties (for proofs and added details we refer the reader to Section 3.1 of V. Stahl's thesis [Stahl 1996]):

- **Inclusion Order:** $O\left(\mathrm{w}\left(\underline{\overline{\mathbf{d}}}\right)\right)$. The Horner Form interval extension has excess width that decreases linearly with the width of the input interval. Therefore, the Horner form has the same inclusion order as the natural extension.
- **Inclusion Monotonicity:** The Horner Form interval extension is isotonic.
- **Non-Overestimation:** The Horner Form interval extension has zero overestimation[3] when all the intermediate intervals arising during its evaluation do not straddle zero. This is equivalent to the condition that the natural extension has no overestimation, see Section 2.3.3.

The Horner Form can be trivially extended to arbitrary Taylor Forms:

$$p(x) = a_0+(x-c)^{i_1}(a_1+(x-c)^{i_2}(a_2+(x-c)^{i_3}(\ldots+(x-c)^{i_k}(a_k+(x-c)^{i_{k+1}}a_{k+1})))).$$

[3] Zero excess width.

Once again, many different Horner Forms exist in the multivariate case. We will call the combination of a Taylor Form with the Horner evaluation the Horner-Taylor Form. The Horner-Taylor Form can be used to enhance many of the interval extensions presented in the following chapters with the same benefits listed above (when direct power computations are not used).

3.2.2 Centered and Mean Value Forms

The Centered Form is the simplest of the quadratically convergent interval extensions. Centered Forms have been studied extensively since the beginning of interval analysis. The initial idea was published by Moore [Moore 1966]. The Centered Form of a univariate function f is:

$$f(x) = f(c) + g_c(x,c)(x-c),$$

where c is called the *center*. The Centered Form interval extension is:

$$\mathcal{R}(f)(\underline{x}) \subseteq \mathcal{C}(f)(\underline{x}) = f(c) + \mathcal{I}(g_c)(\underline{x},c)(\underline{x}-c),$$

where $\mathcal{I}(g_c)$ is any inclusion function of g_c, and the center c is usually a point inside the interval $\underline{x}$. The most common choice of center is the midpoint of $\underline{x}$. Thus, the Centered Form interval extension may require the computation of a new expression of g_c for every input interval $\underline{x}$. The multivariate case is similar.

Hansen developed the Centered Form for polynomials in [Hansen 1969]. Explicit expressions for g proved difficult to obtain for functions other than polynomials. Centered Forms for rational functions of the form p/q were discussed by Ratschek in [Ratschek 1980]. Even in this relatively simple case the computation of an expression of g_c requires all partial derivatives of p and q up to their respective degrees, which made the Centered Form computationally expensive. Expanding f into a suitable Taylor Model and computing the Centered Form of the polynomial term is a simpler but less accurate solution.

A closely related interval extension is the *Mean Value Form*. The mean value interval exten-

sion is nothing else than the interval extension of the well-known mean value theorem:

$$f(x) = f(c) + f'(\xi)(x-c),$$

where ξ is some number between x and c. While the exact value of ξ cannot be easily computed, an inclusion function is:

$$\mathcal{R}(f)(\underline{x}) \subseteq \mathcal{M}(f)(\underline{x}) = f(c) + \mathcal{I}(f')(\underline{x})(\underline{x}-c).$$

Although the similarity with Centered Forms is apparent, the Mean Value Form does not fit the definition of the Centered Form given above. Krawczyk and Nickel were the first to develop a general theory that covered both Moore's Centered Form and the Mean Value Form in [Krawczyk and Nickel 1981]. Their paper also gave a proof of the quadratic inclusion order of the generalized Centered Form.

The excess width of the Mean Value Form can in general be reduced if one uses slope functions instead of derivatives. Slope function are defined as:

$$\text{Slope function: } g_c(x) = \frac{f(x)-f(c)}{x-c},$$

and the Slope Form has the following expression:

$$\mathcal{R}(f)(\underline{x}) \subseteq \mathcal{S}(f)(\underline{x}) = f(c) + \mathcal{I}(g_c)(\underline{x})(\underline{x}-c).$$

Another improvement, due to Baumann, see [Baumann 1988], is the *Bicentered (Baumann) Form*. Baumann showed how to compute two centers c_* and c^* such that the intersection of the two corresponding Centered Forms produce the smallest bound for the range.

3.2.2.1 Summary of Properties

We summarize the properties of the Centered Form interval extension below (for proofs and added details we refer the reader to Sections 3.1 and 3.2 of V. Stahl's thesis [Stahl 1996]):

- **Inclusion Order:** $O\left(\left(\mathrm{w}\left(\underline{\overline{\mathbf{d}}}\right)\right)^2\right)$. Centered Form interval extensions have excess width that decreases quadratically with respect to the width of the input interval. In general, Centered Forms produce better range inclusions than the natural extension for narrow input intervals. On wider intervals the natural extension is usually better.

- **Inclusion Monotonicity:** The Centered Form interval extension is isotonic.

- **Non-Overestimation:** If the center is not on the boundary of the interval $\underline{\overline{x}}$ the Centered Form always overestimates the range.

3.2.3 Taylor Forms Revisited

Another way to look at Taylor Forms is as a generalization of the Centered Form. If the interval extension of g_c (see the previous section for the definition of g_c) is itself a Centered Form then one obtains the second order Horner-Taylor Form interval extension:

$$\mathcal{R}(f)(\underline{\overline{x}}) \subseteq f(c) + (g(c) + \mathcal{I}(h)(\underline{\overline{x}}, c)(\underline{\overline{x}} - c))(\underline{\overline{x}} - c).$$

Taylor Forms of any order can be obtained by recursion—although we should remember that the inclusion order will stay quadratic.

3.2.4 Bernstein Forms

The Bernstein Form inclusion function is the interval extension of the expansion of a function with respect to the (non-orthogonal) basis of Bernstein polynomials. The connections with the theory of Bezier splines are well known.

The **j**-th multivariate *Bernstein polynomial* of multi-order **n** is defined as:

$$\mathbf{j}\text{-th Bernstein polynomial of degree } \mathbf{n}\text{: } B_{(\mathbf{j},\mathbf{n})}(\mathbf{x}) = \begin{pmatrix} \mathbf{n} \\ \mathbf{j} \end{pmatrix} \mathbf{x}^{\mathbf{j}}(\mathbf{1}-\mathbf{x})^{\mathbf{k}-\mathbf{j}},$$

where $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{j}, \mathbf{n} \in \mathbb{N}^m$ such that $j_i \leq n_i,\ \forall i \leq m$.

3.2.4.1 Bernstein Forms for Polynomials

Every multivariate polynomial $p : \mathbb{R}^m \to \mathbb{R}$ of multi-degree $\mathbf{n}$ can be written as a linear combination of the Bernstein polynomials of order $\mathbf{n}$ and greater:

$$\text{Bernstein expansion of order } n\text{: } p(\mathbf{x}) = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}} b_{(\mathbf{j},\mathbf{n})} B_{(\mathbf{j},\mathbf{n})}(\mathbf{x}).$$

If p is of the form:

$$p(\mathbf{x}) = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}} a_{\mathbf{i}} \mathbf{x}^{\mathbf{i}},$$

then the coefficients of the Bernstein expansion can be computed with the following formula:

$$\mathbf{j}\text{-th Bernstein coefficient (of degree } \mathbf{n}\text{): } b_{(\mathbf{j},\mathbf{n})} = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{j}} a_{\mathbf{i}} \frac{\begin{pmatrix} \mathbf{j} \\ \mathbf{i} \end{pmatrix}}{\begin{pmatrix} \mathbf{n} \\ \mathbf{i} \end{pmatrix}}.$$

Because the Bernstein basis is not orthogonal, Bernstein coefficients of a polynomial of degree $\mathbf{n}$ with respect to a Bernstein basis of higher order are usually not equal to zero. This is an important property and is related to the process of *degree elevation*, whereby the coefficients of a Bernstein expansion of higher order can be computed directly from the coefficients of a lower order expansion.

It is well known that the range of p on the unit interval box $[0,1]^m$ is enclosed between the smallest and largest values of the Bernstein coefficients. Thus, we can define the *Bernstein Form* interval extension of order $\mathbf{n}$ as follows:

$$\text{Bernstein Form: } \mathcal{B}(p)([0,1]^m) = \left[\min_{\mathbf{j}}(b_{(\mathbf{j},\mathbf{n})}), \max_{\mathbf{j}}(b_{(\mathbf{j},\mathbf{n})}) \right].$$

To compute the range over an arbitrary interval $\overline{\underline{\mathbf{x}}}$ the polynomial must first be composed with the appropriate affine translation and scaling that maps $\overline{\underline{\mathbf{x}}}$ into $[0,1]$, before computing the Bernstein expansion. This is equivalent to computing Taylor coefficients with an additional

scaling of the variable. Therefore, the Bernstein Form is more expensive to compute than the Taylor Form; the added cost is that of scaling the Taylor coefficients plus the computation of the Bernstein coefficients minus the cost of evaluating the range of the polynomial in Taylor Form.

As we mentioned before, a polynomial of degree **n** can be exactly represented by a Bernstein expansion of order **n** or larger. In general, Bernstein expansions of larger orders produce tighter inclusion functions. Since the coefficients of the higher order expansion can be computed relatively inexpensively through degree elevation, one could try to dynamically improve enclosures this way.

3.2.4.2 Bernstein Forms for Other Types of Functions

Bernstein inclusion functions for functions other than polynomials can be generated by first approximating the function with a Taylor Model or another equivalent polynomial expansion with interval, and then computing the Bernstein form of the approximation. The mapping to the interval $[0,1]$ can be done concurrently with the computation of the approximating polynomial.

The classic Bernstein approximation of a function f is defined as:

$$\text{Bernstein approximation: } B_{(n)}(f,\mathbf{x}) = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}} f\left(\frac{\mathbf{j}}{\mathbf{n}}\right) B_{(\mathbf{j},\mathbf{n})}(\mathbf{x}).$$

The above approximation can be used to define a Bernstein Form provided that bounds for $| f(\mathbf{x}) - B_{(n)}(f,\mathbf{x}) |$ can be computed. However, this type of Bernstein approximations converge too slowly to be efficient in practice. Taylor, Chebyshev or minimax approximations should generate better results.

Of course, higher order Bernstein expansions produce better inclusion functions than lower order ones, at the expense of more computation. One has to first decide the degree of the approximation polynomial then the order of the Bernstein Form of the polynomial.

3.2.4.3 Short Chronology

The idea of using Bernstein expansions to bound the range of functions dates back to [Rivlin 1970]. The first generalization to intervals other than $[0,1]$ was given by Rokne in

[Rokne 1977]. Rokne went on to publish several papers on the subject, among which we mention [Rokne 1978] and [Rokne 1981]. The multivariate case was studied by Garloff in [Garloff 1985].

3.2.4.4 Summary of Properties

We summarize the properties of the Bernstein Form interval extension below (for proofs and added details we refer the reader to Section 3.4 of V. Stahls thesis [Stahl 1996]):

- **Inclusion Order:** $O\left(\left(\mathrm{w}\left(\underline{\overline{\mathbf{d}}}\right)\right)^2\right)$. The excess width of the Bernstein Form interval extension decreases quadratically with the width of the input interval.
- **Inclusion Monotonicity:** The Bernstein Form interval extension is isotonic.
- **Non-Overestimation:** The Bernstein Form interval extension gives the exact range iff $\min_j(b_{(\mathbf{j},\mathbf{n})}) \in \{b_{(\mathbf{0},\mathbf{n})}, b_{(\mathbf{n},\mathbf{n})}\}$ and $\max_j(b_{(\mathbf{j},\mathbf{n})}) \in \{b_{(\mathbf{0},\mathbf{n})}, b_{(\mathbf{n},\mathbf{n})}\}$.

3.3 Interval Newton Methods for the Inclusion of the Roots of Nonlinear Systems of Equations

In this section we discuss improvements of the basic divide and conquer algorithm for the inclusion of the roots of nonlinear systems presented in Section 2.4. The methods we discuss here are versions of Interval Newton—an extension of the familiar Newton's method to equations with interval coefficients.

Interval Newton methods are based on the availability of an *Interval Newton operator*:

$$\text{Interval Newton operator: } \mathit{IntervalNewtonOperator}\left(\underline{\overline{\mathbf{x}}}\right) \subseteq \underline{\overline{\mathbf{x}}}\,.$$

The Interval Newton operator is a contraction.

The pseudocode of a generic Interval Newton algorithm is shown in Figure 3.1. The **GenericNewtonContraction** procedure uses Interval Newton operators to shrink candidate intervals as much as possible before performing any subdivision. Several possible implementations are discussed in the next sections.

```
IntervalNewtonSolve (
  in f:    function whose solutions we are seeking
  in d:    domain interval
  in ε:    maximum size of a solution interval
  in γ:    Interval Newton contraction coefficient
  out S_ε: [empty] interval covering of the solutions of f in d
)
{
  create stack: [empty] stack of subintervals to be examined;
  put d on stack;
  while ( stack is not empty ) do
  {
    pop x from stack;
    // estimate the range of f on x
    compute y = J(f)(x);
    if ( 0 ∈ y )
    // there could be solutions in x
    {
      set x = GenericNewtonContraction ( in f, in x, in ε, in γ );
      if ( x ≠ ∅ )
      {
        if ( w(x) < ε )
          append x to S_ε;
        else
        {
          Subdivide ( in x, out x_1, out x_2 );
          put x_1 on stack;
          put x_2 on stack;
        }
      }
    }
  }
  return; // search is exhausted
}
```

Figure 3.1: The generic Interval Newton algorithm for solving nonlinear systems of equations. The values between square brackets listed next to variable declarations represent initial values.

3.3.1 Linear Interval Equations

Interval Newton operators are derived from linearizations of the (system of) equations. Reminiscent of the Centered Form, the original nonlinear equations 2.1 can be linearized as follows:

$$\begin{aligned}
&\mathbf{f}(\mathbf{x}) = 0 \;\Leftrightarrow \\
&\mathbf{f}(\mathbf{c}) + \mathbf{g_c}(\mathbf{x})(\mathbf{x}-\mathbf{c}) = 0 \;\Leftrightarrow \\
&\mathbf{g_c}(\mathbf{x})(\mathbf{x}-\mathbf{c}) = -\mathbf{f}(\mathbf{c}),
\end{aligned}$$

where $\mathbf{g_c}(\mathbf{x})$ is a slope function of $\mathbf{f}(\mathbf{x})$:

$$\text{Slope function: } \mathbf{g_c}(\mathbf{x}) = \frac{\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{c})}{\mathbf{x}-\mathbf{c}}.$$

By bounding the slope function $\mathbf{g_c}(\mathbf{x})$ over $\overline{\underline{\mathbf{x}}}$ one obtains a linear interval equation:

$$\overline{\underline{\mathbf{G}}}_{\mathbf{c}}(\mathbf{x}-\mathbf{c}) = -\mathbf{f}(\mathbf{c}),$$

where $\overline{\underline{\mathbf{G}}}_{\mathbf{c}} = \mathcal{I}(\mathbf{g_c})(\overline{\underline{\mathbf{x}}})$. All the solutions of the nonlinear equation 2.1 are included in the solution set of the above linear interval equation.

The Mean Value Form can also be used to linearize the system in much the same way. The interval matrix $\overline{\underline{\mathbf{G}}}_{\mathbf{c}}$ is replaced by the interval hull of the set of Jacobian matrices of $\mathbf{f}(\mathbf{x})$ over $\overline{\underline{\mathbf{x}}}$:

$$\{\mathbf{f}(\mathbf{x}) = 0\} \Leftrightarrow \left\{\mathbf{f}'(\xi)(\mathbf{x}-\mathbf{c}) = -\mathbf{f}(\mathbf{c})\right\},$$

resulting in the linear interval equation below:

$$\overline{\underline{J}}(\mathbf{x}-\mathbf{c}) = -\mathbf{f}(\mathbf{c}),$$

where $\overline{\underline{J}} = \mathcal{I}(\mathbf{f}')(\overline{\underline{\mathbf{x}}})$. For obvious reasons, the solution set of the linear interval equation above also includes all the solutions of the original equation.

We reduced the problem of finding an interval covering of the nonlinear equation 2.1 to that

of finding a covering of a linear interval equation of the form:

$$\overline{\underline{A}}\,(\mathbf{x}) = \overline{\underline{\mathbf{b}}}\,. \tag{3.1}$$

Following common practice we replaced the real vector on the right hand side of the above equation with the interval valued vector $\overline{\underline{\mathbf{b}}}$.

3.3.2 The Interval Newton Operator

```
IntervalNewtonContraction (
  in f:    function whose solutions we are seeking
  in x̄:    initial bounds of solution
  in ε:    maximum size of a solution interval
  in γ:    Interval Newton contraction coefficient
)
{
  bool continue = true;
  while (continue) do
  {
    set Ā = J(f')(x̄);
    set x̄* = x̄ ∩ (m(x̄) − Ā^-1 f(m(x̄)));
    if (w(x̄*) > γ w(x̄)) // not enough improvement
      set continue = false;
    if (x̄* = ∅) // no solutions in x̄
      set continue = false;
    set x̄ = x̄*;
  }
  return x̄; // search is exhausted
}
```

Figure 3.2: A recursive Interval Newton contraction algorithm for solving nonlinear systems of equations. This function replaces the generic **NewtonContraction** in the Interval Newton algorithm in Figure 3.1.

We now describe the original Interval Newton operator, due to Hansen, see [Hansen 1978]. First we need to define some concepts related to interval matrices.

A square interval matrix $\overline{\underline{A}}$ is called *regular* if it does not contain any singular matrices:

$$\text{Regular interval matrix } \overline{\underline{A}}\text{: } \forall A \in \overline{\underline{A}}\,,\ \det(A) \neq 0.$$

The *inverse of a regular interval matrix* $\overline{\underline{A}}$ is defined as the interval hull of the set of inverses of the real matrices contained inside it:

$$\text{Inverse of regular interval matrix: } \overline{\underline{A}}^{\,-1} = \left[\left[\, A^{-1} \mid \forall A \in \overline{\underline{A}}\, \right]\right].$$

We are now ready to define the *Interval Newton operator* for the linear interval equation 3.1:

$$\text{Interval Newton operator: } \mathit{NewtonOperator}\left(\overline{\underline{\mathbf{x}}}\right) = \overline{\underline{\mathbf{x}}} \cap \left(\overline{\underline{A}}^{\,-1}\,\overline{\underline{\mathbf{b}}}\right).$$

The Interval Newton operator has two very important properties:

1. If $\mathit{NewtonOperator}\left(\overline{\underline{\mathbf{x}}}\right) \subset \operatorname{int}\left(\overline{\underline{\mathbf{x}}}\right)$ then there is a *unique* solution inside $\overline{\underline{\mathbf{x}}}$, and
2. If $\mathit{NewtonOperator}\left(\overline{\underline{\mathbf{x}}}\right) = \emptyset$ then there are no solutions inside $\overline{\underline{\mathbf{x}}}$.

In the case of a nonlinear equation $\mathbf{f}(\mathbf{x}) = 0$ and a mean value linearization, the Interval Newton operator becomes:

$$\mathit{NewtonOperator}\left(\overline{\underline{\mathbf{x}}}\right) = \overline{\underline{\mathbf{x}}} \cap \left(\mathrm{m}\left(\overline{\underline{\mathbf{x}}}\right) - \mathcal{J}\left(\mathbf{f}'\right)\ \overline{\underline{\mathbf{x}}}^{\,-1}\mathbf{f}\left(\mathrm{m}\left(\overline{\underline{\mathbf{x}}}\right)\right)\right).$$

A recursive algorithm using the Interval Newton operator is shown in Figure 3.2. This algorithm replaces the generic **NewtonContraction** function in the Newton solver shown in Figure 3.1.

3.3.3 Preconditioning

Assume the square interval matrix $\overline{\underline{A}}$ contains at least one nonsingular matrix A. If we multiply both sides of 3.1 by A^{-1} we obtain the equivalent linear interval equation below:

$$\overline{\underline{A}}^{\,*}\mathbf{x} = \overline{\underline{\mathbf{b}}}^{\,*}, \tag{3.2}$$

where we set $\underline{\overline{A}}^* = A^{-1}\,\underline{\overline{A}}$ and $\underline{\overline{\mathbf{b}}}^* = A^{-1}\,\underline{\overline{\mathbf{b}}}$. If $\underline{\overline{A}}$ from the original equation is relatively narrow then $\underline{\overline{A}}^*$ is likely to be a narrow interval around the identity, meaning the diagonal entries will be narrow intervals around one, while the rest of the entries will be narrow intervals close to zero. This can simplify the computation of the solution set of 3.1 as shown in the next two sections.

3.3.4 The Krawczyk Operator

The Krawczyk operator was introduced by Krawczyk in [Krawczyk and Neumaier 1984]. We describe the main idea below.

Consider the preconditioned linear interval equation 3.2. If we add $\mathbf{x}$ to both sides of the equation we get:

$$\mathbf{x} + \underline{\overline{A}}^*\mathbf{x} = \mathbf{x} + \underline{\overline{\mathbf{b}}}^*.$$

Finally, move $\underline{\overline{A}}^*\mathbf{x}$ to the other side and rearrange to get the following fixed point equation:

$$\mathbf{x} = \underline{\overline{\mathbf{b}}}^* - \left(\underline{\overline{A}}^* - I\right)\mathbf{x}.$$

If the preconditioned interval matrix $\underline{\overline{A}}^*$ is sufficiently close to the identity matrix the fixed point equation above is also a contraction. Thus, the *Krawczyk operator* is:

$$\text{Krawczyk operator: } \mathit{KrawczykOperator}(\underline{\overline{\mathbf{x}}}) = \underline{\overline{\mathbf{x}}} \cap \left(\underline{\overline{\mathbf{b}}}^* - \left(\underline{\overline{A}}^* - I\right)\underline{\overline{\mathbf{x}}}\right).$$

The Krawczyk operator shares the same properties as the interval Newton operator listed in Section 3.3.2 In addition, when the uniqueness conditions are satisfied, the Krawczyk operator converges quadratically to the solution.

In the case of a nonlinear equation $\mathbf{f}(\mathbf{x}) = 0$ and a mean value linearization, the Krawczyk operator becomes:

$$\begin{aligned}
\mathit{KrawczykOperator}(\underline{\overline{\mathbf{x}}}) &= \underline{\overline{\mathbf{x}}} \cap \left(\mathrm{m}(\underline{\overline{\mathbf{x}}}) - C^{-1}\mathbf{f}(\mathrm{m}(\underline{\overline{\mathbf{x}}})) - \left(C^{-1}\,\underline{\overline{A}} - I\right)\underline{\overline{\mathbf{x}}}\right), \text{ where} \\
\underline{\overline{A}} &= \mathcal{J}(\mathbf{f}')(\underline{\overline{\mathbf{x}}}), \text{ and} \\
C &= \mathrm{m}\left(\underline{\overline{A}}\right).
\end{aligned}$$

```
KrawczykContraction (
  in f:    function whose solutions we are seeking
  in x:    initial bounds of solution
  in ε:    maximum size of a solution interval
  in γ:    Interval Newton contraction coefficient
)
{
  bool continue = true;
  while (continue) do
  {
    set A = J(f')(x);
    set C = m(A);
    set x* = x ∩ (m(x) - C^-1 f(m(x)) - (C^-1 A - I) x);
    if (w(x*) > γ w(x)) // not enough improvement
      set continue = false;
    if (x* = ∅) // no solutions in x
      set continue = false;
    set x = x*;
  }
  return x; // search is exhausted
}
```

Figure 3.3: A recursive Krawczyk contraction algorithm for solving nonlinear systems of equations. This function replaces the generic **NewtonContraction** in the Interval Newton algorithm in Figure 3.1.

A recursive algorithm using the Krawczyk operator is shown in Figure 3.2. This algorithm replaces the generic **NewtonContraction** function in the Newton solver shown in Figure 3.1.

3.3.5 The Hansen-Sengupta Algorithm

Another method for solving the preconditioned linear interval equation 3.2 uses an interval version of Gauss-Seidel iteration. The method was introduced by Hansen and Sengupta in [Hansen and Sengupta 1981].

The *interval Gauss-Seidel operator* is defined as follows:

$$\text{Gauss-Seidel operator: } \mathit{GaussSeidelOperator}(\overline{\mathbf{x}}, i) = \overline{\mathbf{x}}_i \cap \left(\mathrm{m}(\overline{\mathbf{x}}) - \frac{\overline{\mathbf{b}}_i^* - \sum_{j \neq i} \overline{A}_{ij}^* \overline{\mathbf{x}}_j}{\overline{A}_{ii}^*} \right).$$

HansenSenguptaContraction (
  in f: function whose solutions we are seeking
  in $\overline{\underline{\mathbf{x}}}$**:** initial bounds of solution
  in ε**:** maximum size of a solution interval
  in γ**:** Interval Newton contraction coefficient
)
{
  bool $continue = true$;
  while $(continue)$ **do**
  {
    set $\overline{\underline{A}} = \mathcal{J}(\mathbf{f}')(\overline{\underline{\mathbf{x}}})$;
    set $\overline{\underline{A}}^* = \mathrm{m}\left(\overline{\underline{A}}\right)^{-1} \overline{\underline{A}}$;
    set $\overline{\underline{\mathbf{x}}}^* = \overline{\underline{\mathbf{x}}}$;
    set $\overline{\underline{\mathbf{b}}}^* = -\mathrm{m}\left(\overline{\underline{A}}\right)^{-1} \mathbf{f}(\mathrm{m}(\overline{\underline{\mathbf{x}}}))$;
    for $(i = 1;\ i \le n;\ i{+}{+})$;
      set $\overline{\underline{\mathbf{x}}}_i^* = \overline{\underline{\mathbf{x}}}_i^* \cap \left(\mathrm{m}(\overline{\underline{\mathbf{x}}}_i) - \dfrac{\overline{\underline{\mathbf{b}}}_i^* - \sum_{j \neq i} \overline{\underline{A}}_{ij}^* \overline{\underline{\mathbf{x}}}_j^*}{\overline{\underline{A}}_{ii}^*} \right)$;
    if $(\mathrm{w}(\overline{\underline{\mathbf{x}}}^*) > \gamma\, \mathrm{w}(\overline{\underline{\mathbf{x}}}))$ // not enough improvement
      set $continue = false$;
    if $(\overline{\underline{\mathbf{x}}}^* = \emptyset)$ // no solutions in $\overline{\underline{\mathbf{x}}}$
      set $continue = false$;
    set $\overline{\underline{\mathbf{x}}} = \overline{\underline{\mathbf{x}}}^*$;
  }
  return $\overline{\underline{\mathbf{x}}}$; // search is exhausted
}

Figure 3.4: A recursive Hansen-Sengupta contraction algorithm for solving nonlinear systems of equations. This function replaces the generic **NewtonContraction** in the Interval Newton algorithm in Figure 3.1.

The Gauss-Seidel operator takes in an interval vector $\overline{\underline{\mathbf{x}}}$ and an index i and returns an interval representing the updated i^{th} component of $\overline{\underline{\mathbf{x}}}$.

The algorithm for the Hansen-Sengupta contraction is shown in Figure 3.4. The Gauss-Seidel iteration is performed inside the "for" loop. Note that the results of each Gauss-Seidel step are immediately used in all subsequent computations.

3.3.6 Linear Tightening

Linear tightening was introduced by V. Stahl and is described in detail in his Ph.D. thesis, see [Stahl 1996]. It is similar to the Hansen-Sengupta method without preconditioning, i.e. it performs Gauss-Seidel on the linear interval equation without preconditioning it. This idea is somewhat similar to the ideas we present in Chapter 5, however the linearizations used are different. In his thesis, Stahl reports that significant speedups can be achieved using tightening.

Chapter 4

Corner Taylor Form Inclusion Functions

4.1 Introduction

In this chapter we present the first contribution of the thesis: the *Corner Taylor form inclusion function.* The main benefit of the Corner Taylor Form is that it always produces bounds that are tighter than those produced by the natural extension. To date, the Corner Taylor Form is the only quadratically convergent inclusion function with this property.

In this and subsequent chapters we ignore the effects of roundoff errors and assume that all operations on reals are exact. Of course, roundoff errors will affect the results we present to a small degree. However, this effect only becomes significant when interval arguments of expressions are very narrow.

The chapter begins with a discussion of sign coherent intervals and posynomial decompositions.

4.2 Sign-Coherent Intervals

Sign coherent intervals were introduced in Section 2.2. If $0 \leq \underline{x}$ the interval is called a *positive interval*, and we write $\overline{\underline{x}} > 0$ or $\overline{\underline{x}} \in \mathbb{IR}_+$. Conversely, if $\overline{x} \leq 0$ we call the interval a *negative interval*, and write $\overline{\underline{x}} < 0$ or $\overline{\underline{x}} \in \mathbb{IR}_-$. If $\underline{x} = 0$ or $\overline{x} = 0$ we say the interval is a *zero-bound interval*. A zero-bound positive interval is called a *zero-positive interval*. Similarly, a zero-

bound negative interval is called a *zero-negative interval*. Positive and zero-positive intervals are called *non-negative intervals*, and we write $\overline{\underline{x}} \geq 0$ or $\overline{\underline{x}} \in \mathbb{IR}_{+}^{*}$, while negative and zero-negative intervals are called *non-positive intervals*, and we write $\overline{\underline{x}} \leq 0$ or $\overline{\underline{x}} \in \mathbb{IR}_{-}^{*}$. Non-negative and non-positive intervals are the two types of *sign coherent intervals*. An interval that has both positive and negative values is called a *zero-straddling interval*.

In summary:

Positive interval:	$\overline{\underline{x}} > 0$ iff $\underline{x} > 0$,
Negative interval:	$\overline{\underline{x}} < 0$ iff $\overline{x} < 0$,
Zero-positive interval:	$\overline{\underline{x}} \geq 0$ iff $\underline{x} = 0$,
Zero-negative interval:	$\overline{\underline{x}} \leq 0$ iff $\overline{x} = 0$,
Zero-straddling interval:	$\overline{\underline{x}} \lessgtr 0$ iff $\overline{x} > 0$ and $\underline{x} < 0$.

It is well known that, in general, interval multiplication is not distributive with respect to addition. However, there are cases when distributivity does hold. As we are going to use these cases during the course of our proofs, we review them here:

Distributivity holds:

$$\begin{cases} \overline{\underline{x}}\left(\overline{\underline{y}} \pm \overline{\underline{z}}\right) = \overline{\underline{x}}\,\overline{\underline{y}} \pm \overline{\underline{x}}\,\overline{\underline{z}} \text{ if } \underline{x} = \overline{x} \text{ (thin factor)}, \\ \overline{\underline{x}}\left(\overline{\underline{y}} + \overline{\underline{z}}\right) = \overline{\underline{x}}\,\overline{\underline{y}} + \overline{\underline{x}}\,\overline{\underline{z}} \text{ if } \overline{\underline{y}} \geq 0 \text{ and } \overline{\underline{z}} \geq 0 \text{ (non-negative terms)}, \\ \overline{\underline{x}}\left(\overline{\underline{y}} + \overline{\underline{z}}\right) = \overline{\underline{x}}\,\overline{\underline{y}} + \overline{\underline{x}}\,\overline{\underline{z}} \text{ if } \overline{\underline{y}} \leq 0 \text{ and } \overline{\underline{z}} \leq 0 \text{ (non-positive terms)}, \\ \overline{\underline{x}}\left(\overline{\underline{y}} - \overline{\underline{z}}\right) = \overline{\underline{x}}\,\overline{\underline{y}} - \overline{\underline{x}}\,\overline{\underline{z}} \text{ if } \overline{\underline{y}} \geq 0 \text{ and } \overline{\underline{z}} \leq 0 \text{ (non-negative terms variation)}, \\ \overline{\underline{x}}\left(\overline{\underline{y}} - \overline{\underline{z}}\right) = \overline{\underline{x}}\,\overline{\underline{y}} - \overline{\underline{x}}\,\overline{\underline{z}} \text{ if } \overline{\underline{y}} \leq 0 \text{ and } \overline{\underline{z}} \geq 0 \text{ (non-positive terms variation)}, \\ \overline{\underline{x}}\left(\overline{\underline{y}} \pm \overline{\underline{z}}\right) = \overline{\underline{x}}\,\overline{\underline{y}} \pm \overline{\underline{x}}\,\overline{\underline{z}} \text{ if } \overline{\underline{x}} \geq 0,\ \underline{y} = 0 \text{ and } \overline{z} = 0 \\ \qquad\qquad \text{(positive factor, zero-straddling terms)}, \\ \overline{\underline{x}}\left(\overline{\underline{y}} \pm \overline{\underline{z}}\right) = \overline{\underline{x}}\,\overline{\underline{y}} \pm \overline{\underline{x}}\,\overline{\underline{z}} \text{ if } \overline{\underline{x}} \leq 0,\ \underline{y} = 0 \text{ and } \overline{z} = 0 \\ \qquad\qquad \text{(negative factor, zero-straddling terms)}. \end{cases}$$

4.3 Sign-Coherent Interval Decomposition

Any interval $\underline{\bar{x}}$ can be decomposed into the difference of two non-negative intervals, $\underline{\bar{x}}_{\oplus}$ and $\underline{\bar{x}}_{\ominus}$, as follows:

$$\text{Non-negative part:} \quad \underline{\bar{x}}_{\oplus} = \begin{cases} \underline{\bar{x}} \cap \mathbb{IR}_+, \text{ if } \underline{\bar{x}} \cap \mathbb{IR}_+ \text{ is not empty} \\ [0,0], \text{ otherwise} \end{cases}$$

$$\text{Non-positive part:} \quad \underline{\bar{x}}_{\ominus} = \begin{cases} -\underline{\bar{x}} \cap \mathbb{IR}_+, \text{ if } -\underline{\bar{x}} \cap \mathbb{IR}_+ \text{ is not empty} \\ [0,0], \text{ otherwise} \end{cases}$$

$$\text{SC decomposition:} \quad \underline{\bar{x}} = \underline{\bar{x}}_{\oplus} - \underline{\bar{x}}_{\ominus}.$$

We call the above the *sign-coherent (SC) decomposition.* If $\underline{\bar{x}}$ is a zero-straddling interval then its SC decomposition will consist of two zero-positive intervals. If $\underline{\bar{x}}$ does not contain zero the SC decomposition will be comprised of one positive interval and the zero interval. For example, let $\underline{\bar{x}} = [-3,7]$. Then $\underline{\bar{x}}_{\oplus} = [0,7]$, $\underline{\bar{x}}_{\ominus} = [0,3]$. If $\underline{\bar{y}} = [3,7]$, then $\underline{\bar{y}}_{\oplus} = \underline{\bar{y}} = [3,7]$, and $\underline{\bar{y}}_{\ominus} = [0,0]$.

SC decompositions of intervals have the following useful properties:

Proposition 4.3.1. *Let $\underline{\bar{x}}$ be a sign-coherent interval and let $\underline{\bar{y}}$ be any interval. Then:*

1. $(-\underline{\bar{x}}) = \underline{\bar{x}}_{\ominus} - \underline{\bar{x}}_{\oplus}$,
2. $\underline{\bar{x}}\,\underline{\bar{y}} = \underline{\bar{x}}\,(\underline{\bar{y}}_{\oplus} - \underline{\bar{y}}_{\ominus}) = \underline{\bar{x}}\,\underline{\bar{y}}_{\oplus} - \underline{\bar{x}}\,\underline{\bar{y}}_{\ominus}$.

Proof:

1. $(-\underline{\bar{x}}) = (-\underline{\bar{x}})_{\oplus} - (-\underline{\bar{x}})_{\ominus} = \underline{\bar{x}}_{\ominus} - \underline{\bar{x}}_{\oplus}$,
2. If $\underline{\bar{y}}$ is a zero-straddling interval then:

$$\underline{\bar{x}}\,\underline{\bar{y}} = \left[\bar{x}\,\underline{y}\,, \bar{x}\,\bar{y}\right] = \underline{\bar{x}}\,\underline{\bar{y}}_{\oplus} - \underline{\bar{x}}\,\underline{\bar{y}}_{\ominus}.$$

If $\underline{\bar{y}}$ is not zero-straddling then either $\underline{\bar{y}}_{\oplus}$ or $\underline{\bar{y}}_{\ominus}$ is the zero interval $[0,0]$ and the result

follows trivially.

□

4.4 Posynomials

Posynomials are polynomials with non-negative coefficients:

$$\text{Posynomial: } p(\mathbf{x}) = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}} a_{\mathbf{i}} \mathbf{x}^{\mathbf{i}},\ a_{\mathbf{i}} \geq 0.$$

We extend this notion to include posynomials with interval-valued coefficients.

Definition 4.4.1. *A multivariate interval posynomial* $\underline{\overline{p}}$ *is a multivariate polynomial with all non-negative interval coefficients:*

$$\underline{\overline{p}}\,(\mathbf{x}) = \sum_{\mathbf{i} \in I} \underline{\overline{a}}_{\mathbf{i}} \mathbf{x}^{\mathbf{i}},\ \underline{\overline{a}}_{\mathbf{i}} \in \mathbb{IR}_{+}^{*n}.$$

The following proposition lists a few important properties of interval posynomials and, by extension, of real posynomials as well:

Proposition 4.4.1. *Let* $\underline{\overline{p}}\,(\mathbf{x})$ *be a multivariate posynomial with interval coefficients. Then:*

1) **Non-Negativity:** $\underline{\overline{p}}\,(\mathbf{x}) \in \mathbb{IR}_{+}^{*}$, *for any* $\mathbf{x} \in \mathbb{R}_{+}^{*n}$.

2) **Partial Derivatives:** *The vector partial derivative* $\underline{\overline{p}}^{(\mathbf{k})}(\mathbf{x})$ *is a posynomial for any integer vector* $\mathbf{k}$.

3) **Strict Monotonicity:** *If* $\mathbf{x}$ *and* $\mathbf{y}$ *are two non-negative real vectors such that* $\mathbf{x}_i \leq \mathbf{y}_i$ *for all integers* i, *and there exists an integer* j *such that* $\mathbf{x}_j < \mathbf{y}_j$ *then:*

$$\underline{p}\,(\mathbf{x}) < \underline{p}\,(\mathbf{y}),\ \textit{and}$$

$$\overline{p}\,(\mathbf{x}) < \overline{p}\,(\mathbf{y}).$$

4) **Range Inclusion Function:** *The range of a posynomial* $\underline{\overline{p}}\,(\mathbf{x})$ *over an interval* $\underline{\overline{\mathbf{x}}} \geq 0$ *is:*

$$\mathcal{R}\left(\underline{\overline{p}}\right)\left(\underline{\overline{\mathbf{x}}}\right) = \left[\underline{p}\left(\underline{\mathbf{x}}\right), \overline{p}\left(\overline{\mathbf{x}}\right)\right].$$

5) **Taylor Form Inclusion Function:** *For any non-negative interval vector* $\underline{\overline{\mathbf{x}}}$*, and any real vector* $\mathbf{c}$ *such that* $0 \leq c_i \leq \underline{x}_i$ *for all i, the Taylor Form inclusion function at* $\mathbf{c}$ *is exactly equal (ignoring roundoff errors) to the range of the posynomial* $\underline{\overline{p}}$*:*

$$Taylor\left(\underline{\overline{p}}, \mathbf{c}\right)\left(\underline{\overline{\mathbf{x}}}\right) = \mathcal{R}\left(\underline{\overline{p}}\right)\left(\underline{\overline{\mathbf{x}}}\right).$$

Proofs are straightforward and follow directly from the definitions.

It is important to remember that the properties of posynomials listed above will not hold if they are evaluated outside of $\mathbb{IR}_{+}^{*n}$.

4.5 The Posynomial Decomposition of a Polynomial

The posynomial decomposition is a surprisingly powerful tool for the analysis of various inclusion properties of Taylor Forms. We make use of it extensively in the proofs of the theorems that follow in the next sections.

Let $\underline{\overline{p}}\,(\mathbf{x})$ be a multivariate polynomial with interval coefficients, and let $\mathbf{c}$ be the expansion point for the exact Taylor Form expansion below:

$$Taylor\left(\underline{\overline{p}}, \mathbf{c}\right)(\mathbf{x}) \quad = \quad \sum_{\mathbf{i} \in I} \frac{\overline{\underline{p}}^{(\mathbf{i})}(\mathbf{c})}{\mathbf{i}!}(\mathbf{x}-\mathbf{c})^{\mathbf{i}}.$$

We restrict the values of $\mathbf{x}$ such that $(\mathbf{x}-\mathbf{c}) \in \mathbb{R}_{+}^{*n}$, i.e. $x_i \geq c_i$ for all i. We define the *P-posynomial*

and the *N-posynomial* of the Taylor Form $Taylor\left(p,\mathbf{c}\right)\left(\mathbf{x}\right)$ as follows:

$$\text{P-posynomial:}\quad Taylor_{\oplus}\left(\underline{\overline{p}},\mathbf{c}\right)\left(\mathbf{x}\right) = \sum_{\substack{\mathbf{i}\in I \\ \underline{\overline{p}}^{(\mathbf{i})}(\mathbf{c})\cap\mathbb{R}_{+}^{*}\neq\emptyset}} \frac{\left(\underline{\overline{p}}^{(\mathbf{i})}(\mathbf{c})\right)_{\oplus}}{\mathbf{i}!}\left(\mathbf{x}-\mathbf{c}\right)^{\mathbf{i}},$$

$$\text{N-posynomial:}\quad Taylor_{\ominus}\left(\underline{\overline{p}},\mathbf{c}\right)\left(\mathbf{x}\right) = \sum_{\substack{\mathbf{i}\in I \\ \underline{\overline{p}}^{(\mathbf{i})}(\mathbf{c})\cap\mathbb{R}_{-}^{*}\neq\emptyset}} \frac{\left(\underline{\overline{p}}^{(\mathbf{i})}(\mathbf{c})\right)_{\ominus}}{\mathbf{i}!}\left(\mathbf{x}-\mathbf{c}\right)^{\mathbf{i}},$$

where $\left(\underline{\overline{p}}^{(\mathbf{i})}(\mathbf{c})\right)_{\oplus}$ and $\left(\underline{\overline{p}}^{(\mathbf{i})}(\mathbf{c})\right)_{\ominus}$ are the non-negative and non-positive parts of the SC decomposition of $\underline{\overline{p}}^{(\mathbf{i})}(\mathbf{c})$ respectively.

We will also use the following shorthand notation:

$$\text{P-posynomial:}\quad \underline{\overline{p}}_{\oplus|\mathbf{c}}\left(\mathbf{x}\right) = Taylor_{\oplus}\left(\underline{\overline{p}},\mathbf{c}\right)\left(\mathbf{x}\right)$$

$$\text{N-posynomial:}\quad \underline{\overline{p}}_{\ominus|\mathbf{c}}\left(\mathbf{x}\right) = Taylor_{\ominus}\left(\underline{\overline{p}},\mathbf{c}\right)\left(\mathbf{x}\right).$$

Furthermore, if $\mathbf{c}=\mathbf{0}$ we write:

$$\text{P-posynomial:}\quad \underline{\overline{p}}_{\oplus}\left(\mathbf{x}\right) = MacLaurin_{\oplus}\left(\underline{\overline{p}}\right)\left(\mathbf{x}\right)$$

$$\text{N-posynomial:}\quad \underline{\overline{p}}_{\ominus}\left(\mathbf{x}\right) = MacLaurin_{\ominus}\left(\underline{\overline{p}}\right)\left(\mathbf{x}\right),$$

where the posynomials $MacLaurin_{\oplus}\left(\underline{\overline{p}}\right)\left(\mathbf{x}\right)$ and $MacLaurin_{\ominus}\left(\underline{\overline{p}}\right)\left(\mathbf{x}\right)$ are defined the natural way.

With the above definitions, the interval polynomial $\underline{\overline{p}}$ can be written as the difference of the P and N-posynomials in $\left(\mathbf{x}-\mathbf{c}\right)$. This is called the *posynomial decomposition* of $\underline{\overline{p}}$ at $\mathbf{c}$:

$$\text{Posynomial decomposition: } \underline{\overline{p}}\left(\mathbf{x}\right) = \underline{\overline{p}}_{\oplus|\mathbf{c}}\left(\mathbf{x}\right) - \underline{\overline{p}}_{\ominus|\mathbf{c}}\left(\mathbf{x}\right).$$

If p is a polynomial with real coefficients then the P-posynomial is comprised of the terms

with positive coefficients while the N-posynomial is comprised of all the terms with negative coefficients in absolute value. For example, let us consider the following bivariate polynomial and its Taylor Form at the point $(1,-1)^T$:

$$\begin{aligned} \mathit{Taylor}\left(p,(1,-1)^T\right)(\mathbf{x}) &= 3(\mathbf{x}_1-1)^2(\mathbf{x}_2+1)^2-(\mathbf{x}_1-1)(\mathbf{x}_2+1)+3 \\ &= \left[3(\mathbf{x}_1-1)^2(\mathbf{x}_2+1)^2+3\right]-\left[(\mathbf{x}_1-1)(\mathbf{x}_2+1)\right]. \end{aligned}$$

Then the P-posynomial is $3(\mathbf{x}_1-1)^2(\mathbf{x}_2+1)^2+3$ and the N-posynomial is $(\mathbf{x}_1-1)(\mathbf{x}_2+1)$:

$$\begin{aligned} p_{\oplus|(1,-1)^T}(\mathbf{x}) &= 3(\mathbf{x}_1-1)^2(\mathbf{x}_2+1)^2+3 \\ p_{\ominus|(1,-1)^T}(\mathbf{x}) &= (\mathbf{x}_1-1)(\mathbf{x}_2+1). \end{aligned}$$

4.6 Taylor Form Excess Width is Due to One Interval Minus Operation

In this section we use the posynomial decomposition to show that the excess width of any natural or Taylor Form interval extension—ignoring roundoff errors—is due to one interval minus operation.

Proposition 4.6.1. *Let* $\mathit{Taylor}\left(\underline{\overline{p}},\mathbf{c}\right)$ *be a Taylor Form of the interval polynomial* $\underline{\overline{p}}$*, and let* $\underline{\overline{\mathbf{x}}}$ *be an interval vector such that* $\underline{\overline{\mathbf{x}}}-\mathbf{c}\geq 0$*. Let* $\mathcal{T}\left(\underline{\overline{p}},\mathbf{c}\right)$ *be the interval extension of the Taylor Form above. Then:*

$$\mathcal{T}\left(\underline{\overline{p}},\mathbf{c}\right)\left(\underline{\overline{\mathbf{x}}}\right)=\underline{\overline{p}}_{\oplus|\mathbf{c}}\left(\underline{\overline{\mathbf{x}}}\right)-\underline{\overline{p}}_{\ominus|\mathbf{c}}\left(\underline{\overline{\mathbf{x}}}\right),$$

and:

$$\mathrm{w}\left(\mathcal{T}\left(\underline{\overline{p}},\mathbf{c}\right)\left(\underline{\overline{\mathbf{x}}}\right)\right)=\mathrm{w}\left(\underline{\overline{p}}_{\oplus|\mathbf{c}}\left(\underline{\overline{\mathbf{x}}}\right)\right)+\mathrm{w}\left(\underline{\overline{p}}_{\ominus|\mathbf{c}}\left(\underline{\overline{\mathbf{x}}}\right)\right),$$

The proof is a simple application of the posynomial decomposition.

4.7 Reduction to the Non-Negative Quadrant

The rest of the chapter uses posynomial decompositions to analyze the inclusion properties of natural and Taylor Form interval extensions of polynomials. For simplicity, all arguments are restricted to non-negative intervals. In this section we show that all other cases can be reduced to the non-negative quadrant without loss of generality.

First we prove that the range over a zero-straddling interval vector $\overline{\underline{\mathbf{x}}}$ computed by the natural extension of a polynomial p is equal to the union of the ranges computed by the same natural extension over the sign-coherent interval vectors $\overline{\underline{\mathbf{x}}}_{\oplus}$ and $-\overline{\underline{\mathbf{x}}}_{\ominus}$.

Proposition 4.7.1. *Let $\overline{\underline{p}}$ be a multivariate polynomial with interval coefficients and $\overline{\underline{\mathbf{x}}} = \overline{\underline{\mathbf{x}}}_{\oplus} - \overline{\underline{\mathbf{x}}}_{\ominus}$ be a zero-straddling interval vector. Then:*

$$\sum_{\mathbf{i}\leq\mathbf{n}} \overline{\underline{a}}_{\mathbf{i}}\, \overline{\underline{\mathbf{x}}}^{\,\mathbf{i}} = \left(\sum_{\mathbf{i}\leq\mathbf{n}} \overline{\underline{a}}_{\mathbf{i}}\, \overline{\underline{\mathbf{x}}}_{\oplus}^{\,\mathbf{i}}\right) \cup \left(\sum_{\mathbf{i}\leq\mathbf{n}} \overline{\underline{a}}_{\mathbf{i}} (-\overline{\underline{\mathbf{x}}}_{\ominus})^{\mathbf{i}}\right).$$

Proof:

For any $a, b > 0$ and positive integer k the following holds:

$$\begin{aligned} [-a,b]^k &= ([-a,0]\cup[0,b])^k \\ &= [-a,0]^k \cup [0,b]^k. \end{aligned}$$

Also, for any intervals $\overline{\underline{x}}$, $\overline{\underline{y}}$, and $\overline{\underline{z}}$ the following is true:

$$\begin{aligned} \left(\overline{\underline{x}} \cup \overline{\underline{y}}\right) + \overline{\underline{z}} &= (\overline{\underline{x}} + \overline{\underline{z}}) \cup \left(\overline{\underline{y}} + \overline{\underline{z}}\right), \\ \left(\overline{\underline{x}} \cup \overline{\underline{y}}\right) \overline{\underline{z}} &= (\overline{\underline{x}}\ \overline{\underline{z}}) \cup \left(\overline{\underline{y}}\ \overline{\underline{z}}\right). \end{aligned}$$

Therefore, if $0 \in \underline{\overline{\mathbf{x}}}_k$ for all integers k:

$$\begin{aligned}
\sum_{\mathbf{i}\leq\mathbf{n}} \underline{\overline{a}}_{\mathbf{i}} \underline{\overline{\mathbf{x}}}^{\mathbf{i}} &= \sum_{\mathbf{i}\leq\mathbf{n}} \underline{\overline{a}}_{\mathbf{i}} \left(\underline{\overline{\mathbf{x}}}^{\mathbf{i}}_{\oplus} \cup (-\underline{\overline{\mathbf{x}}}_{\ominus})^{\mathbf{i}} \right) \\
&= \sum_{\mathbf{i}\leq\mathbf{n}} \left(\underline{\overline{a}}_{\mathbf{i}} \underline{\overline{\mathbf{x}}}^{\mathbf{i}}_{\oplus} \cup (-\underline{\overline{a}}_{\mathbf{i}} \underline{\overline{\mathbf{x}}}_{\ominus})^{\mathbf{i}} \right) \\
&= \left(\sum_{\mathbf{i}\leq\mathbf{n}} \underline{\overline{a}}_{\mathbf{i}} \underline{\overline{\mathbf{x}}}^{\mathbf{i}}_{\oplus} \right) \cup \left(\sum_{\mathbf{i}\leq\mathbf{n}} \underline{\overline{a}}_{\mathbf{i}} (-\underline{\overline{\mathbf{x}}}_{\ominus})^{\mathbf{i}} \right).
\end{aligned}$$

□

Finally we show that the range enclosure computed over an interval vector $\underline{\overline{\mathbf{x}}}$ with mixed sign-coherent components is the same as the range enclosure of a properly transformed Taylor Form over the non-negative interval vector $|\underline{\overline{\mathbf{x}}}|$.

Proposition 4.7.2. *Let $\underline{\overline{p}}$ be a multivariate polynomial with interval coefficients, and $\underline{\overline{\mathbf{x}}}$ be a component-wise sign coherent interval vector. Let J be the set of indices of non-negative components of $\underline{\overline{\mathbf{x}}}$, i.e. $\underline{\overline{\mathbf{x}}}_{j\in J} \in \mathbb{IR}^*_+$, and $\underline{\overline{\mathbf{x}}}_{j\notin J} \in \mathbb{IR}^*_-$. We construct a new interval vector $\underline{\overline{\mathbf{y}}} = |\underline{\overline{\mathbf{x}}}|$, i.e.:*

$$\underline{\overline{\mathbf{y}}}_k = \begin{cases} \underline{\overline{\mathbf{x}}}_k, \text{ if } k \in J, \\ -\underline{\overline{\mathbf{x}}}_k, \text{ if } k \notin J. \end{cases}$$

Let $\underline{\overline{q}}$ be a polynomial such that:

$$\underline{\overline{q}}\,(\mathbf{y}) = \underline{\overline{p}}\,(\mathbf{x}),$$

where:

$$\mathbf{y}_k = \begin{cases} \mathbf{x}_k, \text{ if } k \in J, \\ -\mathbf{x}_k, \text{ if } k \notin J. \end{cases}$$

Then:

1) $\mathcal{N}\left(\underline{\overline{q}}\right)\left(\underline{\overline{\mathbf{y}}}\right) = \mathcal{N}\left(\underline{\overline{p}}\right)\left(\underline{\overline{\mathbf{x}}}\right)$,

2) $\mathcal{T}\left(\underline{\overline{q}}, \mathrm{c}\left(\underline{\overline{\mathbf{y}}}\right)\right)\left(\underline{\overline{\mathbf{y}}}\right) = \mathcal{T}\left(\underline{\overline{p}}, \mathrm{c}\left(\underline{\overline{\mathbf{x}}}\right)\right)\left(\underline{\overline{\mathbf{x}}}\right)$.

The proof is straightforward.

4.8 Corner Taylor Forms With Interval Coefficients

In this section we introduce the Corner Taylor Form inclusion function, $\mathcal{T}_c\left(\underline{\overline{p}}\right)$, for a multivariate polynomial $\underline{\overline{p}}$ with interval coefficients. We prove that the Corner Taylor Form has less excess width than the corresponding natural extension and that it is inclusion isotonic.

Let $\underline{\overline{p}} : \mathbb{R}^n \to \mathbb{IR}$ be a multivariate polynomial with interval coefficients, with expression:

$$\underline{\overline{p}}\,(\mathbf{x}) = \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \underline{\overline{a}}_{\mathbf{i}}\mathbf{x}^{\mathbf{i}}.$$

We define the *Corner Taylor Form* inclusion function $\mathcal{T}_c\left(\underline{\overline{p}}\right)(\underline{\overline{\mathbf{x}}})$ to be the interval extension of the Taylor expansion of $\underline{\overline{p}}$ at the corner $\mathbf{x}_0 = \mathrm{c}(\underline{\overline{\mathbf{x}}})$ of the interval vector $\underline{\overline{\mathbf{x}}}$:

$$\text{Corner Taylor Form: } \mathcal{T}_c\left(\underline{\overline{p}}\right)(\underline{\overline{\mathbf{x}}}) = \mathit{Taylor}\left(\underline{\overline{p}}\,,\mathrm{c}(\underline{\overline{\mathbf{x}}})\right)(\underline{\overline{\mathbf{x}}}).$$

4.8.1 The Corner Taylor Form Always Has Less Excess Width Than the Natural Extension

Next, we prove the main result of this section.

Theorem 4.8.1. *Let* $\underline{\overline{p}} : \mathbb{R}^n \to \mathbb{IR}$ *be a multivariate polynomial with interval coefficients, and* $\underline{\overline{\mathbf{x}}} \in \mathbb{IR}^n$ *be an interval vector. Then*

$$\mathcal{T}_c\left(\underline{\overline{p}}\right)(\underline{\overline{\mathbf{x}}}) \subseteq \mathcal{N}\left(\underline{\overline{p}}\right)(\underline{\overline{\mathbf{x}}}).$$

Proof:

Assume that the interval vector $\underline{\overline{\mathbf{x}}}$ is non-negative, i.e. $\underline{\overline{\mathbf{x}}} \in \mathbb{IR}^{*n}_{+}$. The other cases are reduced to this one by virtue of the results in Section 4.7.

The products $\overline{\underline{\mathbf{x}}}^{\,\mathbf{i}}$ are non-negative intervals, for any multi-index vector $\mathbf{i}$ in J. By applying Proposition 4.3.1 term by term and using the associativity of interval addition we can write the following derivation:

$$\begin{aligned}
\mathcal{N}\left(\overline{\underline{p}}\right)\left(\overline{\underline{\mathbf{x}}}\right) &= MacLaurin\left(\overline{\underline{p}}\right)\left(\overline{\underline{\mathbf{x}}}\right) \\
&= \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \overline{\underline{a}}_{\mathbf{i}}\,\overline{\underline{\mathbf{x}}}^{\,\mathbf{i}} \\
&= \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \left(\overline{\underline{a}}_{\mathbf{i}\oplus} - \overline{\underline{a}}_{\mathbf{i}\ominus}\right)\overline{\underline{\mathbf{x}}}^{\,\mathbf{i}} \\
&= \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \left(\overline{\underline{\mathbf{x}}}^{\,\mathbf{i}}\,\overline{\underline{a}}_{\mathbf{i}\oplus} - \overline{\underline{\mathbf{x}}}^{\,\mathbf{i}}\,\overline{\underline{a}}_{\mathbf{i}\ominus}\right) \\
&= \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \overline{\underline{a}}_{\mathbf{i}\oplus}\,\overline{\underline{\mathbf{x}}}^{\,\mathbf{i}} - \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \overline{\underline{a}}_{\mathbf{i}\ominus}\,\overline{\underline{\mathbf{x}}}^{\,\mathbf{i}}.
\end{aligned}$$

The above expression is the posynomial decomposition of $\overline{\underline{p}}$:

$$\begin{aligned}
\overline{\underline{p}}_{\,\oplus}(\mathbf{x}) &= MacLaurin_{\oplus}\left(\overline{\underline{p}}\right)(\mathbf{x}) = \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \overline{\underline{a}}_{\mathbf{i}\oplus}\mathbf{x}^{\mathbf{i}}, \\
\overline{\underline{p}}_{\,\ominus}(\mathbf{x}) &= MacLaurin_{\ominus}\left(\overline{\underline{p}}\right)(\mathbf{x}) = \sum_{\mathbf{i}\in J\subset\mathbb{N}^n} \overline{\underline{a}}_{\mathbf{i}\ominus}\mathbf{x}^{\mathbf{i}}.
\end{aligned}$$

Using Proposition 4.6.1 we can write:

$$\begin{aligned}
\mathcal{N}\left(\overline{\underline{p}}\right)\left(\overline{\underline{\mathbf{x}}}\right) &= \overline{\underline{p}}_{\,\oplus}\left(\overline{\underline{\mathbf{x}}}\right) - \overline{\underline{p}}_{\,\ominus}\left(\overline{\underline{\mathbf{x}}}\right) \\
&= Taylor\left(\overline{\underline{p}}_{\,\oplus}, \underline{\mathbf{x}}\right)\left(\overline{\underline{\mathbf{x}}}\right) - Taylor\left(\overline{\underline{p}}_{\,\ominus}, \underline{\mathbf{x}}\right)\left(\overline{\underline{\mathbf{x}}}\right) \\
&= \sum_{\mathbf{i}\leq\mathbf{n}} \frac{1}{\mathbf{i}!}\left(\overline{\underline{p}}^{\,(\mathbf{i})}_{\oplus}\left(\underline{\mathbf{x}}\right)\left(\overline{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\right)^{\mathbf{i}} - \overline{\underline{p}}^{\,(\mathbf{i})}_{\ominus}\left(\underline{\mathbf{x}}\right)\left(\overline{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\right)^{\mathbf{i}}\right).
\end{aligned}$$

The coefficients $\overline{\underline{p}}^{\,(\mathbf{i})}_{\oplus}\left(\underline{\mathbf{x}}\right)$ and $\overline{\underline{p}}^{\,(\mathbf{i})}_{\ominus}\left(\underline{\mathbf{x}}\right)$ in the expression above are strictly positive (properties derivatives of interval posynomials). Distributivity doesn't hold even though $\left(\overline{\underline{\mathbf{x}}} - \underline{\mathbf{x}}\right)^{\mathbf{i}}$ is non-

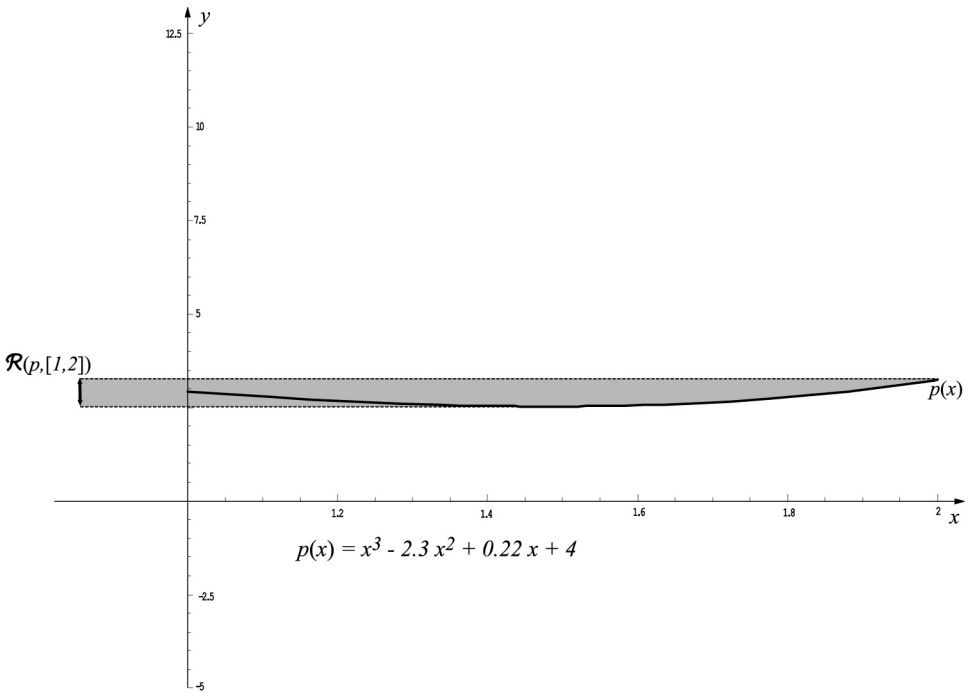


Figure 4.1: The range of the polynomial $p(x)$ on the interval $[1,2]$ is $\mathcal{R}(p)([1,2])$. The exact value of the range on an interval can be difficult to compute.

negative. Using subdistributivity we conclude:

$$\begin{aligned}
\mathcal{N}\left(\underline{\overline{p}}\right)(\underline{\overline{\mathbf{x}}}) &= \sum_{\mathbf{i}\leq\mathbf{n}} \frac{1}{\mathbf{i}!}\left(\underline{\overline{p}}_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}})(\underline{\overline{\mathbf{x}}}-\underline{\mathbf{x}})^{\mathbf{i}} - \underline{\overline{p}}_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}})(\underline{\overline{\mathbf{x}}}-\underline{\mathbf{x}})^{\mathbf{i}}\right) \\
&\supseteq \sum_{\mathbf{i}\leq\mathbf{n}} \frac{1}{\mathbf{i}!}\left(\underline{\overline{p}}_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}}) - \underline{\overline{p}}_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}})\right)(\underline{\overline{\mathbf{x}}}-\underline{\mathbf{x}})^{\mathbf{i}} \\
&= \mathit{Taylor}\left(\underline{\overline{p}}, \underline{\mathbf{x}}\right)(\underline{\overline{\mathbf{x}}}) \\
&= \mathcal{T}_c\left(\underline{\overline{p}}\right)(\underline{\overline{\mathbf{x}}}).
\end{aligned}$$

Equality holds if and only if one of the posynomials $\underline{\overline{p}}_{\oplus}$ or $\underline{\overline{p}}_{\ominus}$ is identically zero. □

4.8.2 Isotonicity of the Corner Taylor Form

Next we prove that the Corner Taylor Form is interval isotonic.

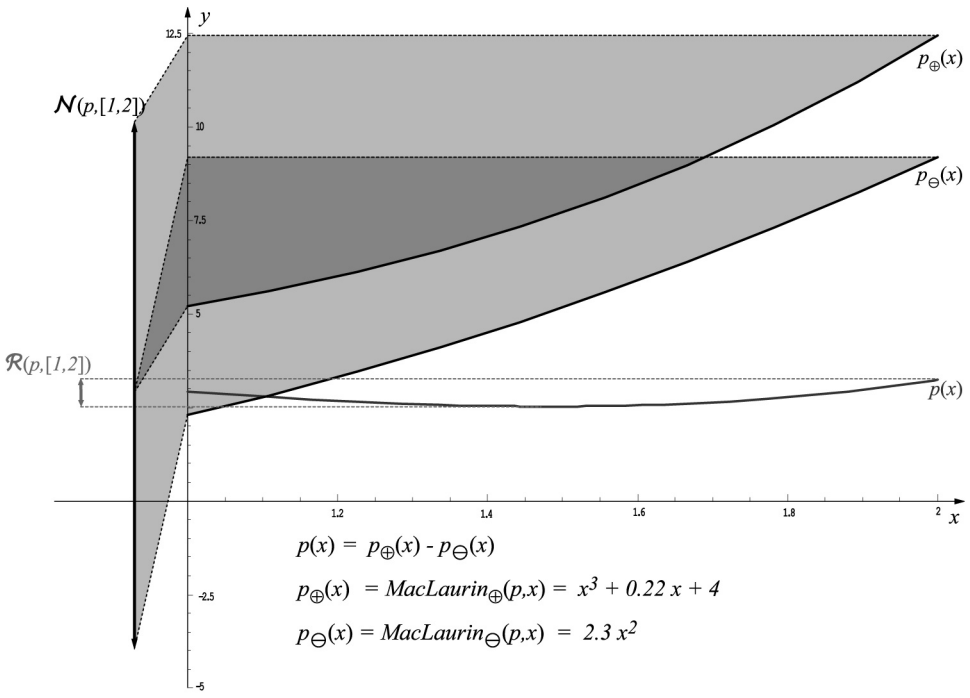


Figure 4.2: The natural extension $\mathcal{N}(p)$ greatly overestimates the range. Proposition 4.6.1 proves that the width of the computed bound is equal to the sum of widths of the ranges of $p_{\oplus}$ and $p_{\ominus}$, the P and N-posynomials of the MacLaurin form of p.

Theorem 4.8.2. *Let* $\underline{\overline{p}}\,(\mathbf{x})$ *be a multivariate polynomial with interval coefficients, and let* $\underline{\overline{\mathbf{x}}} \subseteq \underline{\overline{\mathbf{y}}}$ *be two nested sign-coherent interval vectors. Then:*

$$\mathcal{T}_c\left(\underline{\overline{p}}\right)\left(\underline{\overline{\mathbf{x}}}\right) \subseteq \mathcal{T}_c\left(\underline{\overline{p}}\right)\left(\underline{\overline{\mathbf{y}}}\right).$$

Proof:

Case 1. Assume that $\underline{\overline{\mathbf{y}}}$ is a non-negative interval vector such that:

$$\underline{\mathbf{x}} = \mathrm{c}\left(\underline{\overline{\mathbf{x}}}\right) = \mathrm{c}\left(\underline{\overline{\mathbf{y}}}\right) = \underline{\mathbf{y}}.$$

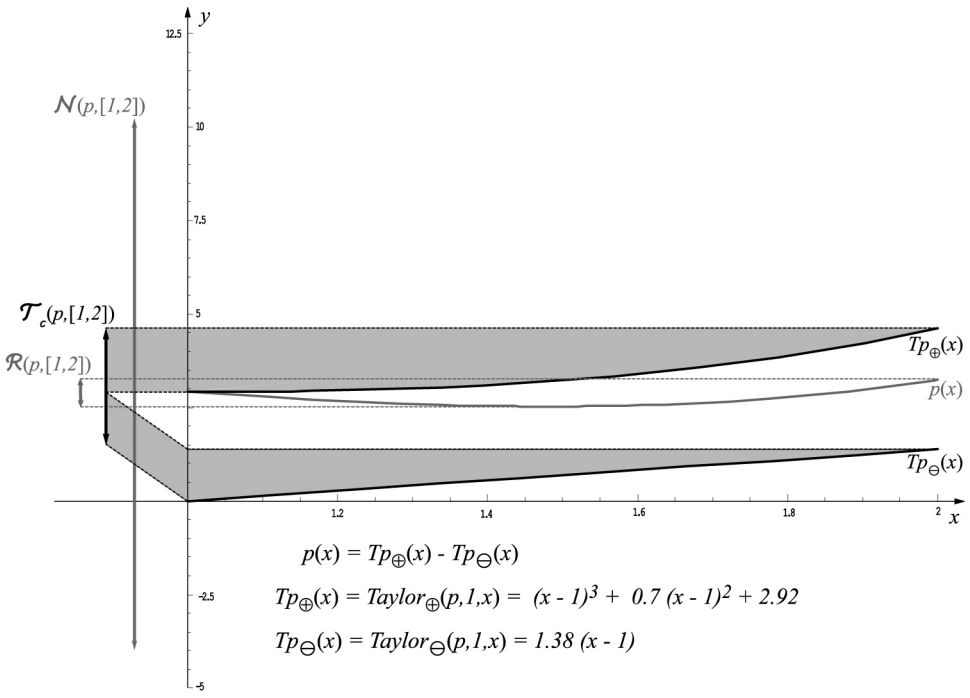


Figure 4.3: The Corner Taylor Form, $\mathcal{T}_c(p)$, produces improved bounds as shown in Theorem 4.9.1. The width of the Corner Taylor Form is equal to the sum of the widths of the ranges of $Tp_\oplus$ and $Tp_\ominus$, the P and N-posynomials of the Taylor Form of $p(x)$ expanded at $x = 1$. Note that $Tp_\oplus$ and $Tp_\ominus$ have smaller ranges than the P and N-posynomials, $p_\oplus$ and $p_\ominus$, of the MacLaurin form (see figure 4.2). Therefore, the Corner Taylor Form inclusion function, $\mathcal{T}_c(p)([1,2])$, produces bounds with significantly less excess width when compared to the natural inclusion function $\mathcal{N}(p)([1,2])$.

The posynomial decomposition of the Corner Taylor Form evaluated on the interval vector $\underline{\overline{\mathbf{y}}}$ is:

$$\mathcal{T}_c\left(\underline{\overline{p}}\right)\left(\underline{\overline{\mathbf{y}}}\right) = Taylor_\oplus\left(\underline{\overline{p}}\,,\underline{\mathbf{y}}\right)\left(\underline{\overline{\mathbf{y}}}\right) - Taylor_\ominus\left(\underline{\overline{p}}\,,\underline{\mathbf{y}}\right)\left(\underline{\overline{\mathbf{y}}}\right).$$

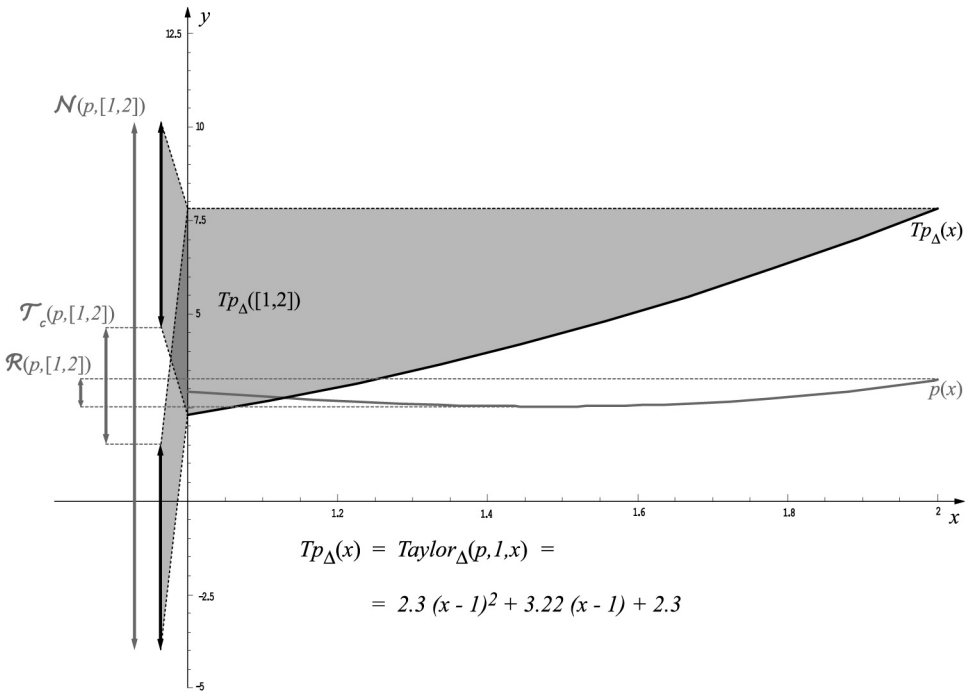


Figure 4.4: The magnitude of the improvement $\mathrm{w}(\mathcal{N}(p)) - \mathrm{w}(\mathcal{T}_c(p))$ can be computed in closed form. It is twice the width of the range of the posynomial $Taylor_\Delta(p,1)(x)$.

Because posynomials are strictly monotonic we can write:

$$\begin{aligned}
\underline{\mathcal{T}_c\left(\overline{p}\right)\left(\overline{\mathbf{y}}\right)} &= \underline{Taylor_\oplus\left(\overline{p},\mathbf{y}\right)\left(\underline{\mathbf{y}}\right)} - \overline{Taylor_\ominus\left(\overline{p},\mathbf{y}\right)\left(\overline{\mathbf{y}}\right)} \\
&= \underline{Taylor_\oplus\left(\overline{p},\mathbf{y}\right)\left(\underline{\mathbf{x}}\right)} - \overline{Taylor_\ominus\left(\overline{p},\mathbf{y}\right)\left(\overline{\mathbf{y}}\right)} \\
&\leq \underline{Taylor_\oplus\left(\overline{p},\mathbf{y}\right)\left(\underline{\mathbf{x}}\right)} - \overline{Taylor_\ominus\left(\overline{p},\mathbf{y}\right)\left(\overline{\mathbf{x}}\right)} \\
&= \underline{Taylor_\oplus\left(\overline{p},\mathbf{x}\right)\left(\underline{\mathbf{x}}\right)} - \overline{Taylor_\ominus\left(\overline{p},\mathbf{x}\right)\left(\overline{\mathbf{x}}\right)} \\
&= \underline{\mathcal{T}_c\left(\overline{p}\right)\left(\overline{\mathbf{x}}\right)}\,.
\end{aligned}$$

The inequality for the upper bounds is derived in similar fashion:

$$\overline{\mathcal{T}_c\left(\overline{p}\right)\left(\overline{\mathbf{y}}\right)} \quad \geq \quad \overline{\mathcal{T}_c\left(\overline{p}\right)\left(\overline{\mathbf{x}}\right)}\,,$$

and this case is proved.

Case 2. Assume once again that $\overline{\underline{\mathbf{y}}}$ is non-negative and $\underline{\mathbf{x}} > \underline{\mathbf{y}}$ so that $\overline{\underline{\mathbf{x}}}$ is a proper subset of $\overline{\underline{\mathbf{y}}}$. We make the substitution $\mathbf{t} = \mathbf{x} - \underline{\mathbf{y}}$ and define the equivalent polynomial $\overline{\underline{q}}(\mathbf{t}) = \overline{\underline{p}}(\mathbf{x})$. Then:

$$\begin{aligned}
\mathcal{T}_c\left(\overline{\underline{p}}\right)\left(\overline{\underline{\mathbf{x}}}\right) &= \mathcal{T}_c\left(\overline{\underline{q}}\right)\left(\overline{\underline{\mathbf{x}}} - \underline{\mathbf{y}}\right) \\
&\subseteq \mathcal{N}\left(\overline{\underline{q}}\right)\left(\overline{\underline{\mathbf{x}}} - \underline{\mathbf{y}}\right) \\
&= \mathit{Taylor}\left(\overline{\underline{p}}, \underline{\mathbf{y}}\right)\left(\overline{\underline{\mathbf{x}}}\right) \\
&\subset \mathit{Taylor}\left(\overline{\underline{p}}, \underline{\mathbf{y}}\right)\left(\overline{\underline{\mathbf{y}}}\right) \\
&= \mathcal{T}_c\left(\overline{\underline{p}}\right)\left(\overline{\underline{\mathbf{y}}}\right),
\end{aligned}$$

and Case 2 is proved. All the other cases reduce to the two cases above. □

4.9 Corner Taylor Forms With Real Coefficients

In this section we restrict our attention to Corner Taylor Forms with real coefficients. For this case we prove some important extended properties.

The first property allows us measure how much tighter the bounds computed by the Corner Taylor Form are when respect to those computed using natural extensions.

The second provides an algorithm for computing the local inclusion order of a Corner Taylor Form as a function of the input interval. This local inclusion order can vary for a given polynomial across its range. It is at least quadratic and can be as high at the degree of the polynomial. In some cases the Corner Taylor Form evaluates the range exactly (up to rounding error). We show how these cases are easily detected.

Of course, all the theorems for Corner Taylor Forms with interval coefficients apply to this case as well.

4.9.1 The Magnitude of the Improvement Over Natural Extensions

Consider a multivariate polynomial $p : \mathbb{R}^n \to \mathbb{R}$ with real coefficients, whose expression is given by:

$$p(\mathbf{x}) = \sum_{\mathbf{i} \in J \subset \mathbb{N}^n} a_{\mathbf{i}} \mathbf{x}^{\mathbf{i}},\ a_{\mathbf{i}} \in \mathbb{R}.$$

The *Corner Taylor Form* inclusion function $\mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}})$ is defined as:

$$\mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}}) = Taylor(p, \mathrm{c}(\overline{\underline{\mathbf{x}}}))(\overline{\underline{\mathbf{x}}}). \tag{4.1}$$

We now prove a stronger version of Theorem 4.8.1:

Theorem 4.9.1. *Let $\overline{\underline{\mathbf{x}}} \in \mathbb{IR}^n$ be a vector of sign-coherent intervals. Then there exists a multivariate posynomial with positive real-valued coefficients:*

$$p_{\Delta|\mathrm{c}(\overline{\underline{\mathbf{x}}})}(\mathbf{x})$$

such that

$$\mathcal{N}(p)(\overline{\underline{\mathbf{x}}}) = \mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}}) + [-1,1]\ p_{\Delta|\mathrm{c}(\overline{\underline{\mathbf{x}}})}(\overline{\underline{\mathbf{x}}}).$$

Proof:

Assume that $\overline{\underline{\mathbf{x}}} \in \mathbb{IR}^{*n}_{+}$ is a vector of non-negative intervals so that $\mathrm{c}(\overline{\underline{\mathbf{x}}}) = \underline{\mathbf{x}}$. The rest of the cases are reduced to this one.

The posynomial decomposition of the MacLaurin form of p is:

$$MacLaurin(p)(\overline{\underline{\mathbf{x}}}) \quad = \quad p_{\oplus}(\mathbf{x}) - p_{\ominus}(\mathbf{x}).$$

The corresponding interval evaluation rules are:

$$\begin{aligned}\mathcal{N}(p)(\underline{\overline{\mathbf{x}}}) &= p_{\ominus}(\underline{\overline{\mathbf{x}}}) - p_{\oplus}(\underline{\overline{\mathbf{x}}}) \\ &= Taylor(p_{\oplus}, \underline{\mathbf{x}})(\underline{\overline{\mathbf{x}}}) - Taylor(p_{\ominus}, \underline{\mathbf{x}})(\underline{\overline{\mathbf{x}}}).\end{aligned}$$

where the properties of posynomials were used to derive the last two steps, see Proposition 4.4.1.

A similar derivation applied to the Corner Taylor Form of p yields the following:

$$\begin{aligned}Taylor(p, \underline{\mathbf{x}})(\mathbf{x}) &= Taylor_{\oplus}(p, \underline{\mathbf{x}})(\mathbf{x}) - Taylor_{\ominus}(p, \underline{\mathbf{x}})(\mathbf{x}), \\ \mathcal{T}_c(p)(\underline{\overline{\mathbf{x}}}) &= Taylor_{\oplus}(p, \underline{\mathbf{x}})(\underline{\overline{\mathbf{x}}}) - Taylor_{\ominus}(p, \underline{\mathbf{x}})(\underline{\overline{\mathbf{x}}}).\end{aligned}$$

Let:

$$\begin{aligned}q_1(\mathbf{x}) &= Taylor(p_{\oplus}, \underline{\mathbf{x}})(\mathbf{x}) - Taylor_{\oplus}(p, \underline{\mathbf{x}})(\mathbf{x}), \\ q_2(\mathbf{x}) &= Taylor(p_{\ominus}, \underline{\mathbf{x}})(\mathbf{x}) - Taylor_{\ominus}(p, \underline{\mathbf{x}})(\mathbf{x}).\end{aligned}$$

We show that q_1 and q_2 are equal posynomials.

Consider the $\mathbf{i}$th coefficient of q_1:

$$\begin{aligned}a_{\mathbf{i}} &= \frac{1}{\mathbf{i}!}\left(p_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}}) - \max\left(0,\ p^{(\mathbf{i})}(\underline{\mathbf{x}})\right)\right) \\ &= \frac{1}{\mathbf{i}!}\left(p_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}}) - \max\left(0,\ p_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}}) - p_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}})\right)\right) \\ &= \frac{1}{\mathbf{i}!}\min\left(p_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}}),\ p_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}})\right).\end{aligned}$$

Similarly the $\mathbf{i}$th coefficient of q_2 is:

$$\begin{aligned}b_{\mathbf{i}} &= \frac{1}{\mathbf{i}!}\left(p_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}}) - \max\left(0,\ -p^{(\mathbf{i})}(\underline{\mathbf{x}})\right)\right) \\ &= \frac{1}{\mathbf{i}!}\left(p_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}}) - \max\left(0,\ p_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}}) - p_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}})\right)\right) \\ &= \frac{1}{\mathbf{i}!}\min\left(p_{\ominus}^{(\mathbf{i})}(\underline{\mathbf{x}}),\ p_{\oplus}^{(\mathbf{i})}(\underline{\mathbf{x}})\right).\end{aligned}$$

Therefore $a_{\mathbf{i}} = b_{\mathbf{i}} \geq 0$. We define the posynomial $p_{\Delta|\,\underline{\mathbf{x}}}\,(\mathbf{x}) = q_1(\mathbf{x}) = q_2(\mathbf{x})$. Then:

$$\begin{aligned} \mathit{Taylor}(p_{\oplus}, \underline{\mathbf{x}})(\mathbf{x}) &= p_{\Delta|\,\underline{\mathbf{x}}}\,(\mathbf{x}) + \mathit{Taylor}_{\oplus}(p, \underline{\mathbf{x}})(\mathbf{x}) \\ \mathit{Taylor}(p_{\ominus}, \underline{\mathbf{x}})(\mathbf{x}) &= p_{\Delta|\,\underline{\mathbf{x}}}\,(\mathbf{x}) + \mathit{Taylor}_{\ominus}(p, \underline{\mathbf{x}})(\mathbf{x}). \end{aligned}$$

Finally:

$$\begin{aligned} \mathcal{N}(p)(\overline{\underline{\mathbf{x}}}) &= \mathit{Taylor}(p_{\oplus}, \mathrm{c}(\overline{\underline{\mathbf{x}}}))(\overline{\underline{\mathbf{x}}}) - \mathit{Taylor}(p_{\ominus}, \mathrm{c}(\overline{\underline{\mathbf{x}}}))(\overline{\underline{\mathbf{x}}}) \\ &= p_{\Delta|\,\underline{\mathbf{x}}}\,(\overline{\underline{\mathbf{x}}}) + \mathit{Taylor}_{\oplus}(p, \mathrm{c}(\overline{\underline{\mathbf{x}}}))(\overline{\underline{\mathbf{x}}}) \\ &\quad - \left(p_{\Delta|\,\underline{\mathbf{x}}}\,(\overline{\underline{\mathbf{x}}}) + \mathit{Taylor}_{\ominus}(p, \mathrm{c}(\overline{\underline{\mathbf{x}}}))(\overline{\underline{\mathbf{x}}})\right) \\ &= \mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}}) + p_{\Delta|\,\underline{\mathbf{x}}}\,(\overline{\underline{\mathbf{x}}}) - p_{\Delta|\,\underline{\mathbf{x}}}\,(\overline{\underline{\mathbf{x}}}) \\ &= \mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}}) + [-1, 1]\, p_{\Delta|\,\underline{\mathbf{x}}}\,(\overline{\underline{\mathbf{x}}}), \end{aligned}$$

and the theorem is proved. □

Theorem 4.9.1 provides a closed form expression for the posynomial $p_{\Delta|\mathbf{c}}(\mathbf{x})$:

$$p_{\Delta|\mathbf{c}}(\mathbf{x}) = \sum_{\mathbf{i} \in \mathbb{N}^n} \frac{\min\left(p_{\oplus}^{(\mathbf{i})}(\mathbf{c}),\ p_{\ominus}^{(\mathbf{i})}(\mathbf{c})\right)}{\mathbf{i}!} (\mathbf{x} - \mathbf{c})^{\mathbf{i}}.$$

It follows directly from Theorem 4.9.1 that the magnitude of the reduction in excess width of the Corner Taylor Form over the natural extension, when $\overline{\underline{\mathbf{x}}}$ is a non-negative interval vector, is given by:

$$\boxed{\mathrm{w}(\mathcal{N}(p)(\overline{\underline{\mathbf{x}}})) - \mathrm{w}(\mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}})) = 2\,\mathrm{w}\left(p_{\Delta|\mathrm{c}(\overline{\underline{\mathbf{x}}})}(\overline{\underline{\mathbf{x}}})\right).}$$

The above expression could be used in algorithms, such as root finding or optimization, to decide when the extra cost of evaluating the Corner Taylor Form is worth it or not.

4.9.2 Convergence Properties

Next we investigate the convergence properties of the excess width of Corner Taylor Forms with real coefficients.

Theorem 4.9.2. *Let $p(\mathbf{x})$ be a multivariate polynomial. The excess width of the Corner Taylor Form $\mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}})$ has quadratic order or better when the width of $\overline{\underline{\mathbf{x}}}$ goes to zero.*

Proof:

Once again we prove the result for the case $\overline{\underline{\mathbf{x}}} \subset \mathbb{IR}_+^{*n}$ so that $c(\overline{\underline{\mathbf{x}}}) = \underline{\overline{\mathbf{x}}}$.

We begin by noticing that the excess width of an inclusion function $\mathcal{I}(p)$ has two components, the upper excess width and the lower excess width, each of which is the minimum of the difference between all possible values of the function $p(\mathbf{y})$ and the upper and lower bounds of $\mathcal{I}(p)$ respectively, as follows:

$$\begin{aligned} \Delta W(\mathcal{I}(p)(\overline{\underline{\mathbf{x}}})) \quad &= \quad \min_{\mathbf{y}\in\overline{\underline{\mathbf{x}}}}\{p(\mathbf{y}) - \inf(\mathcal{I}(p)(\overline{\underline{\mathbf{x}}}))\} \\ &\quad\quad + \min_{\mathbf{y}\in\overline{\underline{\mathbf{x}}}}\{\sup(\mathcal{I}(p)(\overline{\underline{\mathbf{x}}})) - p(\mathbf{y})\}. \end{aligned}$$

Consider the upper excess width of the Corner Taylor Form:

$$\begin{aligned} &\min_{\mathbf{y}\in\overline{\underline{\mathbf{x}}}} \quad \{p(\mathbf{y}) - \inf(\mathcal{T}_c(p)(\overline{\underline{\mathbf{x}}}))\} \\ &= \quad \min_{\mathbf{y}\in\overline{\underline{\mathbf{x}}}} \left\{ \left(p_{\oplus|\underline{\mathbf{x}}}(\mathbf{y}) - p_{\ominus|\underline{\mathbf{x}}}(\mathbf{y})\right) - \left(p_{\oplus|\underline{\mathbf{x}}}(\underline{\mathbf{x}}) - p_{\ominus|\underline{\mathbf{x}}}(\overline{\mathbf{x}})\right) \right\} \\ &= \quad \min_{\mathbf{y}\in\overline{\underline{\mathbf{x}}}} \left\{ \left(p_{\oplus|\underline{\mathbf{x}}}(\mathbf{y}) - p_{\oplus|\underline{\mathbf{x}}}(\underline{\mathbf{x}})\right) - \left(p_{\ominus|\underline{\mathbf{x}}}(\mathbf{y}) - p_{\ominus|\underline{\mathbf{x}}}(\overline{\mathbf{x}})\right) \right\}. \end{aligned}$$

Since the minimum of the above expression is taken over all values of $\mathbf{y} \in \overline{\underline{\mathbf{x}}}$, its value has to be smaller than or equal to the value of the expression at any particular values. We choose two such

values, $\mathbf{y} = \underline{\mathbf{x}}$ and $\mathbf{y} = \overline{\mathbf{x}}$:

$$
\begin{aligned}
\min_{\mathbf{y}\in\,\underline{\overline{\mathbf{x}}}} \quad & \{p(\mathbf{y}) - \inf(\mathcal{T}_c(p)(\underline{\overline{\mathbf{x}}}))\} \quad = \\
= \quad & \min_{\mathbf{y}\in\,\underline{\overline{\mathbf{x}}}} \left\{ \left(p_{\oplus|\,\underline{\mathbf{x}}}(\mathbf{y}) - p_{\oplus|\,\underline{\mathbf{x}}}(\underline{\mathbf{x}})\right) - \left(p_{\ominus|\,\underline{\mathbf{x}}}(\mathbf{y}) - p_{\ominus|\,\underline{\mathbf{x}}}(\overline{\mathbf{x}})\right) \right\} \\
\leq \quad & \begin{cases} \left(p_{\oplus|\,\underline{\mathbf{x}}}(\underline{\mathbf{x}}) - p_{\oplus|\,\underline{\mathbf{x}}}(\underline{\mathbf{x}})\right) - \left(p_{\ominus|\,\underline{\mathbf{x}}}(\underline{\mathbf{x}}) - p_{\ominus|\,\underline{\mathbf{x}}}(\overline{\mathbf{x}})\right) \\ \left(p_{\oplus|\,\underline{\mathbf{x}}}(\overline{\mathbf{x}}) - p_{\oplus|\,\underline{\mathbf{x}}}(\underline{\mathbf{x}})\right) - \left(p_{\ominus|\,\underline{\mathbf{x}}}(\overline{\mathbf{x}}) - p_{\ominus|\,\underline{\mathbf{x}}}(\overline{\mathbf{x}})\right) \end{cases} \\
= \quad & \begin{cases} p_{\ominus|\,\underline{\mathbf{x}}}(\overline{\mathbf{x}}) - p_{\ominus|\,\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \\ p_{\oplus|\,\underline{\mathbf{x}}}(\overline{\mathbf{x}}) - p_{\oplus|\,\underline{\mathbf{x}}}(\underline{\mathbf{x}}) \end{cases} \\
= \quad & \begin{cases} \mathrm{w}\left(p_{\ominus|\,\underline{\mathbf{x}}}(\underline{\overline{\mathbf{x}}})\right) \\ \mathrm{w}\left(p_{\oplus|\,\underline{\mathbf{x}}}(\underline{\overline{\mathbf{x}}})\right) \end{cases}
\end{aligned}
$$

By definition, the terms of the posynomials $p_{\ominus|\,\underline{\mathbf{x}}}$ and $p_{\ominus|\,\underline{\mathbf{x}}}$ are mutually exclusive, i.e. only one of them has nonzero coefficient of multi-index $\mathbf{i}$. Therefore, the lowest degree term of one of these two posynomials must be of quadratic (or higher) degree, so the upper excess width of the Corner Taylor Form must also have quadratic (or higher) order of convergence as well. The fact that the posynomials $p_{\ominus|\,\underline{\mathbf{x}}}$ and $p_{\ominus|\,\underline{\mathbf{x}}}$ are expressed in terms of $(\overline{\mathbf{x}} - \underline{\mathbf{x}})$—an interval vector containing zero—implies that the convergence rate depends only on the width of the interval vector $\underline{\overline{\mathbf{x}}}$ and is independent of its magnitude.

The proof for the lower excess width follows along the same lines. □

4.10 Examples and Results

Consider the polynomial $p(x)$ and the following expressions associated with it:

$$\begin{aligned} \mathit{MacLaurin}(p)(x) &= x^2 - x, \\ \mathit{Taylor}(p, 1.5)(x) &= (x-1.5)^2 + 2(x-1.5) + 0.75, \\ \mathit{Taylor}(p, 1)(x) &= (x-1)^2 + (x-1). \end{aligned}$$

It is easy to check that the range of p over the interval $[1,2]$ is:

$$\mathcal{R}(p)([1,2]) = [0,2].$$

The interval extensions corresponding to each of the above expressions produce the different results when evaluated over the same interval. The natural extension, corresponding to the first expression above, evaluates as follows:

$$\begin{aligned} \mathcal{N}(p)([1,2]) &= \mathit{MacLaurin}(p)([1,2]) \\ &= [1,2]^2 - [1,2] \\ &= [1,4] - [1,2] \\ &= [-1,3] \end{aligned}$$

The Midpoint Taylor Form, corresponding to the second expression above, evaluates as follows:

$$\begin{aligned}
\mathcal{T}_m(p)([1,2]) &= \mathit{Taylor}(p,1.5)([1,2]) \\
&= ([1,2]-1.5)^2 + 2([1,2]-1,5) + 0.75 \\
&= [-0.5,0.5]^2 + 2[-0.5,0.5] + 0.75 \\
&= [0,0.25] + [-1,1] + 0.75 \\
&= [-0.25,2]
\end{aligned}$$

Finally, the Corner Taylor Form, corresponding to the third expression above, evaluates as follows:

$$\begin{aligned}
\mathcal{T}_c(p)([1,2]) &= \mathit{Taylor}(p,1)([1,2]) \\
&= ([1,2]-1)^2 + ([1,2]-1) \\
&= [0,1]^2 + [0,1] \\
&= [0,1] + [0,1] \\
&= [0,2]
\end{aligned}$$

In this example, the Midpoint Taylor Form produced slightly worse bounds that the corresponding Corner Taylor Form. Both inclusion functions produced significantly better bounds than those produced by the natural extension. However, this is not a consistent behavior. While the Corner Taylor Form is guaranteed to always have less excess width than the natural extension, the same is not true of the Midpoint Taylor Form. To illustrate this behavior consider the

fourth order polynomial $q(x)$ and the following expressions:

$$\begin{aligned}
\mathit{MacLaurin}\,(q)\,(x) &= x^4 - x^3 - 12x^2 - 4x + 16 \\
\mathit{Taylor}\,(q, 0.1)\,(x) &= (x-0.1)^4 - 0.6(x-0.1)^3 - 12.24(x-0.1)^2 \\
&\quad -6.426(x-0.1) + 15.4791 \\
\mathit{Taylor}\,(q, 10.1)\,(x) &= (x-10.1)^4 + 39.4(x-10.1)^3 + 569.76(x-10.1)^2 \\
&\quad +3568.774(x-10.1) + 8127.2191
\end{aligned}$$

The range of q on the interval $[0.1, 20.1]$ is:

$$\mathcal{R}\,(q)\,([0.1, 20.1]) = [-50.1944, 150191]$$

Evaluation of the associated inclusion functions on the same interval produces the following results:

$$\begin{aligned}
\mathcal{N}\,(q)\,([0.1, 20.1]) &= \mathit{MacLaurin}\,(q)\,([0.1, 20.1]) = [-13033.1, 163240] \\
\mathcal{T}_m\,(q)\,([0.1, 20.1]) &= \mathit{Taylor}\,(q, 10.1)\,([0.1, 20.1]) = [-66960.5, 150191] \\
\mathcal{T}_c\,(q)\,([0.1, 20.1]) &= \mathit{Taylor}\,(q, 0.1)\,([0.1, 20.1]) = [-9809.04, 160015]
\end{aligned}$$

Comparing the widths of the ranges computed by the various inclusion functions we see that the Corner Taylor Form returns the tightest bounds:

$$\begin{aligned}
\mathrm{w}\,(\mathcal{R}\,(q)\,([0.1, 20.1])) &= \mathrm{w}\,([-50.1944, 150191]) = 150241 \\
\mathrm{w}\,(\mathcal{N}\,(q)\,([0.1, 20.1])) &= \mathrm{w}\,([-13033.1, 163240]) = 176273 \\
\mathrm{w}\,(\mathcal{T}_m\,(q)\,([0.1, 20.1])) &= \mathrm{w}\,([-66960.5, 150191]) = 217151 \\
\mathrm{w}\,(\mathcal{T}_c\,(q)\,([0.1, 20.1])) &= \mathrm{w}\,([-9809.04, 160015]) = 169825
\end{aligned}$$

Notice that in this case the Midpoint Taylor Form produced bounds that are significantly worse than both the Corner Taylor Form and the natural extension.

Evaluation on the interval $[5,15]$ returns similar results. The relevant expressions are:

$$\begin{aligned} \mathit{Taylor}(q,5)(x) &= (x-5)^4 - 19(x-5)^3 - 123(x-5)^2 - 301(x-5) + 196 \\ \mathit{Taylor}(q,10)(x) &= (x-10)^4 + 39(x-10)^3 + 558(x-10)^2 + 3456(x-10) + 7776 \end{aligned}$$

$$\mathcal{R}(q)([5,15]) = [196, 44506],$$

and the computed bounds are:

$$\begin{aligned} \mathcal{N}(q)([5,15]) &= \mathit{MacLaurin}(q)([5,15]) = [-5494, 50196] \\ \mathcal{T}_m(q)([5,15]) &= \mathit{Taylor}(q,10)([5,15]) = [-14379, 44506] \\ \mathcal{T}_c(q)([5,15]) &= \mathit{Taylor}(q,5)([5,15]) = [196, 44506] \end{aligned}$$

$$\begin{aligned} \mathrm{w}(\mathcal{R}(q)([5,15])) &= \mathrm{w}([196, 44506]) = 44310 \\ \mathrm{w}(\mathcal{N}(q)([5,15])) &= \mathrm{w}([-5494, 50196]) = 55690 \\ \mathrm{w}(\mathcal{T}_m(q)([5,15])) &= \mathrm{w}([-14379, 44506]) = 58885 \\ \mathrm{w}(\mathcal{T}_c(q)([5,15])) &= \mathrm{w}([196, 44506]) = 44310 \end{aligned}$$

If the above evaluations were performed as part of a root searching algorithm we notice that only the Corner Taylor Form correctly shows that no roots of q lie in the interval $[5,15]$, while the midpoint form and the natural extension require additional subdivisions to reach to the same conclusion.

The same results are obtained with multivariate polynomials as illustrated below. Let $f^*(x,y)$ be the following fifth order MacLaurin power series:

$$f^*(x,y) = MacLaurin^{<5,5>}\left(\cos 2x \sin 3y + \sin 3x \cos 2y - \cos 2x \cos 3y + \sin 3x \sin 2y\right).$$

The polynomial f^* has the following MacLaurin expression:

$$\begin{aligned}
MacLaurin(f^*)(x,y) = & -1 + 3x + 2x^2 \\
& - 4.5x^3 - 0.666x^4 + 2.025x^5 \\
& + 3y + 6xy - 6x^2y \\
& - 9x^3y + 2x^4y + 4.05x^5y \\
& + 4.5y^2 - 6xy^2 - 9x^2y^2 \\
& + 9x^3y^2 + 3x^4y^2 - 4.05x^5y^2 \\
& - 4.5y^3 - 4xy^3 + 9x^2y^3 \\
& + 6x^3y^3 - 3x^4y^3 - 2.7x^5y^3 \\
& - 3.375y^4 + 2xy^4 + 6.75x^2y^4 \\
& - 3x^3y^4 - 2.25x^4y^4 + 1.35x^5y^4 \\
& + 2.025y^5 + 0.8xy^5 - 4.05x^2y^5 \\
& - 1.2x^3y^5 + 1.35x^4y^5 + 0.54x^5y^5
\end{aligned}$$

If we evaluate the associated inclusion functions on the interval $[2,4] \times [2,4]$ we get:

$$\begin{aligned}
\mathcal{N}(f^*)([2,4],[2,4]) &= \mathit{MacLaurin}(f^*)([2,4],[2,4]) &= \left[-636480.04, 1.377 \times 10^6\right] \\
\mathcal{T}_m(f^*)([2,4],[2,4]) &= \mathit{Taylor}(f^*,3)([2,4],[2,4]) &= [-364056, 494616] \\
\mathcal{T}_c(f^*)([2,4],[2,4]) &= \mathit{Taylor}(f^*,2)([2,4],[2,4]) &= [184.493, 718598]
\end{aligned}$$

$$\begin{aligned}
\mathrm{w}(\mathcal{N}(f^*)([2,4],[2,4])) &= \mathrm{w}\left(\left[-636480.04, 1.377 \times 10^6\right]\right) &= 2.014 \times 10^6 \\
\mathrm{w}(\mathcal{T}_m(f^*)([2,4],[2,4])) &= \mathrm{w}([-364056, 494616]) &= 858673 \\
\mathrm{w}(\mathcal{T}_c(f^*)([2,4],[2,4])) &= \mathrm{w}([184.493, 718598]) &= 718414
\end{aligned}$$

The comparative efficiency of the three inclusion functions is illustrated by the solution sets computed with a binary search algorithm. Note that the algorithm converges very slowly when the natural inclusion function is used, see figure 4.5. The Corner Taylor Form inclusion function allows the algorithm to discard fairly large regions early in the search process, see figure 4.7, even when compared to the similarly converging Midpoint Taylor Form, see figure 4.6.

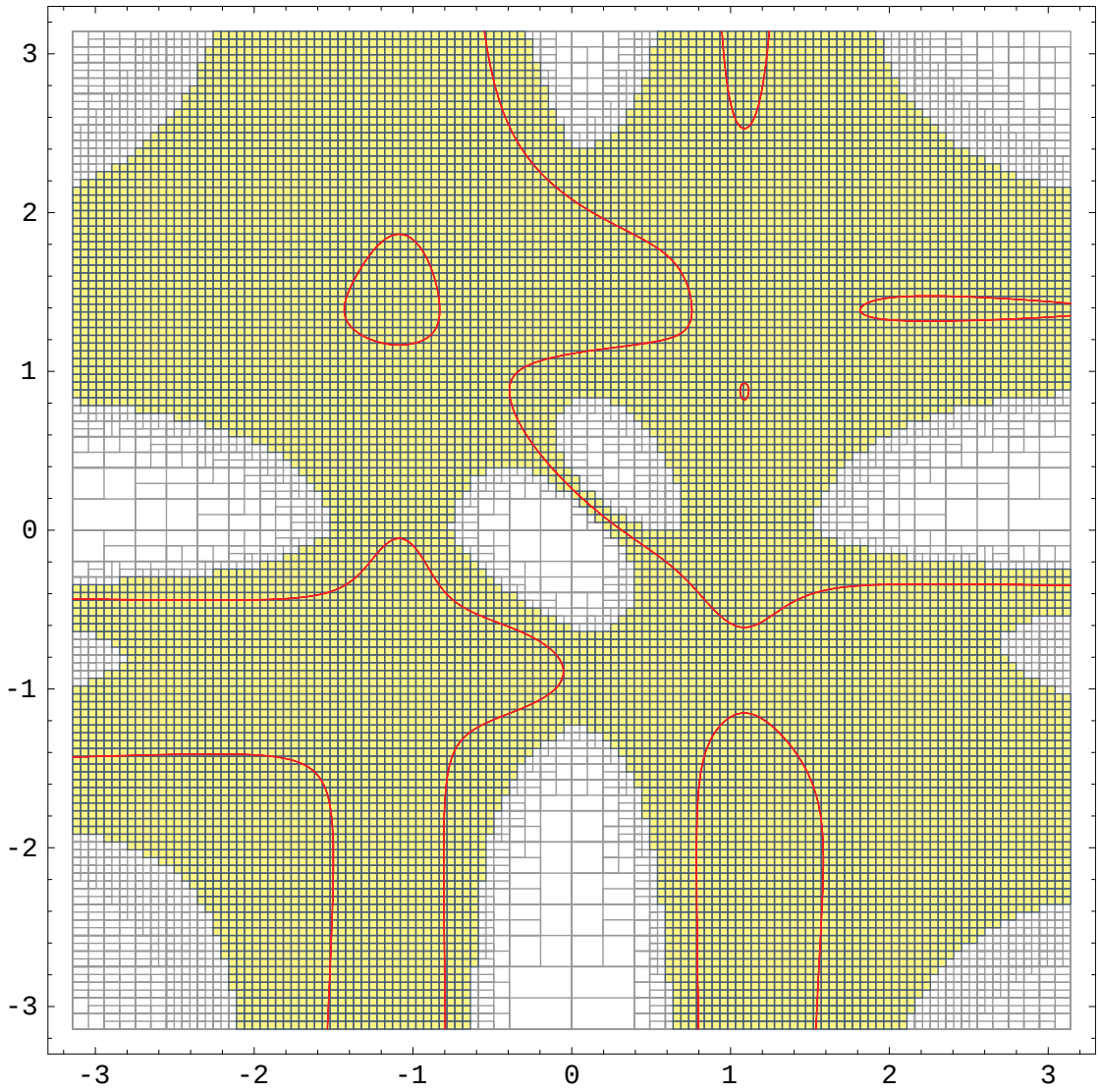


Figure 4.5: Regions in gray indicate possible roots of the fifth order Taylor multinomial $MacLaurin^{<5,5>}(\cos 2x \sin 3y + \sin 3x \cos 2y - \cos 2x \cos 3y + \sin 3x \sin 2y) = 0$ computed using the natural inclusion function. The search process converges very slowly due to the large excess width of the natural inclusion function, retaining many superfluous solution regions. The quality factor is only 0.0571.

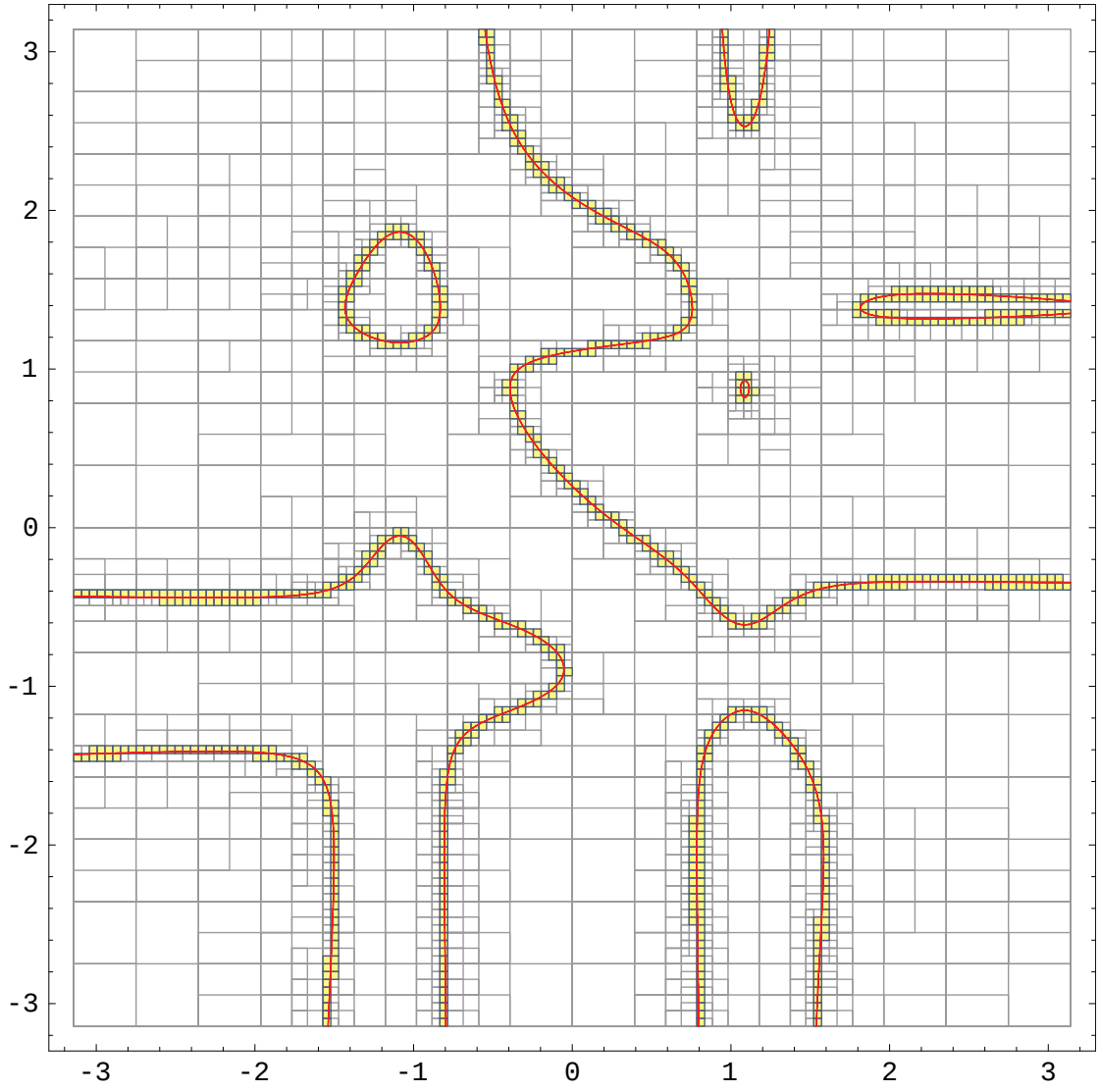


Figure 4.6: Roots of the same multinomial as in figure 4.5 computed using the Midpoint Taylor Form inclusion function. The search process converges quickly once the size of the regions fall under a certain threshold. There is still a fair amount of work being done to eliminate regions where there are no roots as shown, for example, in the upper right corner. Several subdivisions are needed before the region can be declared root free. The quality factor is 0.8997. Compare with figure 4.7.

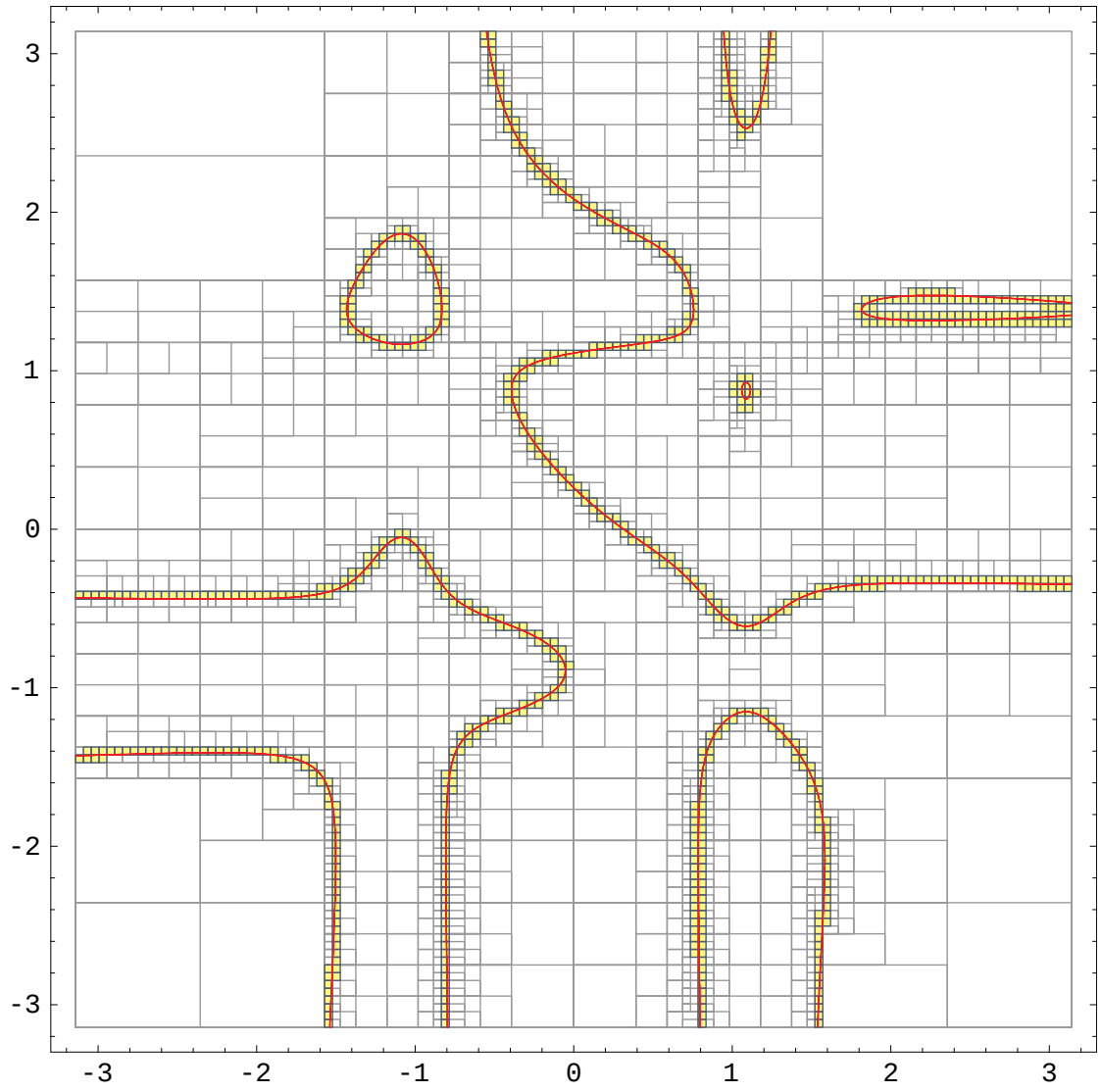


Figure 4.7: Roots of the same multinomial as in figures 4.5 and 4.6 computed using the Corner Taylor Form inclusion function, $\Upsilon_c(f)$. Notice that the Corner Taylor Form is more accurate than the Midpoint Taylor Form for large input regions. The region in the upper right corner is declared root free very early in the subdivision process. The quality factor is 0.8821. Compare with figure 4.6.

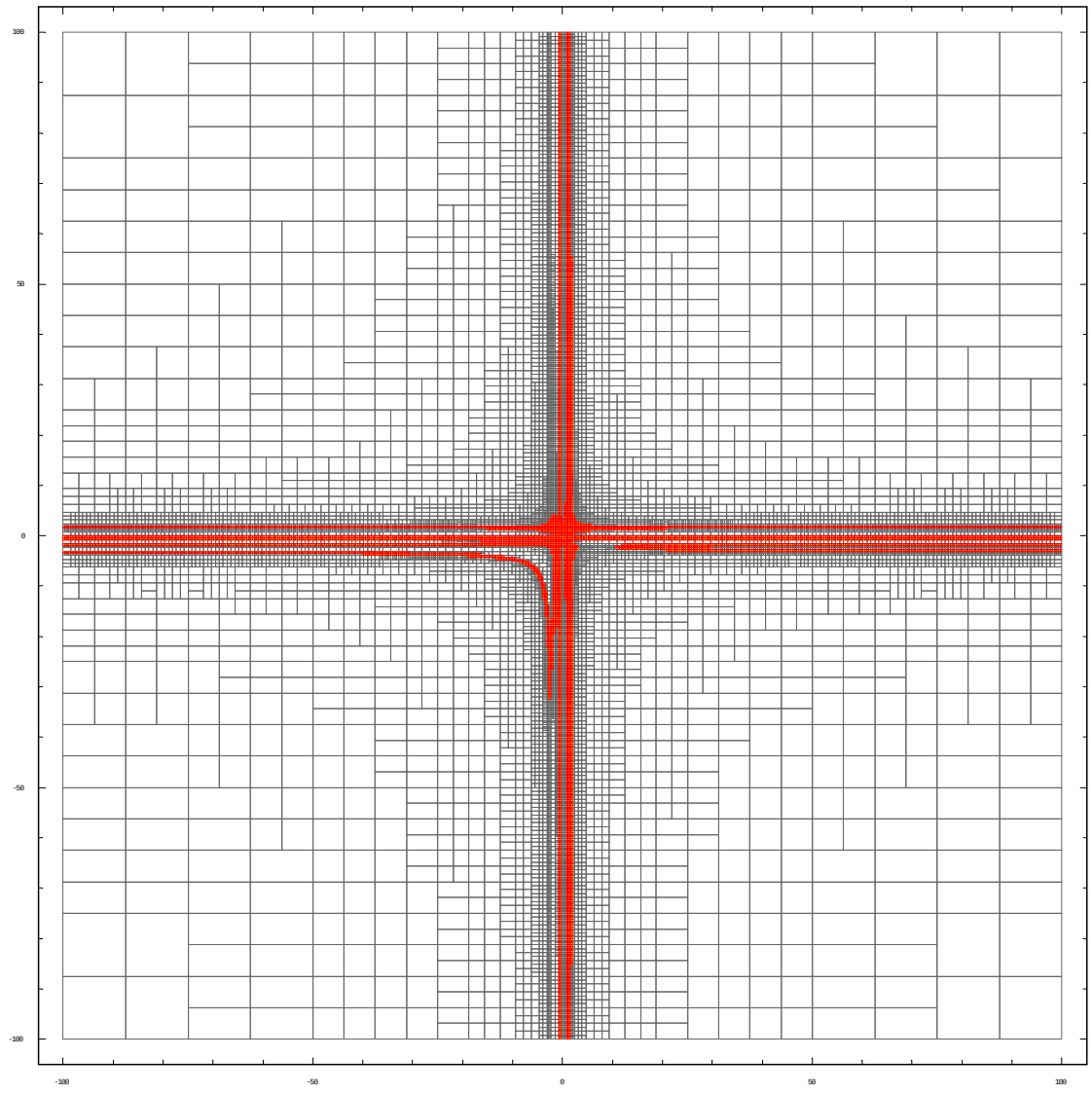


Figure 4.8: Plot of the solution regions produced by divide and conquer with Midpoint Taylor Forms on larger domains. The domain is $[-100,100]^2$. The algorithm found 4,950 solution regions in 36,899 iterations which took 514.765 seconds (Mathematica 5 time). Note that there is a considerable amount of work being done away from the solutions. Compare with figure 4.9.

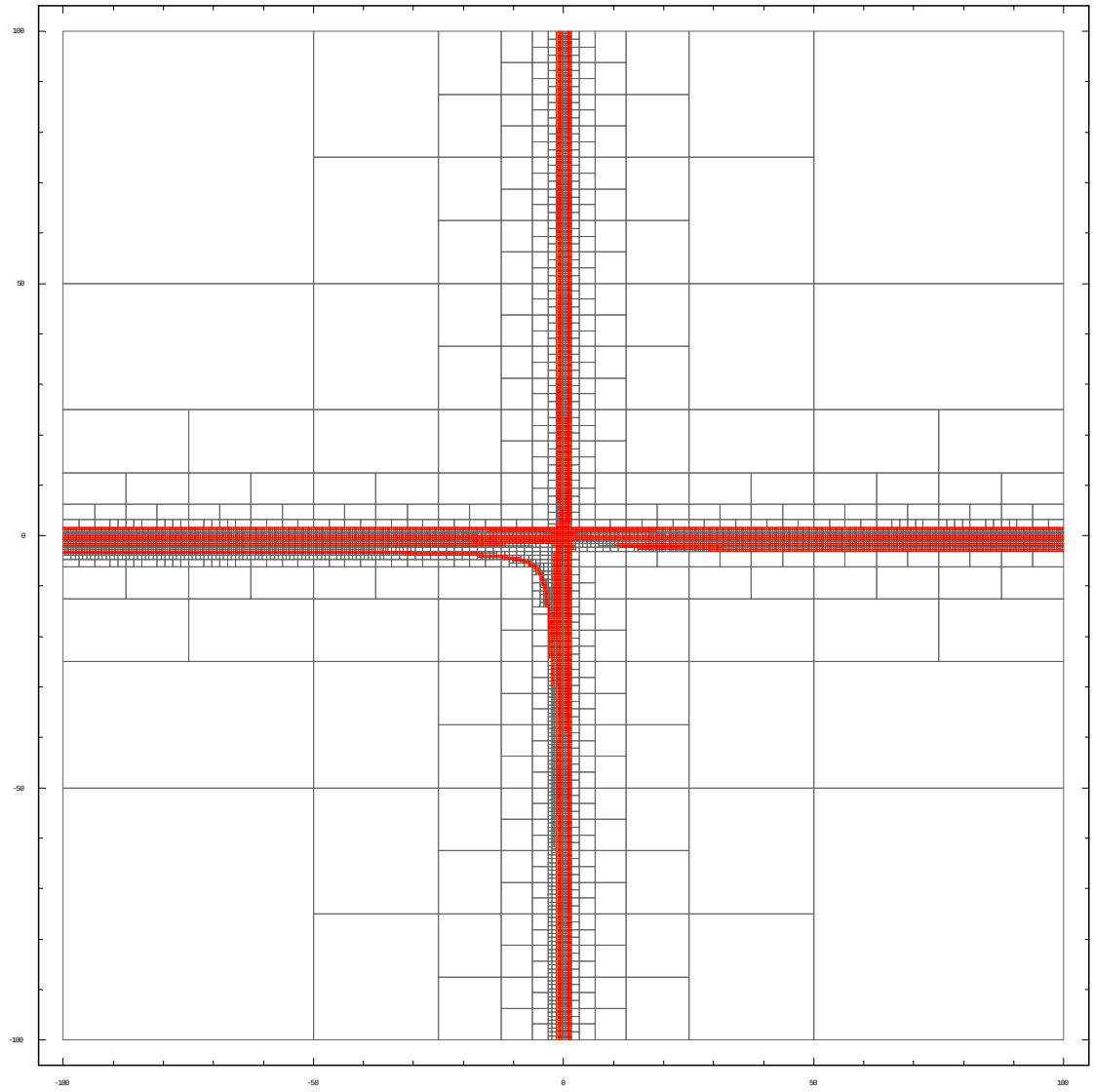


Figure 4.9: Plot of the solution regions produced by divide and conquer with Corner Taylor Forms on larger domains. The domain is $[-100,100]^2$. The algorithm found 5,008 solution regions in 26,115 iterations which took 319.266 seconds (Mathematica 5 time). Away from where solutions are the algorithm eliminates regions at the maximum speed possible with binary subdivision. Compare with figure 4.8.

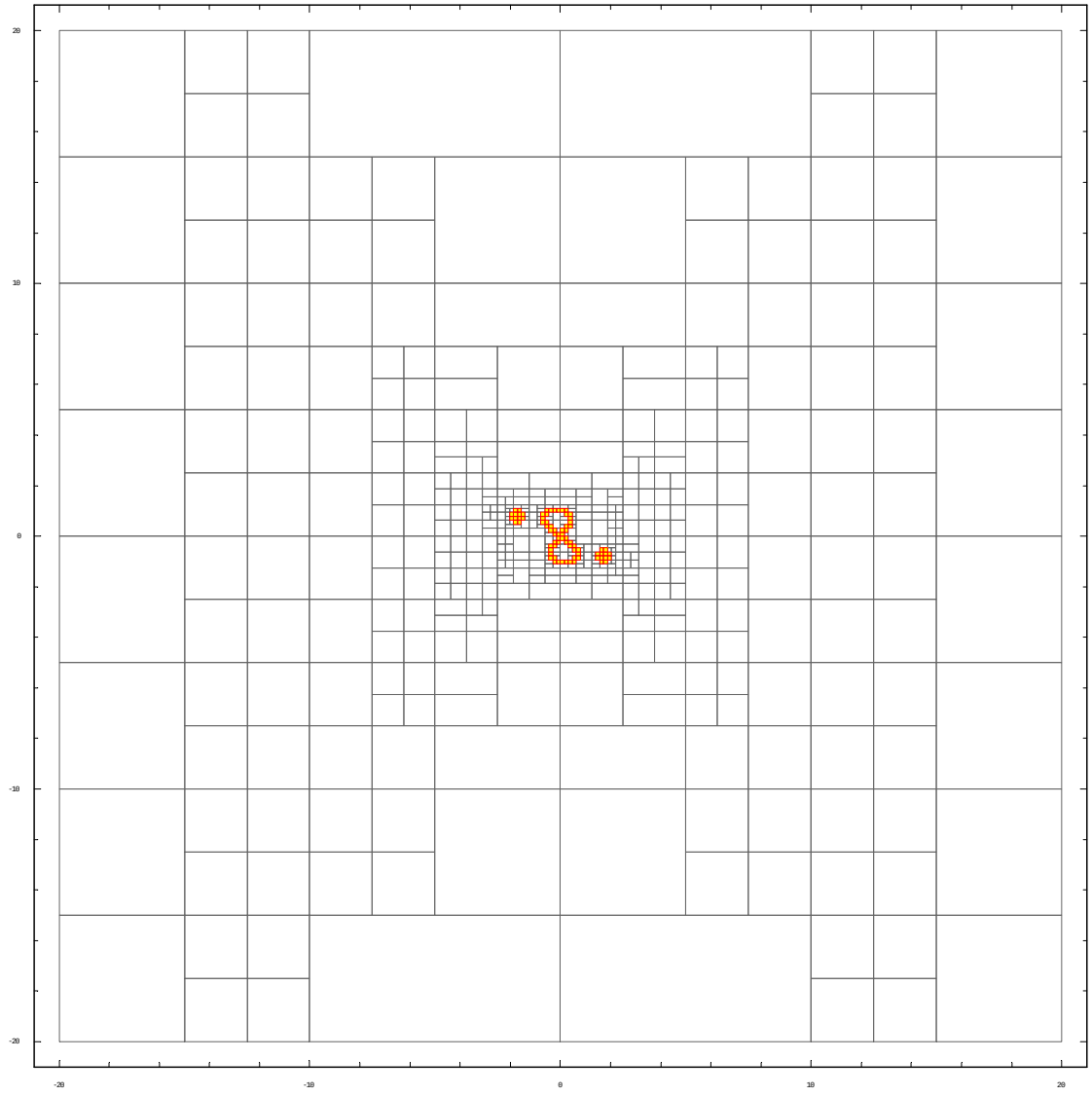


Figure 4.10: Plot of the solution regions produced by divide and conquer with Midpoint Taylor Forms on larger domains for a different function. The domain is $[-20,20]^2$. The algorithm finished in 1,099 iterations which took 5.016 seconds (Mathematica 5 time). Compare with figure 4.11.

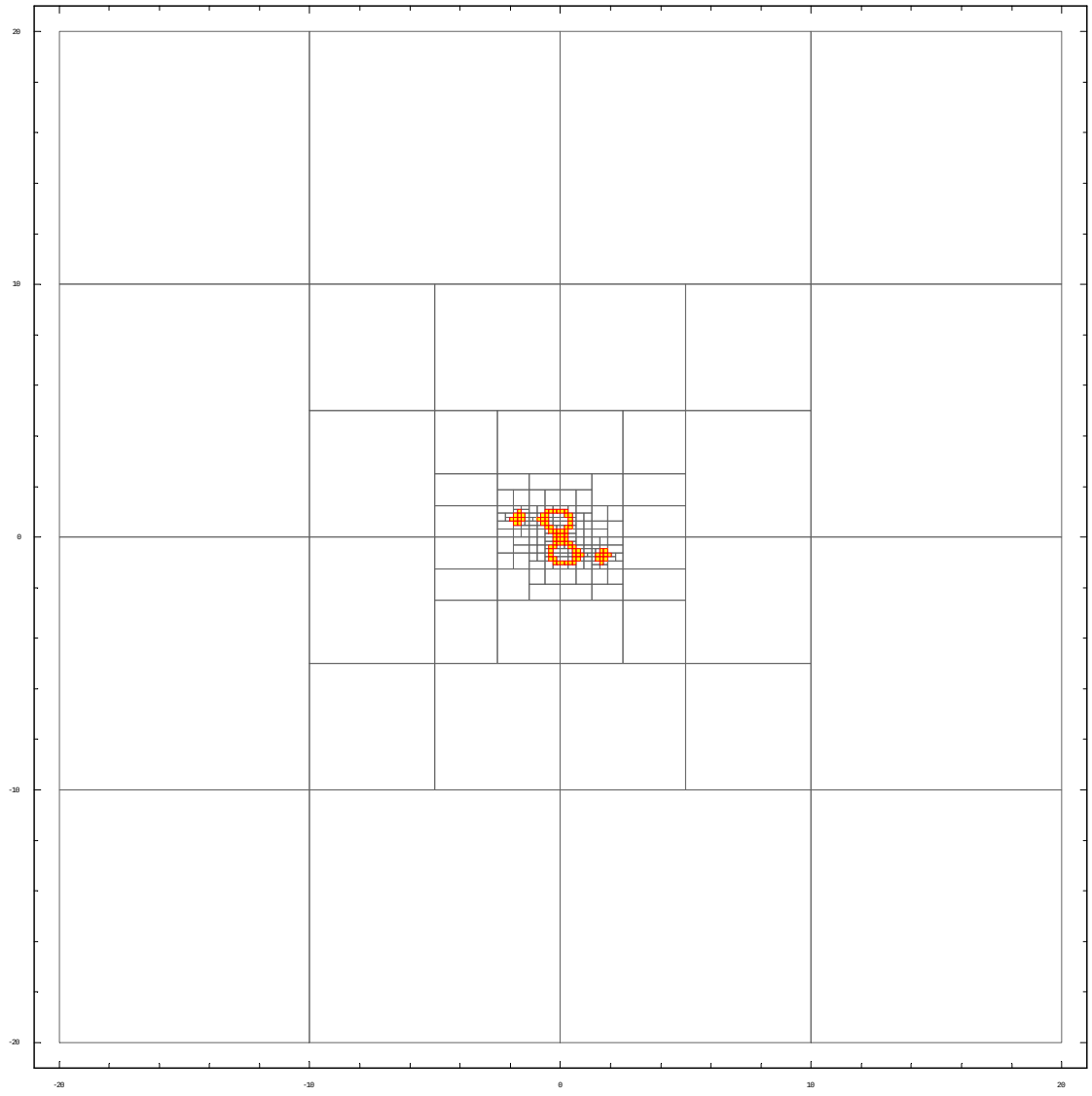


Figure 4.11: Plot of the solution regions produced by divide and conquer with Midpoint Taylor Forms on larger domains for a different function. The domain is $[-20,20]^2$. The algorithm finished in 543 iterations which took 2.531 seconds (Mathematica 5 time). Compare with figure 4.10.

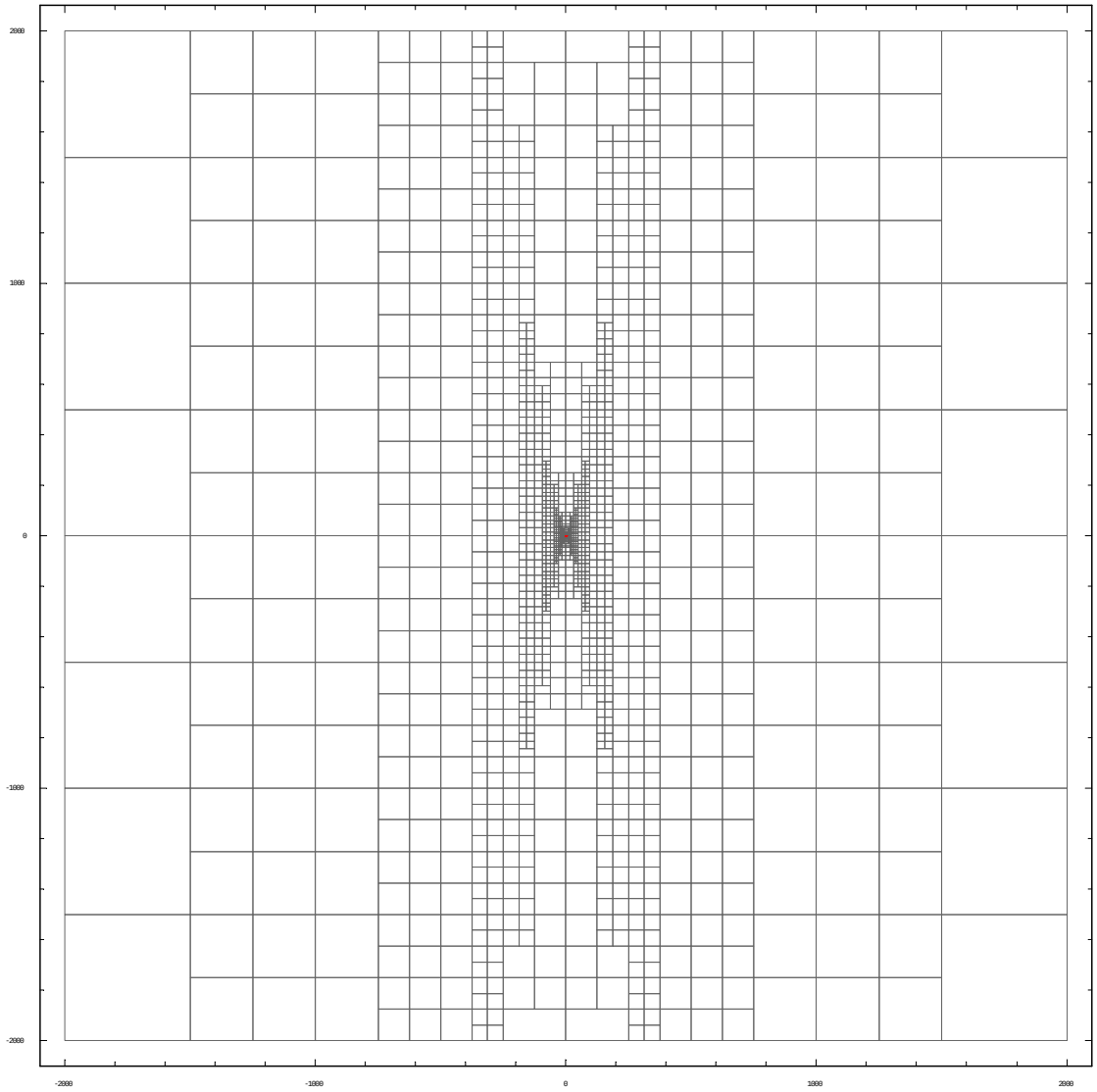


Figure 4.12: Plot of the solution regions produced by divide and conquer with Midpoint Taylor Forms on even larger domains. The domain is increased to $[-2000, 2000]^2$. The algorithm finished in 4,503 iterations which took 21.093 seconds (Mathematica 5 time). Compare with figure 4.13.

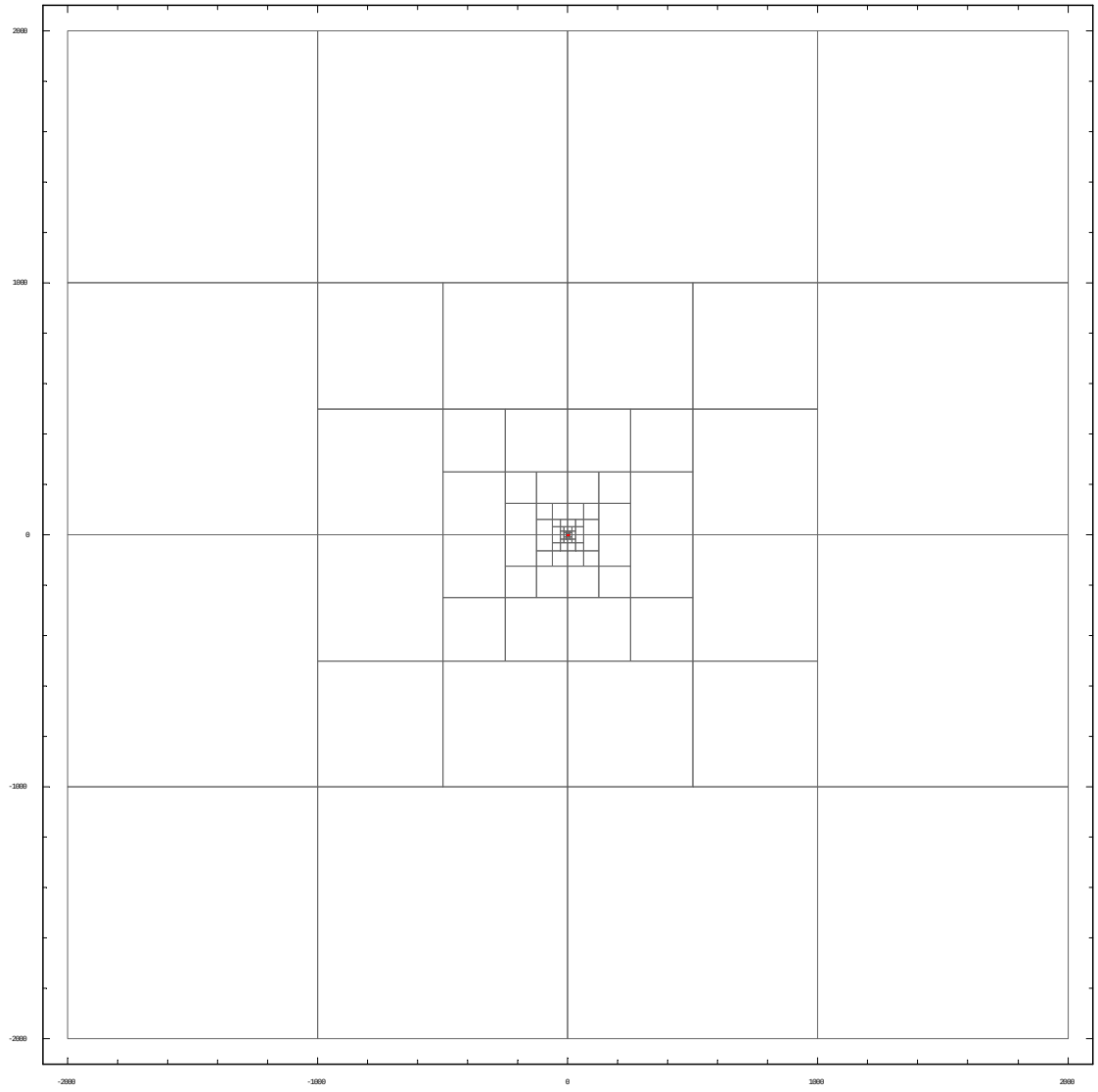


Figure 4.13: Plot of the solution regions produced by divide and conquer with Midpoint Taylor Forms on even larger domains. The domain is increased to $[-2000, 2000]^2$. The algorithm finished in 827 iterations which took 3.875 seconds (Mathematica 5 time). Once again we observe the fastest possible convergence of binary subdivision. Compare with figure 4.12.

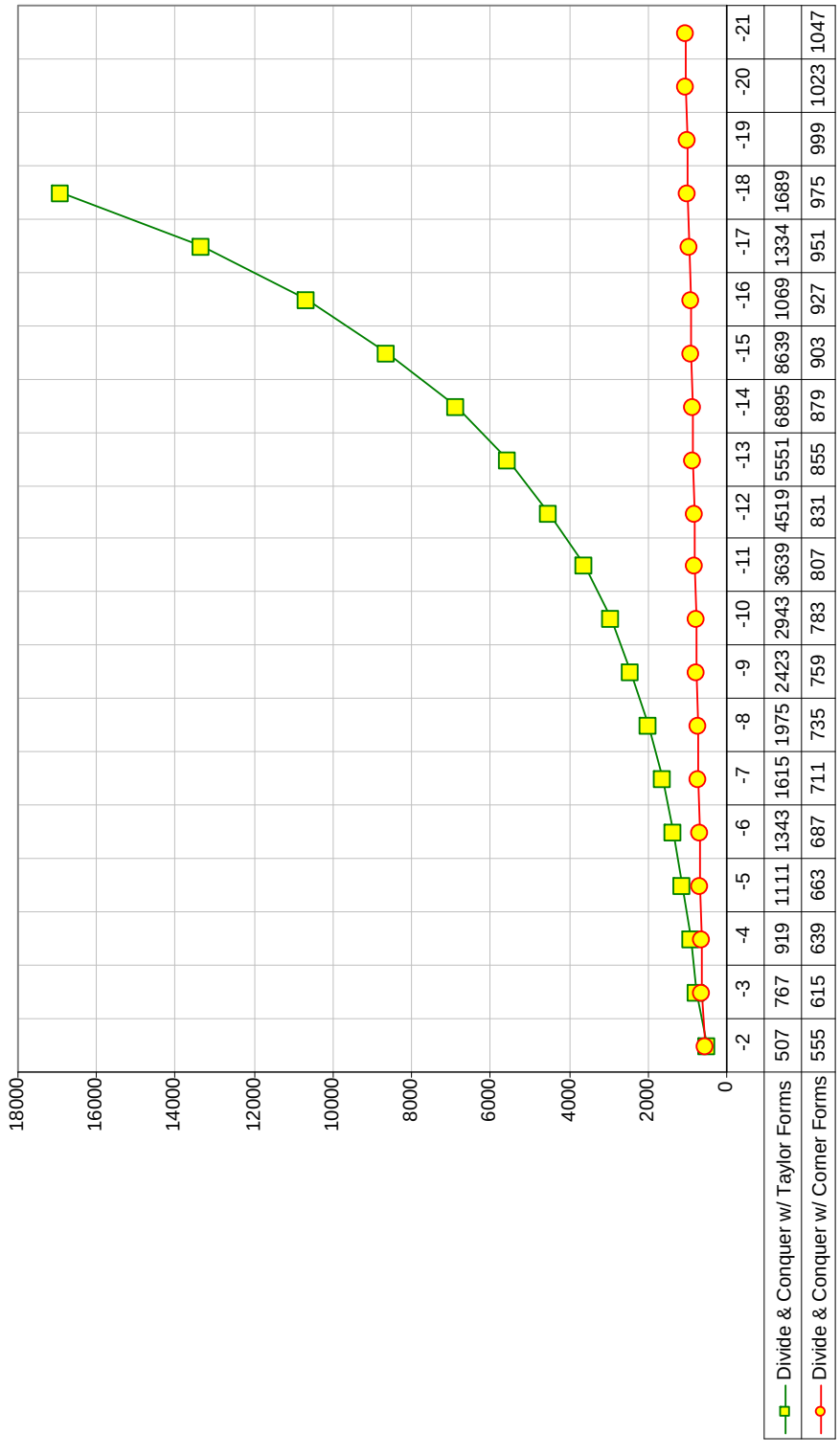


Figure 4.14: Logarithmic plot of the number of iterations required by various interval solution methods versus the size of the solution intervals expressed as a power of 2.

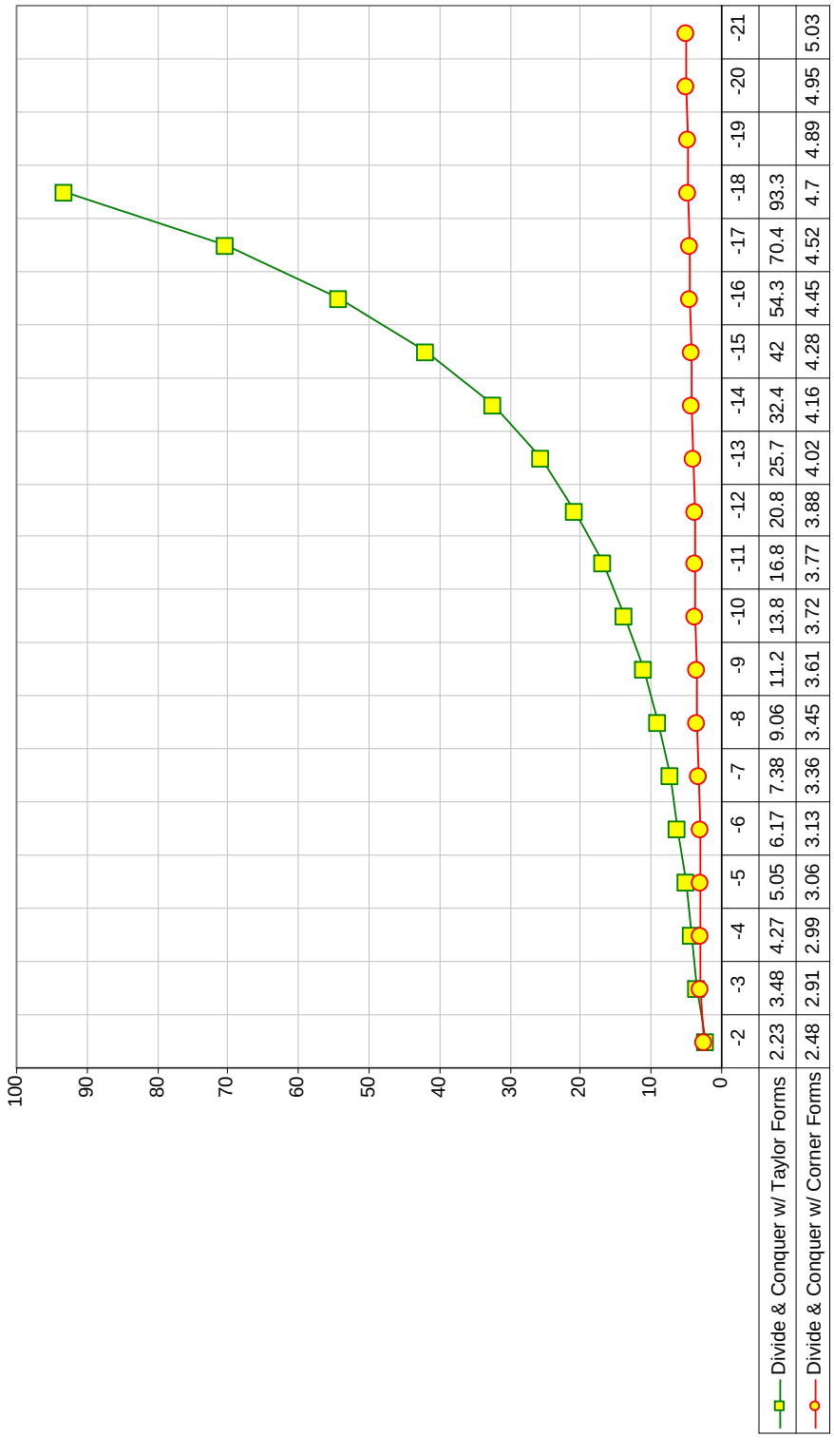


Figure 4.15: Logarithmic plot of the CPU time (Mathematica 5.0) required by various interval solution methods versus the size of the solution intervals expressed as a power of 2.

Chapter 5

Remainder Interval Newton Methods

In this chapter we develop the second contribution of the thesis: the *Remainder Interval Newton* method for nonlinear equations (RIN). We show that RIN methods are more effective in isolating solution regions than conventional Interval Newton methods (IN). For single equations RIN requires order of the square root as many steps as IN. For square systems, RIN is able to isolate solutions much faster than conventional IN does and is therefore more efficient overall.

Although RIN algorithms have slightly different forms for solving single equations and square systems, they follow the general outline below:

1. **Selection:** Pick the next candidate region from the list of regions to be investigated for solutions,
2. **Linearization:** Linearize the function over the candidate region,
3. **Solve Linearized Problem (Crop):** Yield a smaller, "cropped" region that is a tighter enclosure for the solution set. If the cropped region is the empty set there are no solutions anywhere inside our candidate region thus we skip to step 6,
4. **Test:** If the cropped region satisfies the termination criteria then add it to the solution set and skip to step 6,
5. **Subdivide:** (optional) If required, subdivide the cropped candidate region and add the subregions to the list of regions to be investigated for solutions,
6. **Repeat Until Search Is Exhausted.**

The above generalized algorithm is similar to other Interval Newton algorithms based on subdivision and cropping. The differences show up in the way RIN performs linearization and subdivision. Unlike conventional Interval Newton, RIN linearization produces systems of linear equations with real coefficients which can be solved with simple interval versions of any of the conventional methods: matrix inversion, Gauss elimination, Gauss-Seidel, etc. In contrast, IN requires finding solutions of linear equations/systems with interval coefficients which are slower to converge in the large.

Furthermore, RIN subdivision methods take advantage of the special linearization to produce *solution aligned subdivisions* and improve efficiency. Finally, the linearizations themselves conform to non-point solution regions much better than axis aligned boxes. This allows RIN to cover the solution set with much fewer solution regions. The combination of all these optimizations yields speed-ups of the order of the square root—both in the number of steps as well as in the number of solution regions returned which means a smaller memory footprint—when compared to conventional IN and divide and conquer search methods.

5.1 The RIN Algorithm for Roots of Multivariate Nonlinear Equations

Let us begin by investigating the problem of finding all the zeroes of a multivariate nonlinear equation:

$$f(\mathbf{x}) = 0, \tag{5.1}$$

where $f : \underline{\overline{\mathbf{d}}} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$. The RIN algorithm for solving this type of problem is shown in figure 5.1.

The algorithm performs a *depth first* search of the domain $\underline{\overline{\mathbf{d}}}$ and locates all solution subregions. If only one solution is needed the algorithm can be terminated as soon as the first solution region is located.

We chose the following termination criterion: a candidate region is declared a solution region if the range of f over the candidate region contains zero and:

```
RINSolveEquation (
  in f:    function whose solutions we are seeking
  in d:    domain interval
  in ε_s:  maximum size of a solution interval
  in ε_f:  maximum width of the linearized solution
  out S_ε: [empty] interval covering of the solutions of f in d
)
{
  create stack: [empty] stack of subintervals to be examined;
  put d on stack;
  while ( stack is not empty ) do
  {
    pop x from stack;
    // linearize f on x
    // r --- interval remainder
    RINLinearizeEquation ( in f, in x, c, out ∇f(c), out r );
    // estimate the range of f on x
    compute y = J(f)(x);
    if ( 0 ∈ y )
    // there could be solutions in x
    {
      RINCropEquation ( in ∇f(c), in r, c, in/out x );
      if ( w(x) < ε_s or w(r) / ||∇f(c)|| < ε_f )
        append x to S_ε;
      else
      {
        RINSubdivideEquation ( in ∇f(c), in r, in x, out subdivisionList );
        foreach x_i in subdivisionList
        {
          RINCropEquation ( in ∇f(c), in r, in/out x_i );
          if ( x_i is not empty )
            put x_i on stack;
        }
      }
    }
  }
  return; // search is exhausted
}
```

Figure 5.1: The Remainder Interval Newton algorithm for solving a single nonlinear equation.

1. The width of the candidate region is less than a used specified threshold ε_s, or

2. The width of the solution set linearization inside the candidate interval is less than another user specified threshold ε_f.

The second criterion will become clear after we define the linearization process in the next section.

5.1.1 Linearization

We linearize equation 5.1 by computing a first order Taylor expansion with interval Lagrange remainder—hence the name Remainder Interval Newton. This type of Taylor expansions are known as first order Taylor Models, cf. Berz [Berz and Hofstätter 1998].

If f is as in equation 5.1, $\overline{\underline{\mathbf{x}}}$ is a sub-interval vector of the interval domain $\overline{\underline{\mathbf{d}}}$, and $\mathbf{c}$ is a real vector in $\overline{\underline{\mathbf{x}}}$, the following derivation holds for all $\mathbf{x} \in \overline{\underline{\mathbf{x}}}$:

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{c}) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\mathbf{c})(x_i - c_i) + \sum_{u=1}^{n}\sum_{v=1}^{n} \frac{1}{2}\frac{\partial^2 f}{\partial x_u \partial x_v}(\xi)(x_u - c_u)(x_v - c_v) \\ &= f(\mathbf{c}) + (\mathbf{x}-\mathbf{c}) \cdot \nabla f(\mathbf{c}) + (\mathbf{x}-\mathbf{c}) H f(\xi) (\mathbf{x}-\mathbf{c})^{\mathsf{T}}, \end{aligned} \tag{5.2}$$

where $Hf(\xi)$ is the Hessian of f evaluated at $\xi \in [[\mathbf{x}, \mathbf{c}]] \subseteq \overline{\underline{\mathbf{x}}}$. We linearize 5.2 by replacing ξ and $\mathbf{x}$ with the interval vector $\overline{\underline{\mathbf{x}}}$:

$$f(\mathbf{x}) \in f(\mathbf{c}) + (\mathbf{x}-\mathbf{c}) \cdot \nabla f(\mathbf{c}) + (\overline{\underline{\mathbf{x}}} - \mathbf{c}) H f(\overline{\underline{\mathbf{x}}}) (\overline{\underline{\mathbf{x}}} - \mathbf{c})^{\mathsf{T}}. \tag{5.3}$$

If we reorganize 5.3 by moving the constant term $f(\mathbf{c})$ at the end we get:

$$f(\mathbf{x}) \in (\mathbf{x}-\mathbf{c}) \cdot \nabla f(\mathbf{c}) + \underbrace{(\overline{\underline{\mathbf{x}}} - \mathbf{c}) \mathsf{H} f(\overline{\underline{\mathbf{x}}}) (\overline{\underline{\mathbf{x}}} - \mathbf{c})^{\mathsf{T}} + f(\mathbf{c})}_{\overline{\underline{r}}},$$

and the short form of the interval linearization is:

$$f(\mathbf{x}) \in \mathsf{L}_f(\mathbf{x}) = (\mathbf{x}-\mathbf{c}) \cdot \nabla f(\mathbf{c}) + \overline{\underline{r}}\,.$$

```
RINLinearizeEquation (
  in f:        function to be linearized
  in x̲:        interval over which to linearize f
  in c:        linearization point (inside x̲)
  out ∇f(c):   gradient of f
  out r̲:       remainder interval
)
{
  // compute the linear Taylor Model approximation of f:
  compute ∇f(c) at c ∈ x̲;
  compute r̲ = (x̲ − c)Hf(x̲)(x̲ − c)^T + f(c);
  return ∇f(c), r̲;
}
```

Figure 5.2: The Remainder Interval Newton linearization algorithm for solving a single nonlinear equation.

Let S_L be the solution set of the linear equation:

$$\mathsf{L}_f(\mathbf{x}) = 0, \tag{5.4}$$

over the interval $\underline{\mathbf{x}} \in \overline{\underline{\mathbf{d}}}$. The solution set S_L is comprised of all points between the lines implicitly defined by the following equations:

$$\underline{\mathsf{L}}_f(x) = (\mathbf{x}-\mathbf{c})\cdot\nabla f(\mathbf{c}) + \underline{r} = 0$$

$$\overline{\mathsf{L}}_f(x) = (\mathbf{x}-\mathbf{c})\cdot\nabla f(\mathbf{c}) + \overline{r} = 0.$$

An example for a two dimensional function $f:\mathbb{R}^2 \to \mathbb{R}$ is shown in figure 5.3.

It is easy to see that the solution set S of equation 5.1 is always a subset of any solution set S_L of the type defined above. Therefore, an interval covering of the solution set S_L is also a valid covering for S.

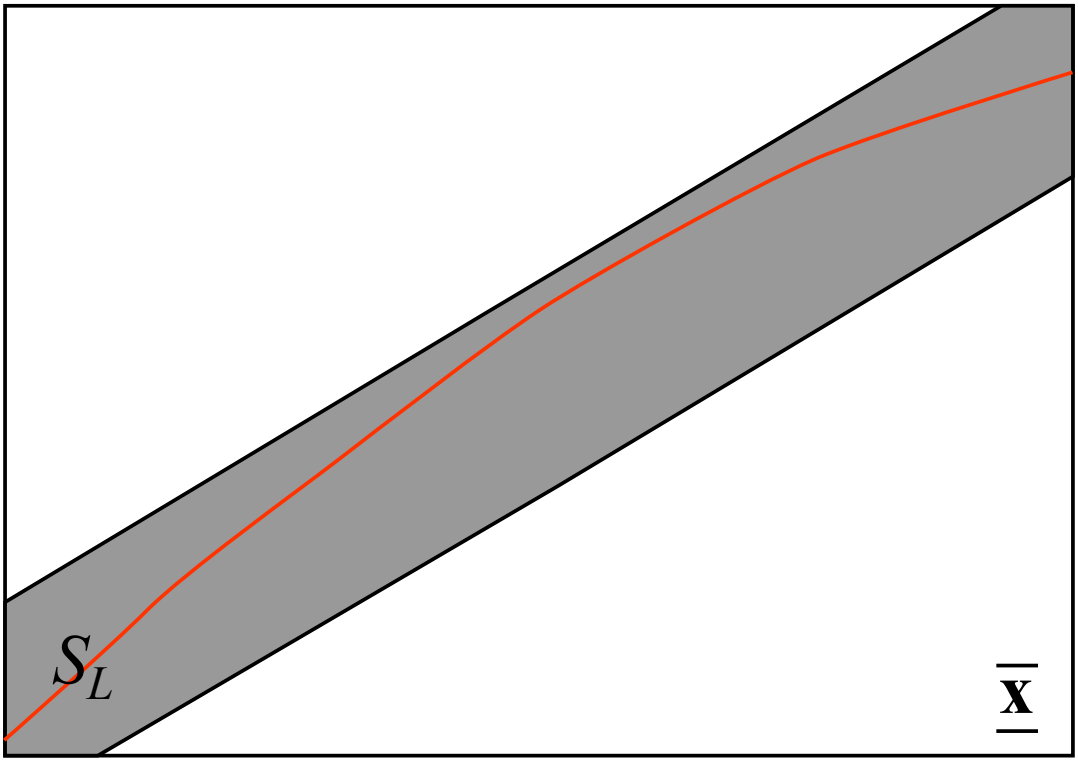


Figure 5.3: An example of a solution set of the linearized equation 5.4. The red curve represents the actual solution of equation 5.1 inside the interval vector $\overline{\underline{\mathbf{x}}}$. The grayed area S_L is the linearized solution.

5.1.2 Cropping

The solution set S_L of the linearized equation 5.4 can intersect the interval $\overline{\underline{\mathbf{x}}}$ in several ways, some of which are shown in figure 5.4.

Cropping—also referred to as (linear) tightening—is the process of computing the interval convex hull of the solution S_L of the linearized equation 5.4 over an interval $\overline{\underline{\mathbf{x}}}$. As we saw in the previous section, cropping does not lose any of the solutions of the original nonlinear equation.

A nice property of the RIN linearization is the fact that cropping can be done in closed form

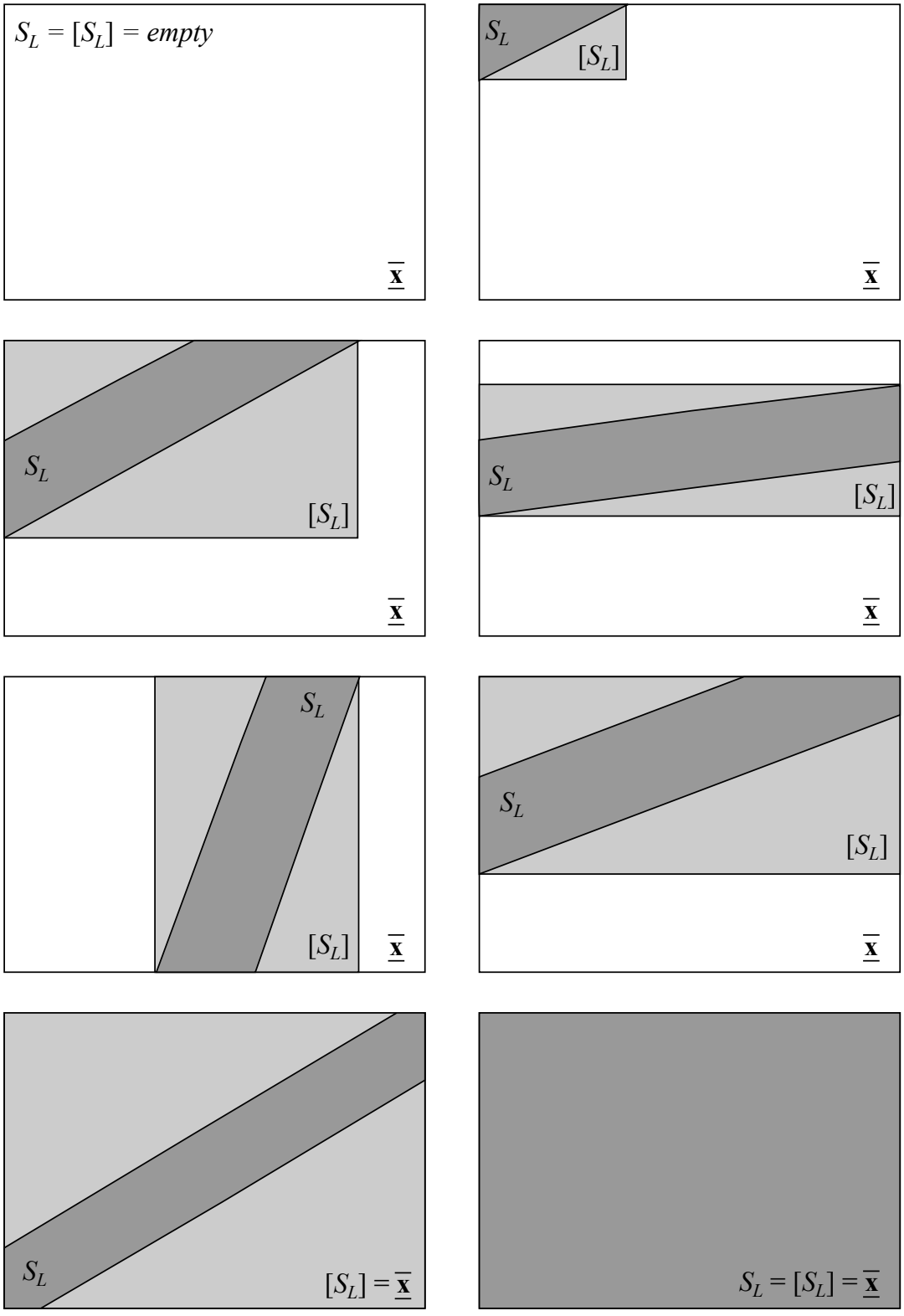


Figure 5.4: Several ways in which the linearized solution S_L can intersect an interval $\underline{\overline{\mathbf{x}}}$. $[[\,S_L\,]]$ is the interval convex hull of the intersection.

RINCropEquation (
  in $\nabla f(\mathbf{c})$**:** gradient of the function whose solutions we are seeking
  in $\overline{\underline{r}}$**:** remainder interval
  in c: linearization point (inside $\overline{\underline{\mathbf{x}}}$)
  in/out $\overline{\underline{\mathbf{x}}}$**:** interval to be cropped
)
{
  // solve the linear equation:
  // $\nabla f(\mathbf{c}) \cdot \mathbf{x} + \overline{\underline{r}} = 0$
  // over the interval $\overline{\underline{\mathbf{x}}}$.
  compute $\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}}) = \overline{\underline{r}} + (\overline{\underline{\mathbf{x}}} - \mathbf{c}) \cdot \nabla f(\mathbf{c})$**;**
  foreach component $\overline{\underline{x}}_i$ **of** $\overline{\underline{\mathbf{x}}}$
    compute $\overline{\underline{x}}_i = \overline{\underline{x}}_i \cap \left(c_i - \dfrac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}}) \ominus \dfrac{\partial f}{\partial x_i}(\mathbf{c})(\overline{\underline{x}}_i - c_i)}{\dfrac{\partial f}{\partial x_i}(\mathbf{c})} \right)$**;**
  return $\overline{\underline{\mathbf{x}}}$**;**
}

Figure 5.5: The Remainder Interval Newton cropping algorithm for solving a single nonlinear equation.

and independently in each dimension using the following formulas:

$$[[\, S_L \,]]_i = \overline{\underline{x}}_i \cap \left(c_i - \frac{\overline{\underline{r}} + \sum_{k \neq i} \frac{\partial f}{\partial x_k}(c_k)(\overline{\underline{x}}_k - c_k)}{\frac{\partial f}{\partial x_i}(c_i)} \right)_{i=1..n} .$$

Efficient component-wise evaluation of the cropping formulas can be done using the *inverse interval minus* operation. Let the range of $\mathsf{L}_f(\mathbf{x})$ be:

$$\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}}) = \overline{\underline{r}} + (\overline{\underline{\mathbf{x}}} - \mathbf{c}) \cdot \nabla f(\mathbf{c}) .$$

Then the cropping formula becomes:

$$[[\,S_L\,]]_i = \overline{\underline{x}}_i \cap \left(c_i - \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}}) \ominus \dfrac{\partial f}{\partial x_i}(c_i)(\overline{\underline{x}}_i - c_i)}{\dfrac{\partial f}{\partial x_i}(c_i)} \right)_{i=1..n},$$

which no longer requires computing the sum for each component. We introduce some simplifying notation:

$$\begin{aligned} \mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i\}} &= \mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}}) \ominus \frac{\partial f}{\partial x_i}(c_i)(\overline{\underline{x}}_i - c_i), \\ \mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i,j\}} &= \mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}}) \ominus \frac{\partial f}{\partial x_i}(c_i)(\overline{\underline{x}}_i - c_i) \ominus \frac{\partial f}{\partial x_j}(c_j)(\overline{\underline{x}}_j - c_j). \end{aligned}$$

With the above notation, the cropping formula is:

$$[[\,S_L\,]]_i = \overline{\underline{x}}_i \cap \left(c_i - \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i\}}}{\dfrac{\partial f}{\partial x_i}(c_i)} \right)_{i=1..n},$$

The next theorem proves that the order in which the components of the interval hull $[[\,S_L\,]]$ are computed is irrelevant and the same result (excluding any roundoff errors) is produced for all permutations.

Theorem 5.1.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable over the interval vector $\overline{\underline{\mathbf{x}}}$. For any arbitrary pair of indices $i, j \leq n$, $i \neq j$, define the cropped intervals $\overline{\underline{x}}_i^*$ and $\overline{\underline{x}}_j^*$ as:*

$$\overline{\underline{x}}_i^* = \overline{\underline{x}}_i \cap \left(c_i - \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i\}}}{\dfrac{\partial f}{\partial x_i}(c_i)} \right) \quad \overline{\underline{x}}_j^* = \overline{\underline{x}}_j \cap \left(c_j - \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,j\}}}{\dfrac{\partial f}{\partial x_j}(c_j)} \right).$$

Furthermore, define $\overline{\underline{x}}_i^{**}$ as:

$$\overline{\underline{x}}_i^{**} = \overline{\underline{x}}_i \cap \left(c_i - \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i,j\}} + \dfrac{\partial f}{\partial x_j}(c_j)(\overline{\underline{x}}_j^* - c_j)}{\dfrac{\partial f}{\partial x_i}(c_i)} \right).$$

$\underline{\overline{x}}_i^{**}$ is different from $\underline{\overline{x}}_i^{*}$ in that it is cropped using the cropped $\underline{\overline{x}}_j^{*}$ in place of the original $\underline{\overline{x}}_j$. Then:

$$\underline{\overline{x}}_i^{*} = \underline{\overline{x}}_i^{**}.$$

Proof:

The following chain of derivations holds:

$$\begin{aligned}
\frac{\partial f}{\partial x_j}(c_j)\,(\underline{\overline{x}}_j^{*} - c_j) &= \frac{\partial f}{\partial x_j}(c_j)\left((\underline{\overline{x}}_j - c_j) \cap \left(-\frac{\mathcal{R}(\mathsf{L}_f)(\underline{\overline{\mathbf{x}}})_{\{\ominus,j\}}}{\frac{\partial f}{\partial x_j}(c_j)}\right)\right) \\
&= \left(\frac{\partial f}{\partial x_j}(c_j)\,(\underline{\overline{x}}_j - c_j)\right) \cap \left(-\frac{\partial f}{\partial x_j}(c_j)\,\frac{\mathcal{R}(\mathsf{L}_f)(\underline{\overline{\mathbf{x}}})_{\{\ominus,j\}}}{\frac{\partial f}{\partial x_j}(c_j)}\right) \\
&= \left(\frac{\partial f}{\partial x_j}(c_j)\,(\underline{\overline{x}}_j - c_j)\right) \cap \left(-\mathcal{R}(\mathsf{L}_f)(\underline{\overline{\mathbf{x}}})_{\{\ominus,j\}}\right)
\end{aligned}$$

If we use the above to rewrite the fraction in the expression of $\underline{\overline{x}}_i^{**}$ we can derive the follow-

ing:

$$\frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i,j\}} + \dfrac{\partial f}{\partial x_j}(c_j)(\underline{\overline{x}}_j^* - c_j)}{\dfrac{\partial f}{\partial x_i}(c_i)} =$$

$$= \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i,j\}} + \left(\left(\dfrac{\partial f}{\partial x_j}(c_j)(\underline{\overline{x}}_j - c_j)\right) \cap \left(-\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,j\}}\right)\right)}{\dfrac{\partial f}{\partial x_i}(c_i)}$$

$$= \frac{\left(\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i,j\}} + \dfrac{\partial f}{\partial x_j}(c_j)(\underline{\overline{x}}_j - c_j)\right) \cap \left(\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i,j\}} \ominus \mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,j\}}\right)}{\dfrac{\partial f}{\partial x_i}(c_i)}$$

$$= \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i\}} \cap \left(-\dfrac{\partial f}{\partial x_i}(c_i)(\underline{\overline{x}}_i - c_i)\right)}{\dfrac{\partial f}{\partial x_i}(c_i)}$$

$$= \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i\}}}{\dfrac{\partial f}{\partial x_i}(c_i)} \cap (c_i - \underline{\overline{x}}_i).$$

Then, $\underline{\overline{x}}_i^{**}$ is equal to:

$$\begin{aligned}
\underline{\overline{x}}_i^{**} &= \underline{\overline{x}}_i \cap \left(c_i - \left(\frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i\}}}{\dfrac{\partial f}{\partial x_i}(c_i)} \cap (c_i - \underline{\overline{x}}_i)\right)\right) \\
&= \underline{\overline{x}}_i \cap \left(c_i - \frac{\mathcal{R}(\mathsf{L}_f)(\overline{\underline{\mathbf{x}}})_{\{\ominus,i\}}}{\dfrac{\partial f}{\partial x_i}(c_i)}\right) \cap \underline{\overline{x}}_i \\
&= \underline{\overline{x}}_i^*,
\end{aligned}$$

which proves our theorem. □

Thus, the RIN cropping formula computes the interval hull of S_L exactly—up to roundoff errors. Note that the cropping operation is usually much cheaper than the computation of a new linearization. We will make use of this fact in our subdivision strategy, which is described in the

next section.

5.1.3 Subdivision

Although cropping can sometimes reduce the size of the interval significantly, often times the reduction is small, and sometimes there is no reduction at all. The bottom two examples in figure 5.4 illustrate this case. We notice, however, that the area of the linearized solution S_L is usually much smaller than the area of the interval, such as in the bottom left example in the same figure. The bottom right example occurs only when $\overline{\underline{\mathbf{x}}}$ is large and the remainder is large as well. As the size of the interval decreases through subdivision the remainder decreases with quadratic order and becomes very small very quickly, see section 5.1.4.

RINSubdivideEquation (
  in $\nabla f(\mathbf{c})$: gradient of the function whose solutions we are seeking
  in $\overline{\underline{r}}$: remainder interval
  in $\overline{\underline{\mathbf{x}}}$: interval to be cropped
  out *subdivisionList*: list of subintervals of $\overline{\underline{\mathbf{x}}}$
)
{
  // subdivide $\overline{\underline{\mathbf{x}}}$ with the gradient of f
  compute index i **corresponding to** $\max_i \left(\overline{\underline{x}}_i \dfrac{\partial f}{\partial x_i}(\mathbf{c}) \right)$;
  compute $subdivSize = \dfrac{\mathrm{w}(\overline{\underline{r}})}{\dfrac{\partial f}{\partial x_i}(\mathbf{c})}$;
  compute *numSubdivisions* = **Max** $\left(2, \dfrac{\mathrm{w}(\overline{\underline{x}}_i)}{subdivSize} \right)$;
  subdivide $\overline{\underline{\mathbf{x}}}$ **into** *numSubdivisions* **subintervals along** i^{th} **dimension**;
  add the subintervals to *subdivisionList*;
  return *subdivisionList*;
}

Figure 5.6: The Remainder Interval Newton subdivision algorithm for solving a single nonlinear equation.

The subdivision algorithm shown in figure 5.6 allows us to crop away much of the area of the interval hull $[[\, S_L \,]]$ outside of the linearized solution S_L. The key steps are as follows:

1. **Subdivision Axis Selection:** Choose the k^{th} coordinate axis to subdivide along.

2. **Subdivision:** Subdivide $[[\, S_L \,]]$ into n subintervals along the k^{th} component.

3. **Crop Each Subdivision:** Each subinterval is cropped using the same linearization.

4. **Return the List of Subintervals.**

There are many different ways to choose the subdivision axis and the number of subdivisions, depending on the desired characteristics of the final interval covering of the solution. If we desire a covering with the smallest number of intervals then the following criteria for selection produce good results:

- **Subdivision Axis Selection:** k is the coordinate axis closest to perpendicular to the gradient of f. This particular choice ensures the maximum reduction of the remainder in the following linearization step. Thus, k corresponds to the maximum

$$\max_k \left(\bar{\underline{x}}_k \frac{\partial f}{\partial x_k}(\mathbf{c}) \right).$$

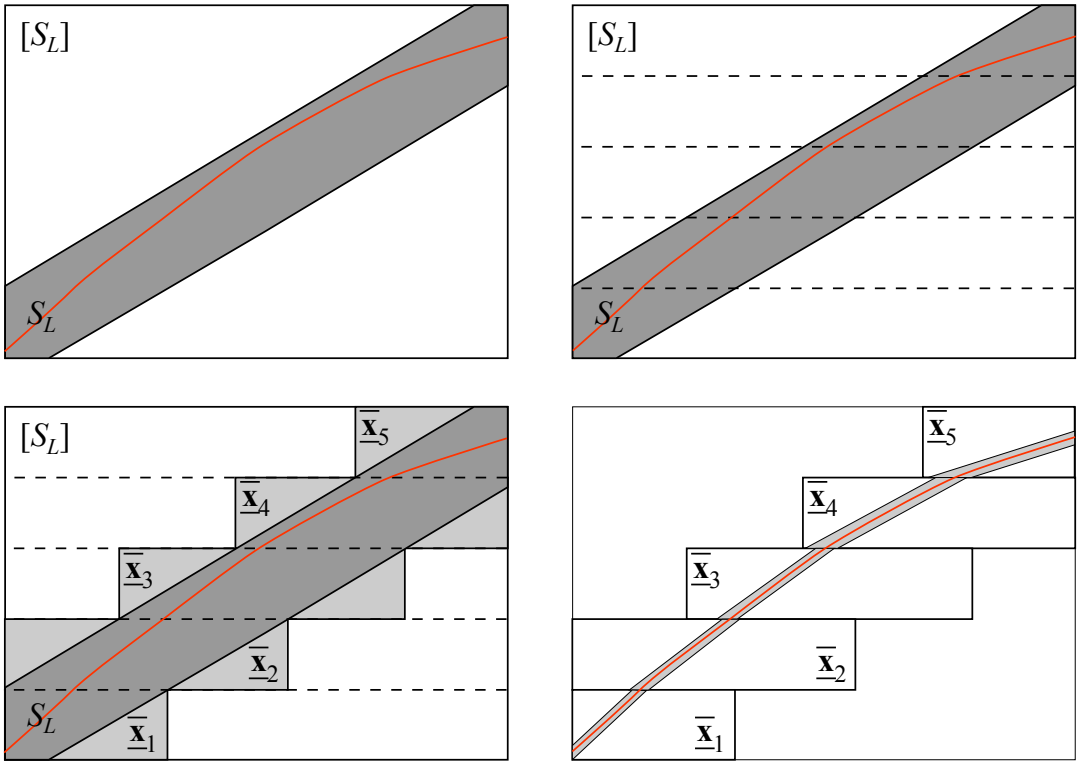
- **Subdivision:** The number of subdivisions is chosen such that the size of the subintervals produced is almost equal to but less than the size of the linearized solution S_L along the k^{th} coordinate axis:

$$n = \max \left(2, \frac{\mathrm{w}(\bar{\underline{x}}_i)}{\mathrm{w}(\bar{\underline{r}})} \frac{\partial f}{\partial x_i}(\mathbf{c}) \right),$$

  where n has to be at least two.

The subdivision process is illustrated graphically in figure 5.1.3. The top left image shows a typical configuration. The top right image shows the subintervals generated after the subdivision step. The bottom left image shows the cropped subintervals. Finally, the bottom right image shows the cropped subintervals and the newly recomputed linearizations. Note the accelerated convergence.

Most interval root finding algorithms employ termination criteria based on the size of the subdivided boxes. RIN methods can use the same criteria. However, they can also use the width



The RIN subdivision process. *Top left image:* a typical configuration. *Top right image:* the subintervals generated after the subdivision step. *Bottom left image:* the cropped subintervals. *Bottom right image:* the cropped subintervals and the newly recomputed linearizations. Note the accelerated convergence.

of the linearized solution S_L as a termination criterion, since the polygon representing it can easily be computed in closed form.

A simple analysis shows that the number of rectangular boxes of width ε needed to tile a curve of length l is of order $O\left(\frac{l}{\varepsilon}\right)$. The same curve requires only order $\sqrt{O\left(\frac{l}{\varepsilon}\right)}$ linearized solution regions of width ε. The same reduction in order occurs in more than two dimensions. The examples shown in section 5.5 confirm that significant savings do occur in practice.

5.1.4 Convergence of the RIN Algorithm

In this section we show that the width of the cropped and subdivided intervals decreases with quadratic order of convergence.

The RIN remainder of a function f over an interval vector $\overline{\underline{\mathbf{x}}}$ was defined as:

$$\overline{\underline{r}} = (\overline{\underline{\mathbf{x}}} - \mathbf{c})\mathsf{H}f(\overline{\underline{\mathbf{x}}})(\overline{\underline{\mathbf{x}}} - \mathbf{c})^{\mathsf{T}} + f(\mathbf{c})$$

Thus, the width of the remainder is:

$$\mathrm{w}(\overline{\underline{r}}) = \mathrm{w}\left((\overline{\underline{\mathbf{x}}} - \mathbf{c})\mathsf{H}f(\overline{\underline{\mathbf{x}}})(\overline{\underline{\mathbf{x}}} - \mathbf{c})^{\mathsf{T}}\right),$$

and the order is:

$$O(\mathrm{w}(\overline{\underline{r}})) \quad = \quad \mathrm{mag}(\mathsf{H}f(\overline{\underline{\mathbf{x}}}))\, O^2(\mathrm{w}(\overline{\underline{\mathbf{x}}})).$$

The width of the remainder is quadratic with respect to the width of the interval region $\overline{\underline{\mathbf{x}}}$. The subdivision algorithm produces intervals of width proportional to the width of the remainder, thus the size of the subdivided intervals $\overline{\underline{\mathbf{x}}}_i$ is also quadratic with respect to the size of the original interval $\overline{\underline{\mathbf{x}}}$. Therefore, the RIN algorithm has quadratic order of convergence to the solution set S.

5.2 The RIN Algorithm for Roots of Square Systems of Nonlinear Equations

We now consider the problem of finding all the zeroes of a square nonlinear system of the form:

$$\mathbf{f}(\mathbf{x}) = 0, \tag{5.5}$$

where $\mathbf{f} : \overline{\underline{\mathbf{d}}} \subseteq \mathbb{R}^n \to \mathbb{R}^n$. The RIN algorithm for solving this type of problem is shown in figure 5.7.

5.2.1 Linearization

Each equation of the system is linearized using the methods presented in the previous sections. Of course, the collection of gradients is the Jacobian of $\mathbf{f}$, and the interval remainder $\overline{\underline{\mathbf{r}}}$ is now a

```
RINSolveSquareSystem (
  in f:    vector of functions whose solutions we are seeking
  in d:    domain interval
  in ε:    maximum size of a solution interval
  in J(·): interval extension operator
  out S_ε: [empty] interval covering of the solutions of f in d
)
{
  create stack: [{d}] stack of subintervals to be examined;
  while ( stack is not empty ) do
  {
    pop x from stack;
    set shrunk = True;
    while ( shrunk ) do
    { // J_f(c) - Jacobian of f evaluated at c
      // r - interval remainder
      RINLinearizeSystem ( in f, in x, c, out J_f(c), out r ); // linearize f
      compute y = ⋂_i J(f_i)(x); // estimate the ranges of f_i on x
      if ( 0 ∈ y )
      { // there could be solutions in x
        if (w(x) < ε)
          append x to S_ε; // found a solution
        else
        {
          RINCropSquareSystem ( in J_f(c), in r, c, in/out x, in/out shrunk );
          if ( shrunk == False )
          {
            RINTightenSystem ( in J_f(c), in r, c, in/out x ); // optional
            BinarySubdivide ( in x, out x_1, out x_2 );
            put x_1 on stack;
            put x_2 on stack;
          }
        }
      }
    }
  }
  return; // search is exhausted
}
```

Figure 5.7: The Remainder Interval Newton algorithm for solving square systems of nonlinear equations.

```
RINLinearizeSystem (
  in f:       functions to be linearized
  in x̄:       interval over which to linearize f
  in c:       linearization point (inside x̄)
  in 𝓘(·):    interval extension operator
  out J_f(c): Jacobian of f evaluated at c
  out r̄:      vector of remainder intervals
)
{
  foreach component f_i of f
  {
    // compute the linear Taylor Model approximation of f_i:
    compute ∇f_i(c) at c ∈ x̄ ;
    compute r̄_i = 𝓘( (x − c) H f_i(x) (x − c)^T ) ( x̄ ) + f_i(c);
  }
  return J_f(c), r̄ ;
}
```

Figure 5.8: The Remainder Interval Newton linearization algorithm for solving square systems of nonlinear equations.

vector of intervals with n components:

$$\mathbf{f}(\mathbf{x}) \in \mathsf{L}_{\mathbf{f}}(\mathbf{x}) = \mathsf{J}_{\mathbf{f}}(\mathbf{c}) \cdot (\mathbf{x} - \mathbf{c}) + \overline{\underline{\mathbf{r}}}\,.$$

Note that the Jacobian is a real valued matrix.The pseudocode for the linearization algorithm is shown in figure 5.8.

As before, the solution set S of the original system 5.5 is a subset of the solution set S_L of the linearization defined above. Therefore, an interval covering of the solution set S_L is also a valid covering for S.

5.2.2 Cropping

Let $A = \mathsf{J}_{\mathbf{f}}(\mathbf{c})$. Then, the cropping formulas are:

$$\overline{\underline{\mathbf{x}}}^{\,*} = \mathbf{c} - A^{-1}\,\overline{\underline{\mathbf{r}}}\,,$$

```
RINCropSquareSystem (
  in J_f(c):     Jacobian of f evaluated at c
  in r:          remainder interval
  in c:          linearization point (inside x)
  in/out x:      interval to be cropped
  in/out shrunk: true if x shrunk, i.e.  isolated a solution
)
{
  // solve the linear system:
  // J_f(c) · (x − c)^T = − r^T
  // over the interval x.
  set A = J_f(c); // matrix of the system
  compute x* = c − A^{-1} · r^T;
  compute shrunk = ( x* ⊂ x );
  set x = x ∩ x*;
  return x, shrunk;
}
```

Figure 5.9: The Remainder Interval Newton cropping algorithm for solving square systems of nonlinear equations.

and:

$$[[\, S_L \,]] = \overline{\mathbf{x}} \cap \overline{\mathbf{x}}^{\,*}.$$

If $\overline{\mathbf{x}}^{\,*} \subset \overline{\mathbf{x}}$ then there is at most one solution of $\mathbf{f}$ inside $\overline{\mathbf{x}}$, a property similar to conventional Interval Newton. The cropping algorithm is shown in figure 5.9.

5.2.3 Tightening

Tightening is an optional step. We found tightening to be a worthwhile optimization (both for RIN and conventional IN) as it often reduces running time by a significant percentage, see section 5.5.

The RIN tightening algorithm is shown in figure 5.10. It is nothing more than successive cropping of each individual equation in the system by the algorithm discussed in section 5.1.2.

Of course, if the cropping method presented in the previous section is successful at reducing the size of $\overline{\mathbf{x}}$ there is no need to perform tightening.

```
RINTightenSystem (
  in J_f(c):     Jacobian of f evaluated at c
  in r:          remainder interval
  in c:          linearization point (inside x)
  in/out x:      interval to be cropped
)
{
  // solve the linear system:
  // J_f(c) · (x − c)^T = − r^T
  // over the interval x.
  foreach component f_i of f
    RINCropEquation ( in ∇f_i(c), in r_i, c, in/out x^(i) );
  return x;
}
```

Figure 5.10: The Remainder Interval Newton tightening algorithm for solving square systems of nonlinear equations.

5.3 RIN vs. Martin Berz's Inversion

Martin Berz proposed a method that computes a Taylor Model of high degree of the inverse of a square system, see [Berz and Hoefkens 2001]. This inverse can then be evaluated at 0 to obtain an inclusion of the solution set of the original system. However, Taylor Model inversion is limited to square systems while RIN is not. Of course, the system must be invertible in over the region considered, so solutions must first be isolated by some other means. Isolating the solutions is probably the difficult part of solving systems in the first place.

In addition, RIN is much easier to implement and use and has lower costs per iteration.

5.4 RIN vs. Makino and Berz's LDB

The *Linearly Dominated Bounder* (LDB) was proposed by Makino and Berz as a method for computing tight inclusion functions over a given interval, see [Makino and Berz 2003]. The method is in effect performing a simplified optimization procedure to isolate the maxima and minima of the function in a given interval.

Some of the cropping formulas used by RIN are similar to formulas used by LDB. How-

ever, LDB has not been described as a method for isolating zeroes of functions, but only as a method for improving upper and lower bound estimates of the range. Moreover, RIN provides a special subdivision method that is vital in ensuring quadratic convergence in the presence of non-point solutions. Without this subdivision method convergence is much slower as shown by the following illustrations.

LDB is useful when very accurate bounds on a fixed interval are needed. Its usefulness diminishes when the bounds are only used as part of a root solving or optimization algorithm.

5.5 Examples and Performance

In this section we give some examples and compare the performance of RIN with other state of the art methods, particularly conventional Interval Newton.

5.5.1 Polynomial Equations

We evaluate the Remainder Interval Newton method on the following bivariate function:

$$f(x,y) = \cos 3x\,(\sin 2y + \cos 2y) + \cos 2x\,(\sin 3y - \cos 3y)\,.$$

For simplicity of implementation, we replaced f with its 5th order bivariate Taylor expansion f^* around the point $(1,1)$ whose expression was given in section 4.10.

The solution set of the equation $f^*(x,y) = 0$ restricted to the interval $[-\pi,\pi] \times [-\pi,\pi]$ is shown in figure 5.11.

We compared the performance of seven solution algorithms:

1. **Divide and Conquer with Natural Extensions:** The most basic interval solution method using binary subdivision and naive interval evaluations.

2. **Divide and Conquer with Taylor Form Interval Extensions:** Binary subdivision with Taylor Forms (centered form) interval evaluation for the range.

3. **Divide and Conquer with Corner Form Interval Extensions:** Same as above but using Corner Taylor Forms.

4. **Interval Newton with Taylor Form Interval Extensions:** Binary subdivision with Interval Newton cropping and Taylor Form range evaluation.

5. **Remainder Interval Newton with Binary Subdivision:** RIN with Corner Forms and binary subdivision - the subdivision scheme described in section 5.1.3 was not used.

6. **RIN with Special Subdivision:** Uses the subdivision scheme described in section 5.1.3. All solutions are axis aligned boxes.

7. **RIN with Non-Box Solutions:** Same as above but solutions are polygons representing the linearized solution regions.

Graphs for the number of iterations required (see figure 5.12), the CPU time required (see figure 5.13), and the number of solution regions produced (see figure 5.14), are shown.

Finally, plots of the solution regions computed by algorithms 1, 2, 3, 4, and 7 are shown in figures 5.15 through 5.23.

5.5.2 Polynomial Systems

To evaluate the performance of the RIN method for square systems we use the system of polynomial equations below:

$$\begin{cases} p(x,y) &= 0 \\ q(x,y) &= 0 \end{cases},$$

where $p(x,y) = (x+2.5)(x+1.5)(x+0.5)(x-0.5)(x-1.5)(x-2.5)(y+2.5)(y+1.5)(y+0.5)(y-0.5)(y-1.5)(y-2.5)$ and $q(x,y) = (x+3)(x+2)(x+1)(x)(x-1)(x-2)(x-3)(y+3)(y+2)(y+1)(y)(y-1)(y-2)(y-3)$.

We compare the performance of four solution algorithms:

1. **Interval Newton:** Interval Newton without tightening. Uses Taylor Forms for range evaluation.

2. **Interval Newton with Tightening:** Interval Newton with tightening and Taylor Forms for range evaluation.

3. **Remainder Interval Newton:** RIN without tightening. Uses Taylor Forms for range evaluation.

4. **Remainder Interval Newton with Tightening:** RIN with tightening and Taylor Forms for range evaluation.

Plots of the solution regions computed by each method are shown in figures 5.24 through 5.27. Note that RIN with tightening is the fastest method.

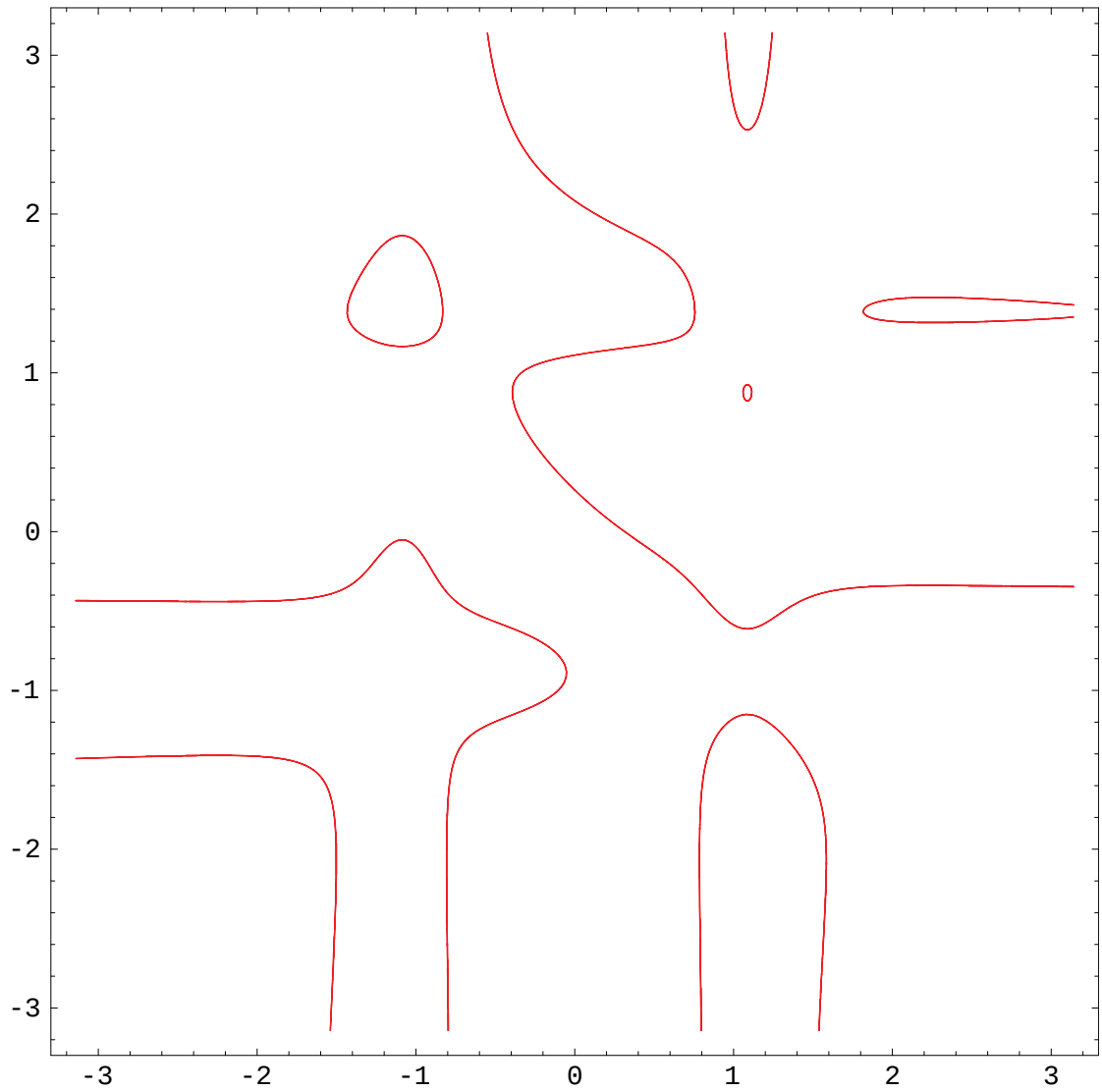
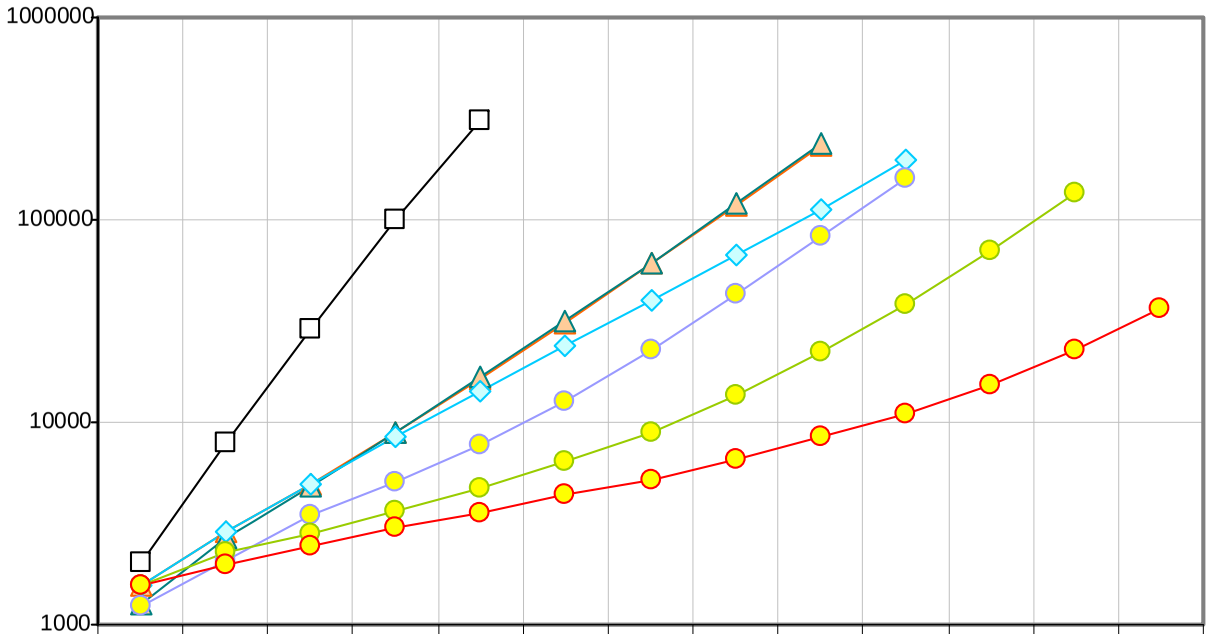
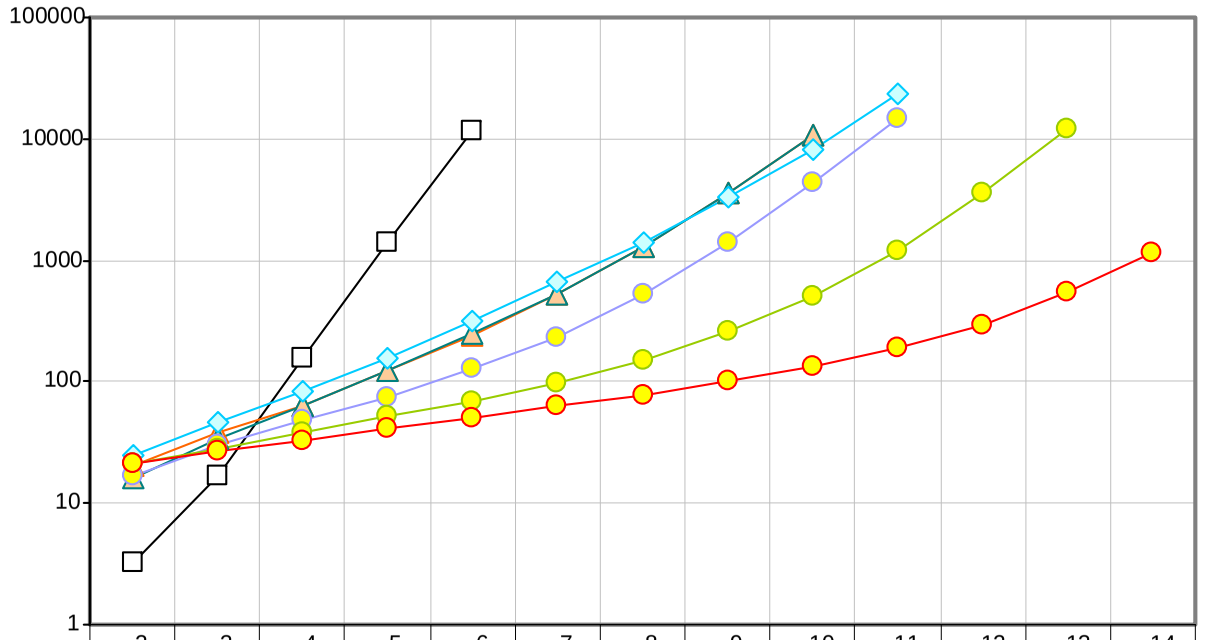


Figure 5.11: The solution set of the 5th order bivariate Taylor expansion around the point $(1,1)$ of the function $f(x,y) = \cos 3x(\sin 2y + \cos 2y) + \cos 2x(\sin 3y - \cos 3y)$ inside the interval $[-\pi,\pi] \times [-\pi,\pi]$. The curves are computed with RIN and are composed of 3,541 linearized solution regions of width at most 2^{-10}.



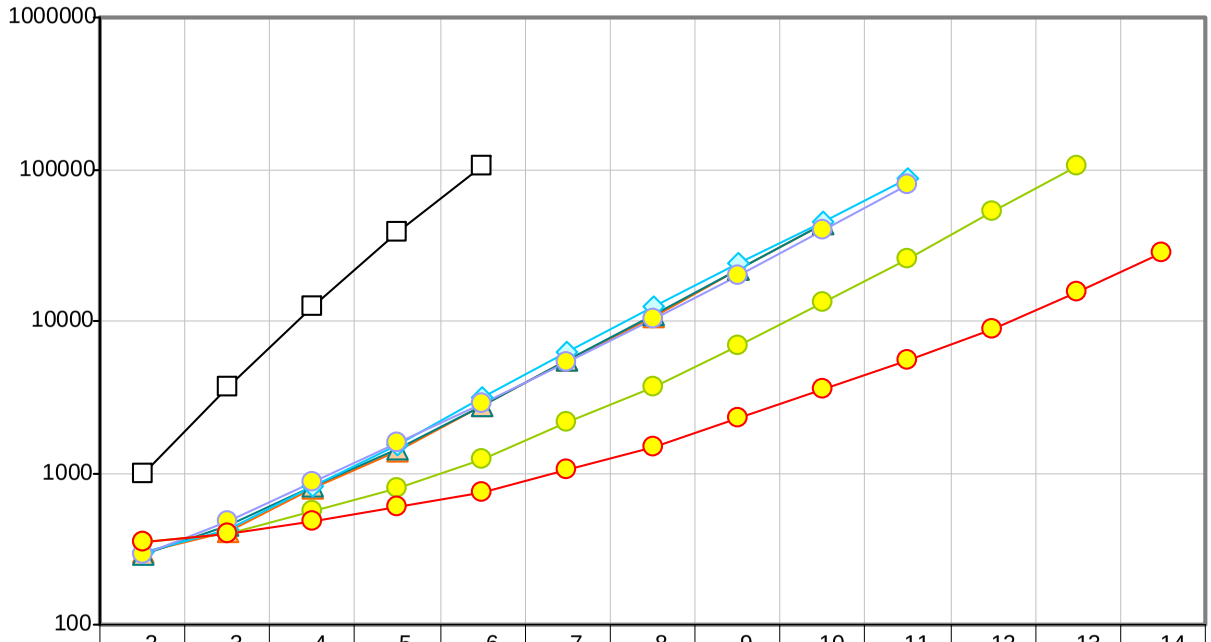
	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14
Divide & Conquer w/ Natural Extension	2035	7853	29105	99071	308167								
Divide & Conquer w/ Taylor Forms	1545	2907	4959	8947	16441	31181	60465	118697	235037				
Divide & Conquer w/ Corner Forms	1267	2653	4841	8973	16703	31631	61121	119581	236117				
Interval Newton w/ Taylor Forms	1545	2907	4953	8385	14287	24135	40379	67493	113747	198257			
RIN w/ Corner Forms and Binary Subdivision	1225	2049	3471	5067	7757	12759	22637	42473	82191	161613			
RIN w/ Corner Forms and Special Subdivision	1569	2261	2790	3650	4674	6406	8921	13674	22207	37739	70050	134361	
RIN w/ Corner Forms and Non Box Solutions	1569	1969	2429	2982	3581	4367	5196	6555	8413	11026	15109	22716	36228

Figure 5.12: Logarithmic plot of the number of iterations required by various interval solution methods versus the size of the solution intervals expressed as a power of 2.



	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14
Divide & Conquer w/ Natural Extension	3.203	16.5	158.75	1394.9	11787								
Divide & Conquer w/ Taylor Forms	20.063	37.813	64.656	121.89	239.47	526.7	1278.4	3559.8	10624				
Divide & Conquer w/ Corner Forms	16.359	34.109	62.672	121.59	247.53	528.38	1284.1	3569.9	10613				
Interval Newton w/ Taylor Forms	24.359	47.328	84.297	157.78	311.89	664.42	1421.8	3346	8173.1	23195			
RIN w/ Corner Forms and Binary Subdivision	16.469	30.734	48.453	74	127	232.78	536.42	1411.4	4336.9	14873			
RIN w/ Corner Forms and Special Subdivision	21.141	27.578	38.125	51.5	69.672	99.125	151.97	257.19	511.47	1185.1	3525.4	11921	
RIN w/ Corner Forms and Non Box Solutions	21.062	26.438	32.828	41.172	49.781	64.656	77.437	101.14	135.75	191.81	297.48	544.17	1173.8

Figure 5.13: Logarithmic plot of the CPU time (Mathematica 5.0) required by various interval solution methods versus the size of the solution intervals expressed as a power of 2.



	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14
Divide & Conquer w/ Natural Extension	986	3692	12407	38110	105977								
Divide & Conquer w/ Taylor Forms	302	415	788	1404	2740	5429	10789	21546	43021				
Divide & Conquer w/ Corner Forms	291	455	807	1450	2778	5473	10832	21585	43065				
Interval Newton w/ Taylor Forms	302	417	807	1508	3144	6284	12392	24042	45392	86463			
RIN w/ Corner Forms and Binary Subdivision	290	481	882	1557	2833	5299	10228	20142	40001	79712			
RIN w/ Corner Forms and Special Subdivision	350	402	559	802	1233	2130	3661	6893	13257	25737	51938	105538	
RIN w/ Corner Forms and Non Box Solutions	350	394	478	605	751	1046	1462	2286	3541	5523	8860	15536	27990

Figure 5.14: Logarithmic plot of the number of solution regions produced by various interval solution methods versus the size of the solution intervals expressed as a power of 2.

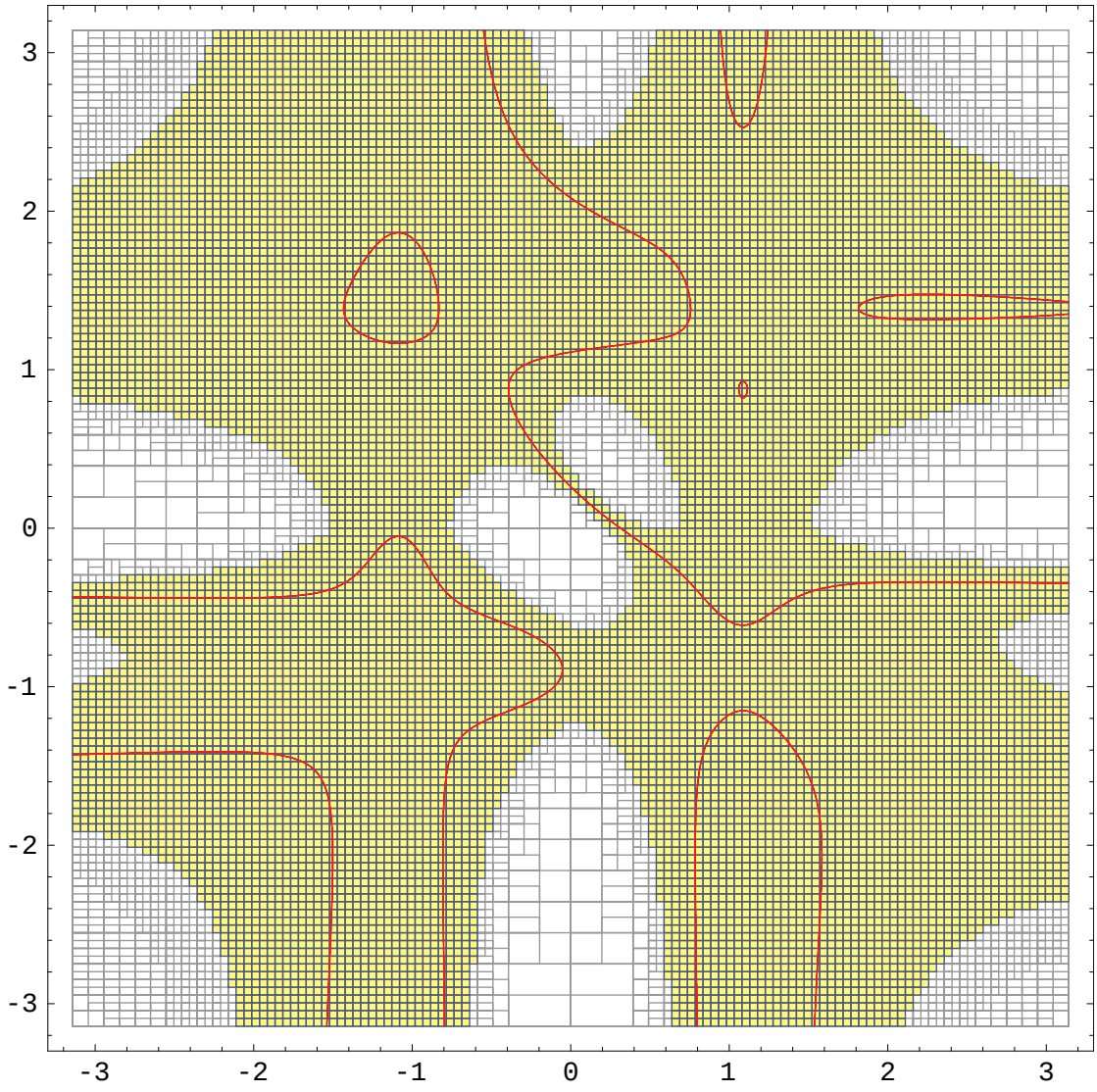


Figure 5.15: Plot of the solution regions produced by Divide and Conquer with Naive Natural Extension. The solution box width is less than 2^{-4}. The algorithm found 12,407 solution regions in 29,105 iterations which took 84.281 seconds (Mathematica 5 time). Note that it would have taken considerably more time to produce the same level of solution separation that was possible using the more advanced methods shown on the following pages.

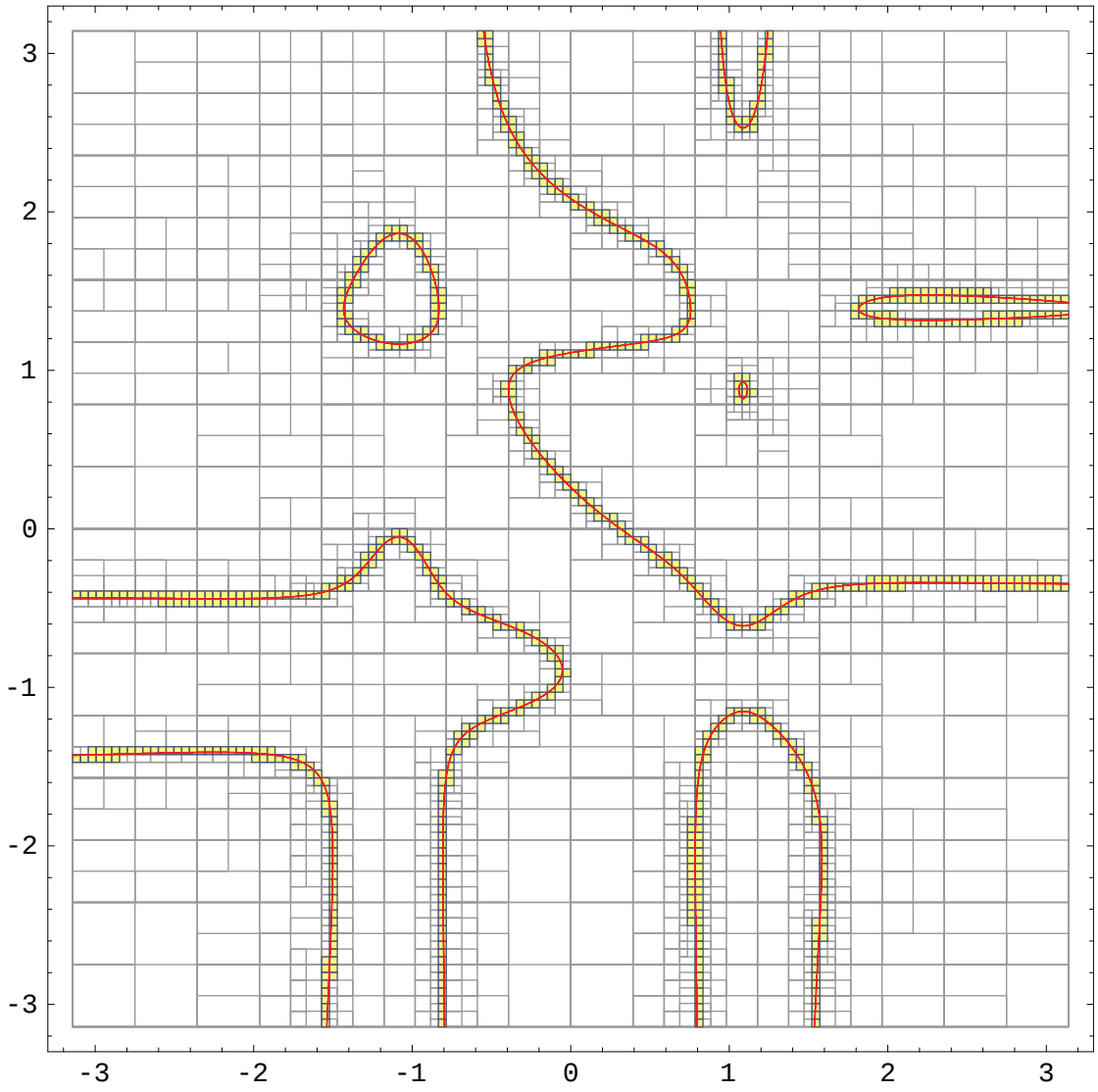


Figure 5.16: Plot of the solution regions produced by Divide and Conquer with Midpoint Taylor Forms. The solution box width is less than 2^{-4}. The algorithm found 788 solution regions in 4,951 iterations which took 63.016 seconds (Mathematica 5 time).

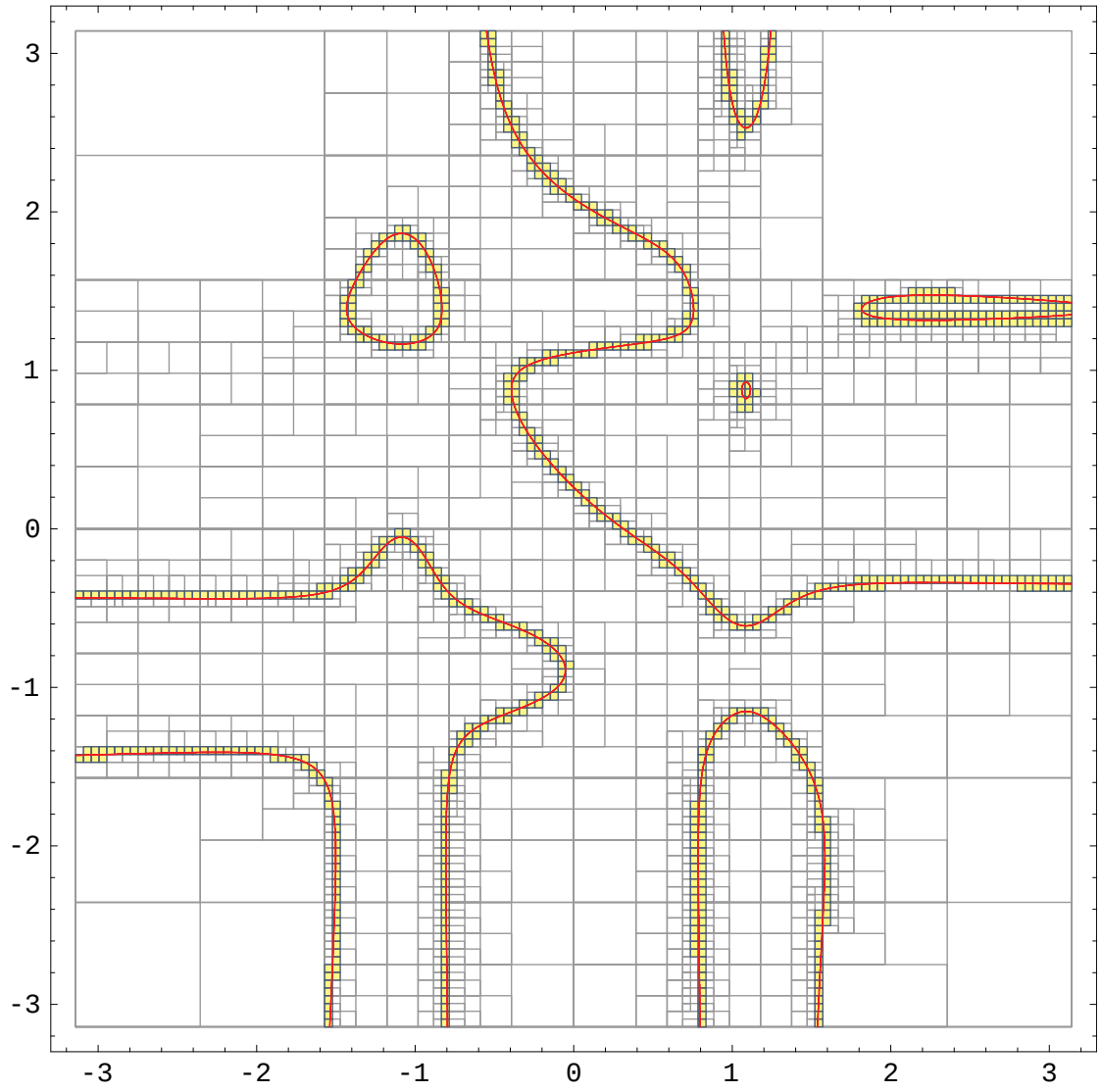


Figure 5.17: Plot of the solution regions produced by Divide and Conquer with Corner Taylor Forms. The solution box width is less than 2^{-4}. The algorithm found 807 solution regions in 4,841 iterations which took 61.312 seconds (Mathematica 5 time).

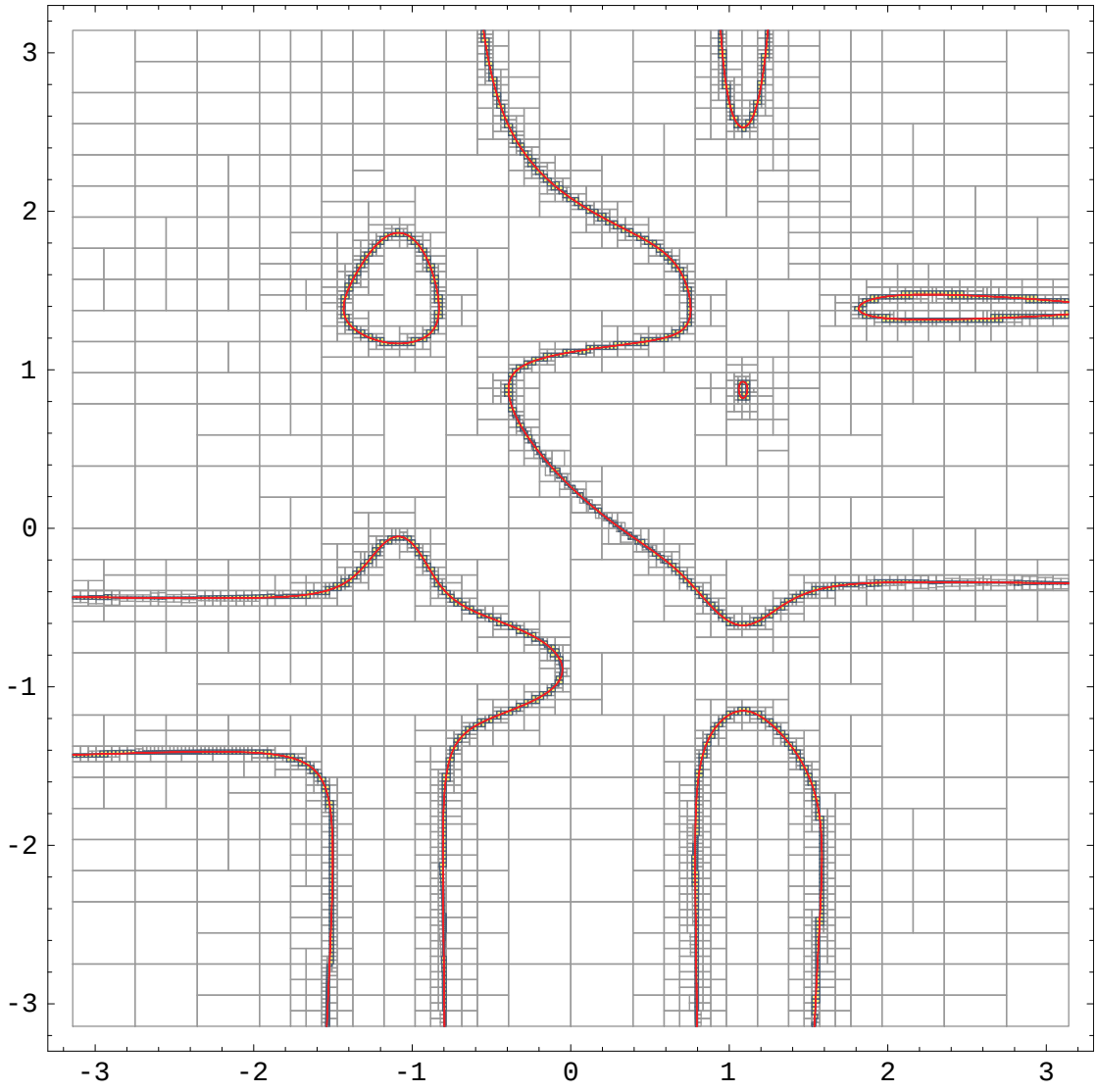


Figure 5.18: Plot of the solution regions produced by the Interval Newton method with Midpoint Taylor Forms. The solution box width is less than 2^{-5}. The algorithm found 1,557 solution regions in 5,067 iterations which took 67.656 seconds (Mathematica 5 time).

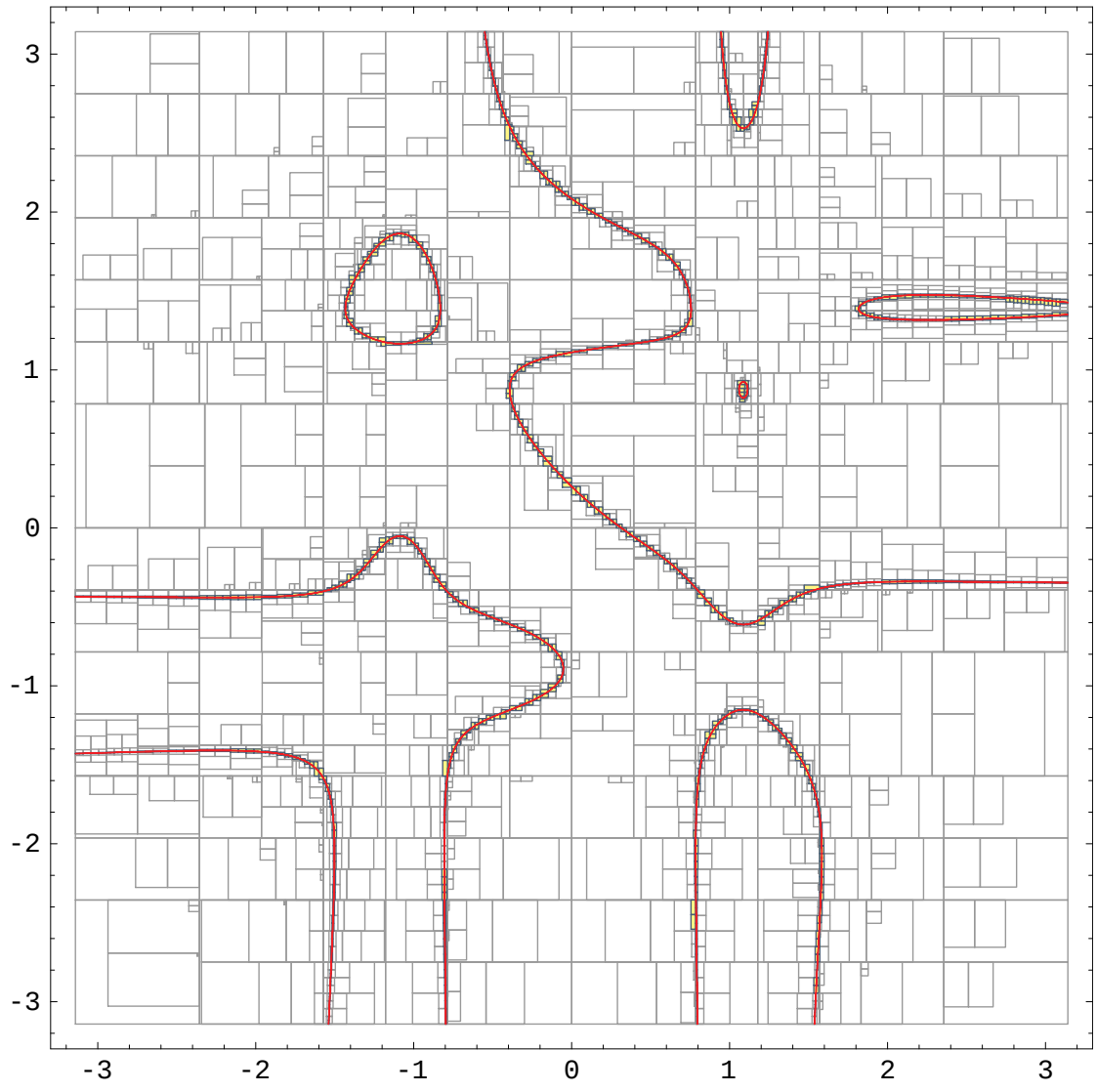


Figure 5.19: Plot of the solution regions produced by the RIN method with Midpoint Taylor Forms and binary subdivision (not using our special subdivision). The linearized solution width is less than 2^{-5}. The algorithm found 947 solution regions in 3,039 iterations which took 69.64 seconds (Mathematica 5 time).

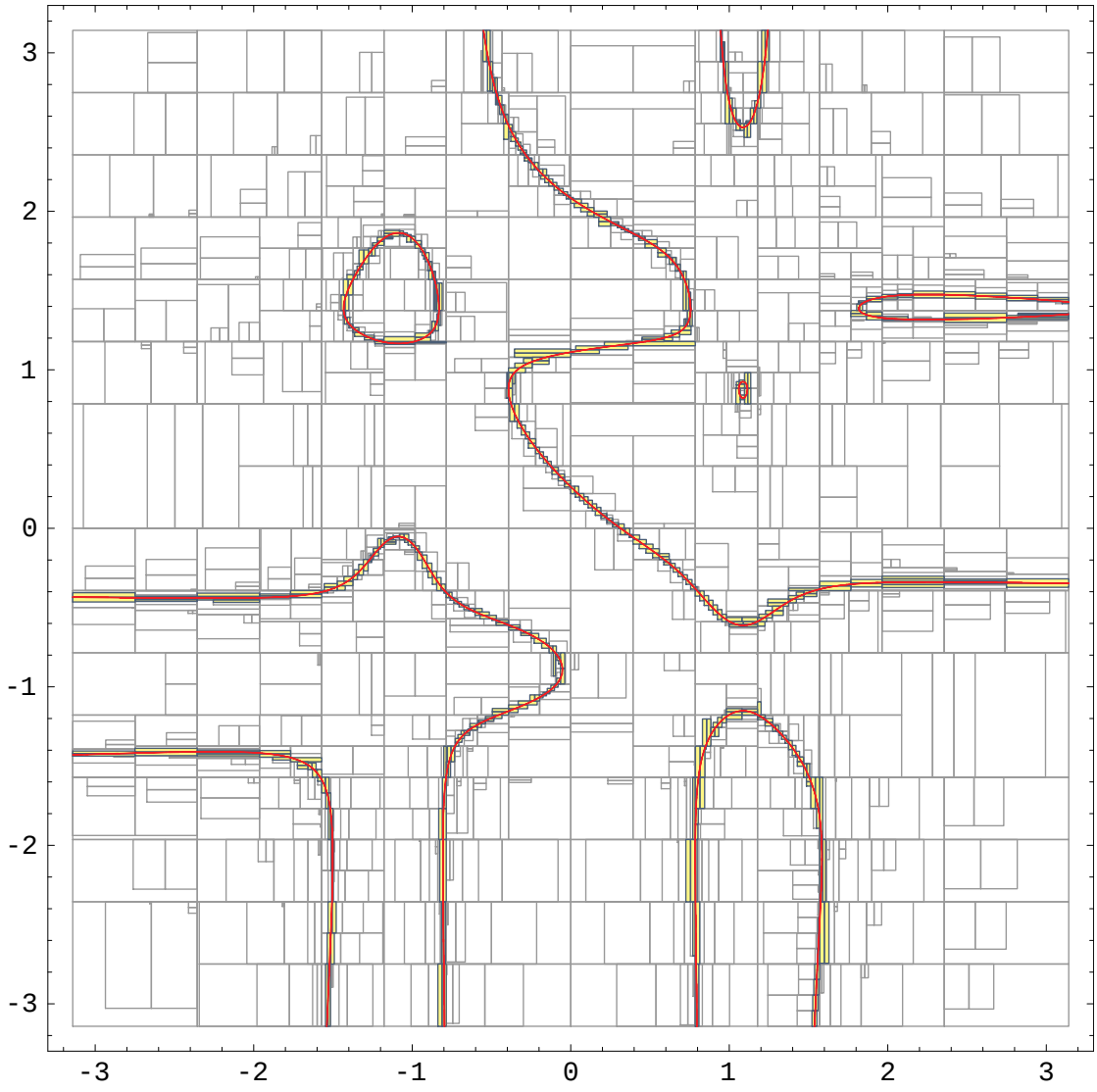


Figure 5.20: Plot of the solution regions produced by the RIN method with Midpoint Taylor Forms and RIN subdivision. The linearized solution width is less than 2^{-5}. The algorithm found 588 solution regions in 2,382 iterations which took 43.328 seconds (Mathematica 5 time).

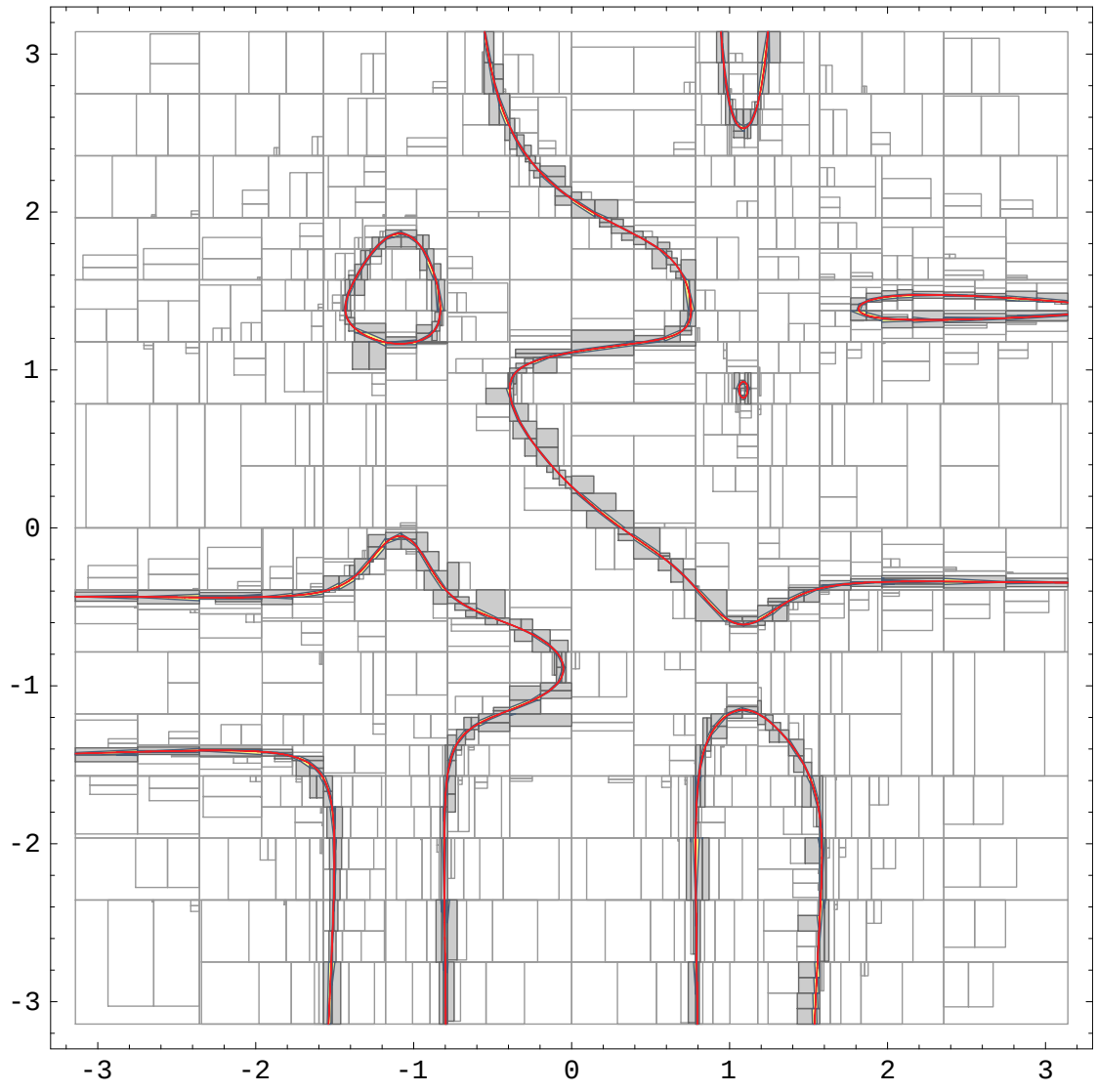


Figure 5.21: Plot of the solution regions produced by the RIN method with Midpoint Taylor Forms, RIN subdivision, and non-box solutions. The linearized solution width is less than 2^{-5}. The algorithm found 353 solution regions in 1,836 iterations which took 26.469 seconds (Mathematica 5 time).

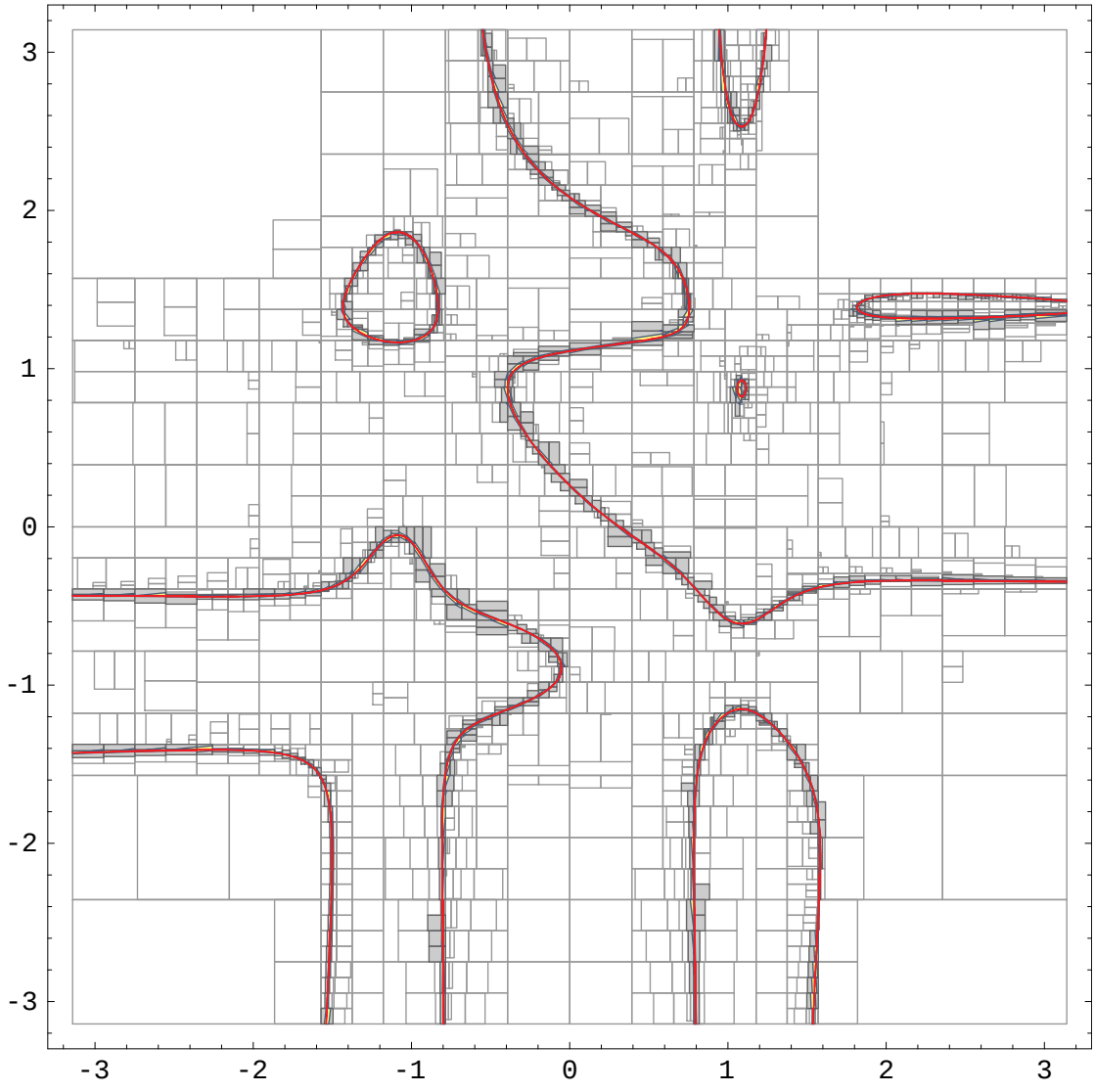


Figure 5.22: Plot of the solution regions produced by the RIN method with Corner Taylor Forms, RIN subdivision, and non-box solutions. The linearized solution width is less than 2^{-5}. The algorithm found 605 solution regions in 2,982 iterations which took 39.469 seconds (Mathematica 5 time).

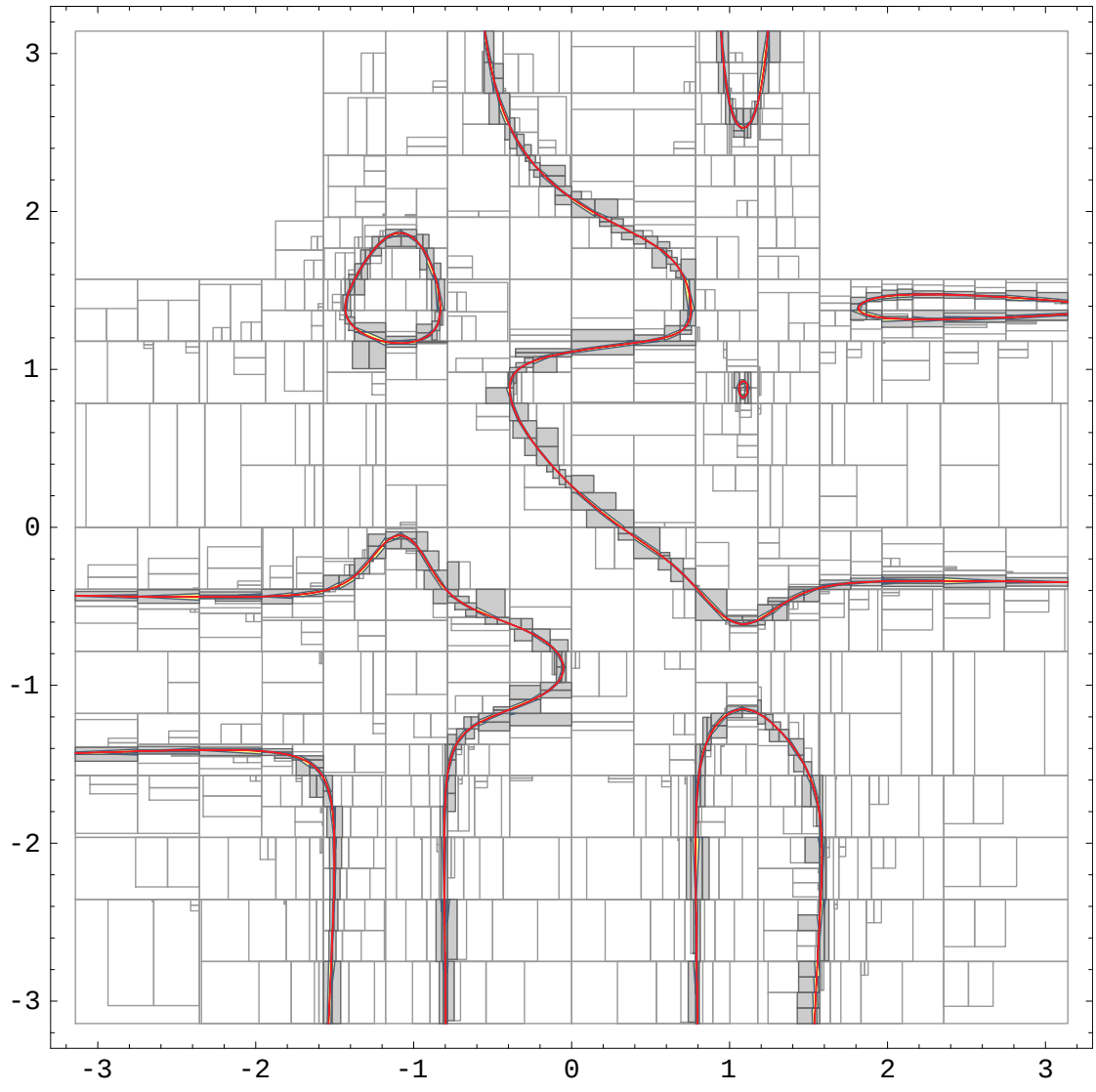


Figure 5.23: Plot of the solution regions produced by the RIN method with Corner and Midpoint Taylor Forms (switch from CTF to MTF when intervals have width less than 1), RIN subdivision, and non-box solutions. The linearized solution width is less than 2^{-5}. The algorithm found 353 solution regions in 1,704 iterations which took 25.197 seconds (Mathematica 5 time).

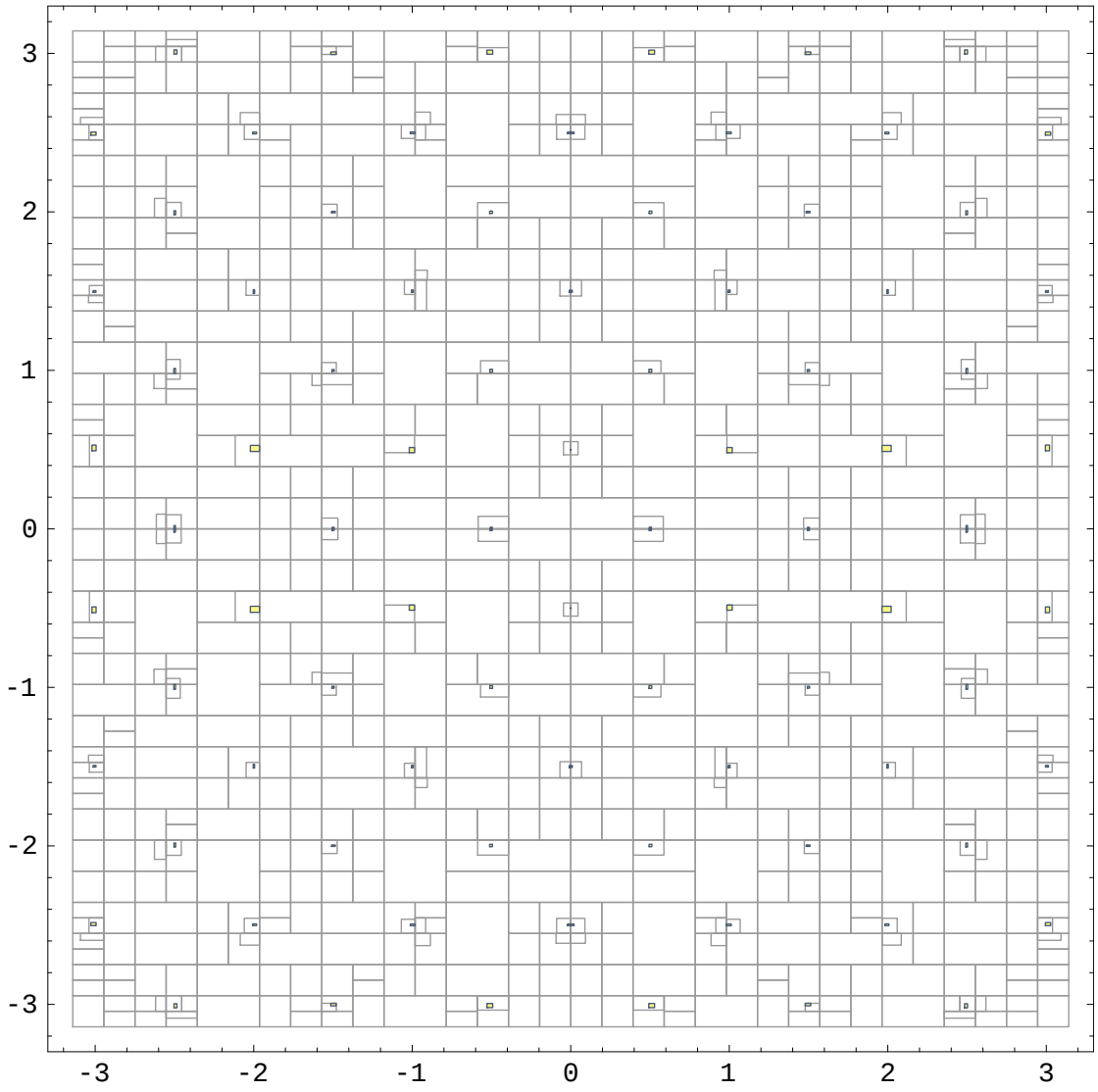


Figure 5.24: Interval Newton without tightening. Solutions of the system of polynomials 5.5.2 over $[-\pi,\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 1,939 iterations which took 72.8 seconds (Mathematica 5, P4@3.06GHz.)

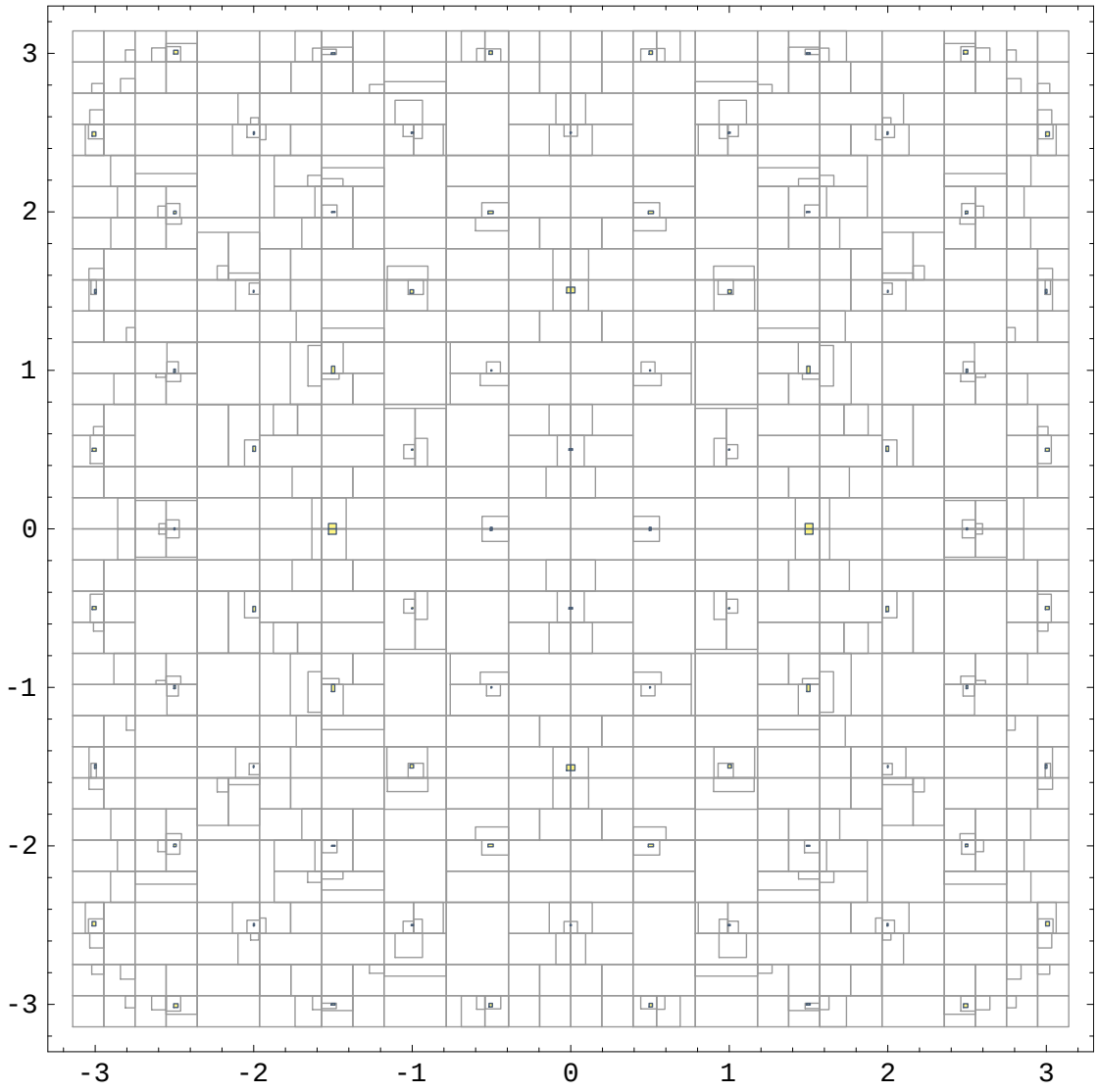


Figure 5.25: Interval Newton with tightening. Solutions of the system of polynomials 5.5.2 over $[-\pi,\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 1,651 iterations which took 63.5 seconds (Mathematica 5, P4@3.06GHz.)

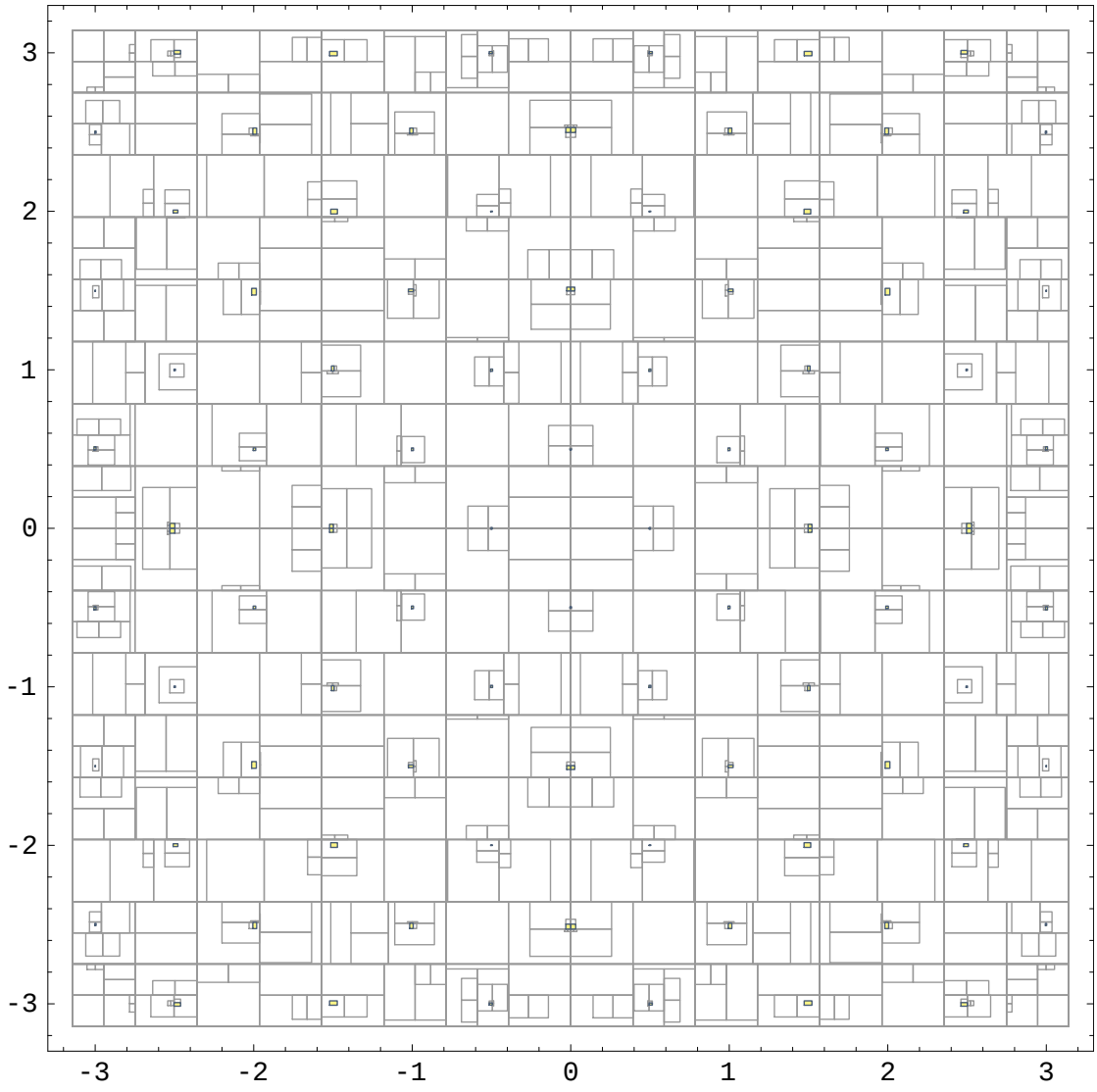


Figure 5.26: Remainder Interval Newton without tightening. Solutions of the system of polynomials 5.5.2 over $[-\pi,\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 1,403 iterations which took 46.3 seconds (Mathematica 5, P4@3.06GHz.)

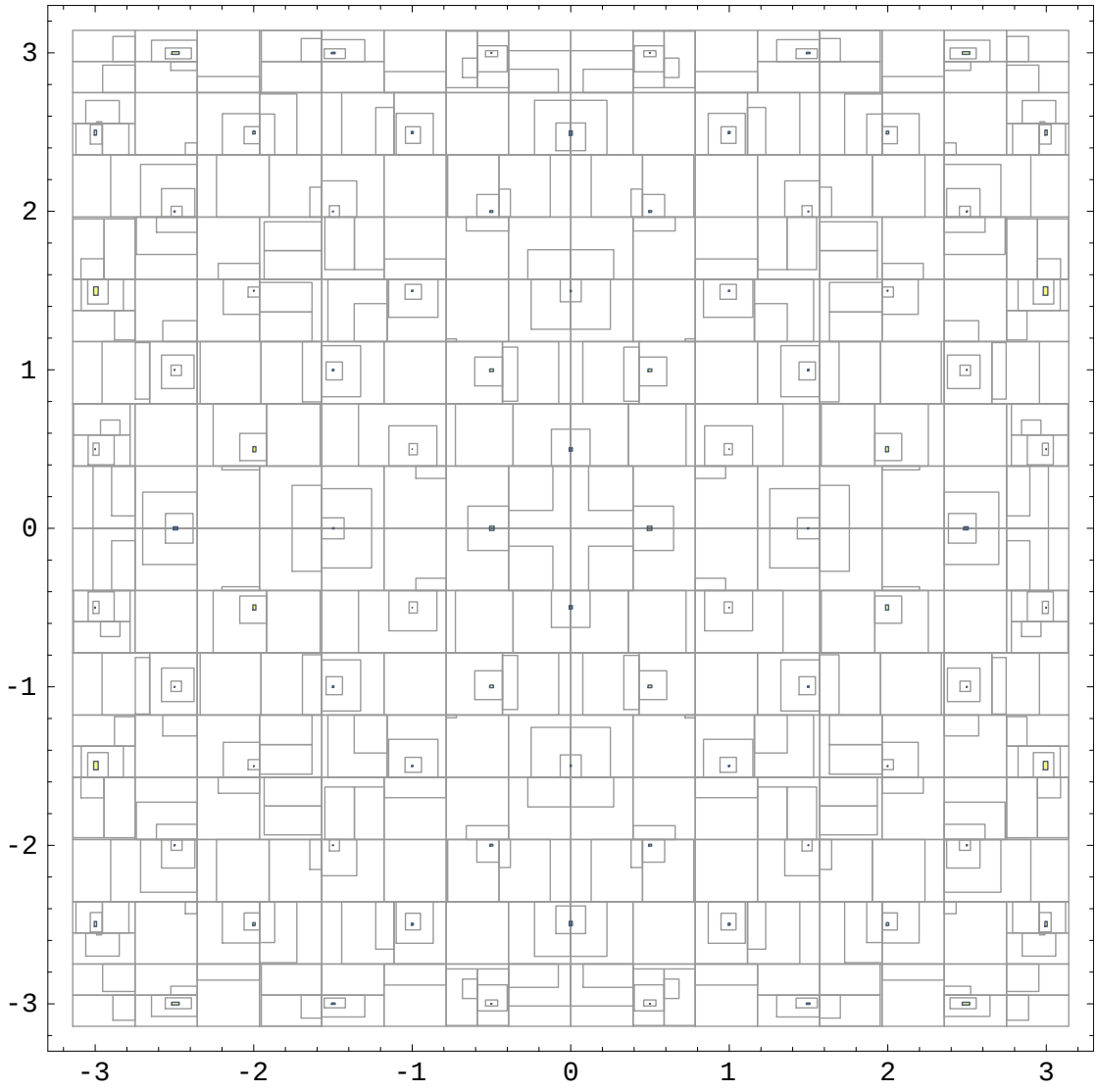


Figure 5.27: Remainder Interval Newton with tightening. Solutions of the system of polynomials 5.5.2 over $[-\pi,\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 1,027 iterations which took 34.6 seconds (Mathematica 5, P4@3.06GHz.)

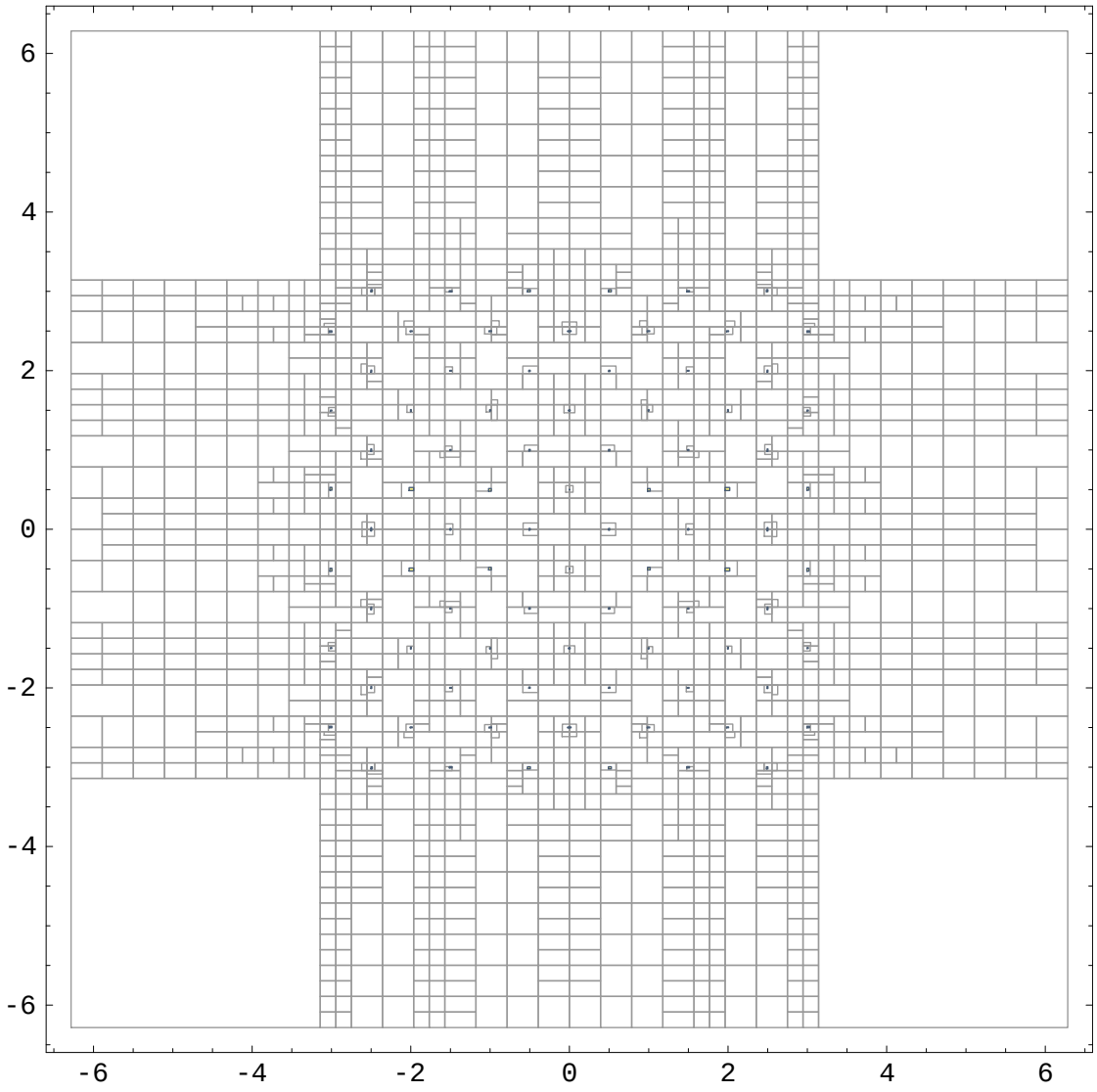


Figure 5.28: Interval Newton without tightening. Solutions of the system of polynomials 5.5.2 over $[-2\pi, 2\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 4,067 iterations which took 157.125 seconds (Mathematica 5, P4@3.06GHz.)

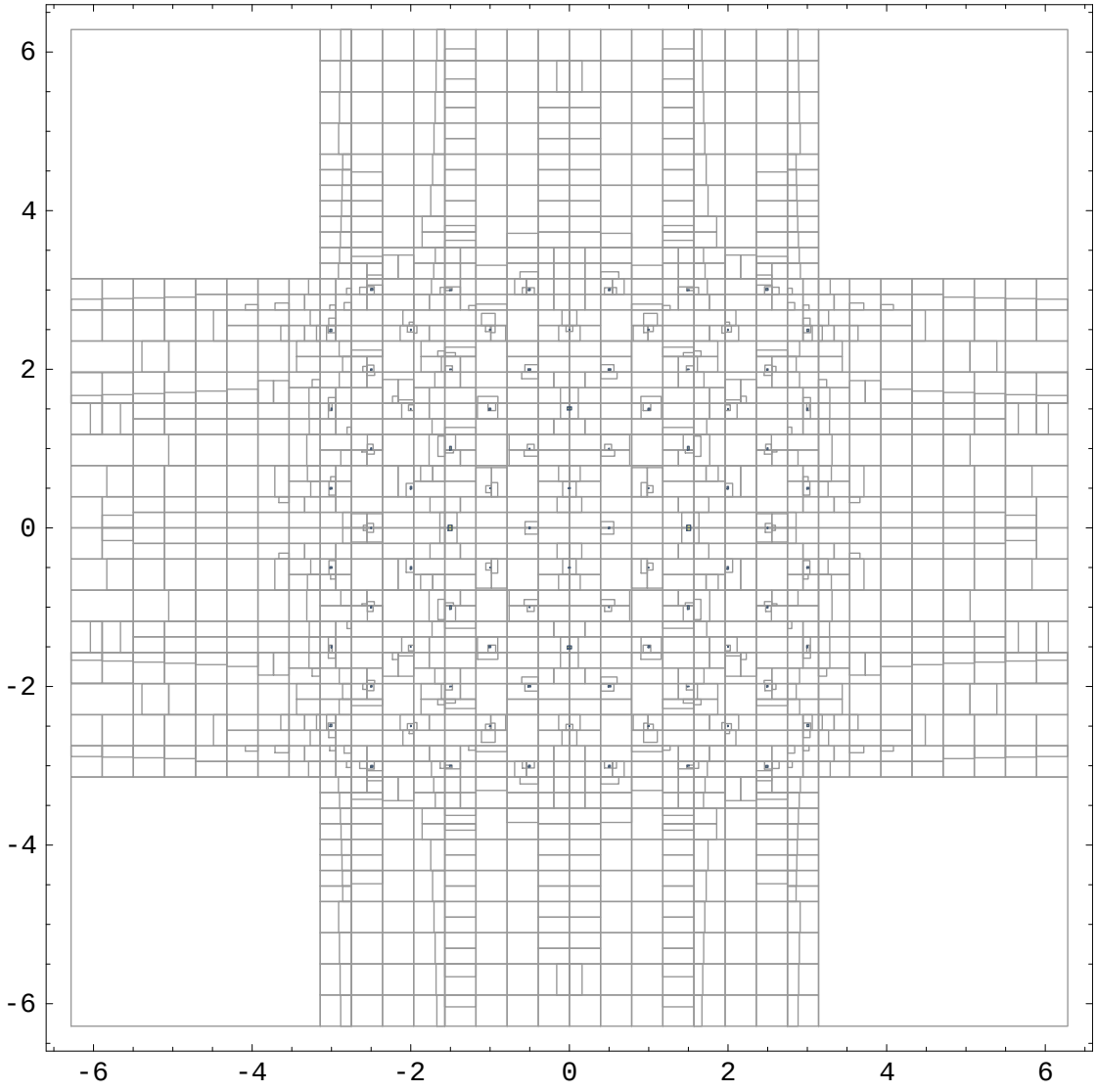


Figure 5.29: Interval Newton with tightening. Solutions of the system of polynomials 5.5.2 over $[-2\pi, 2\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 3,331 iterations which took 129.75 seconds (Mathematica 5, P4@3.06GHz.)

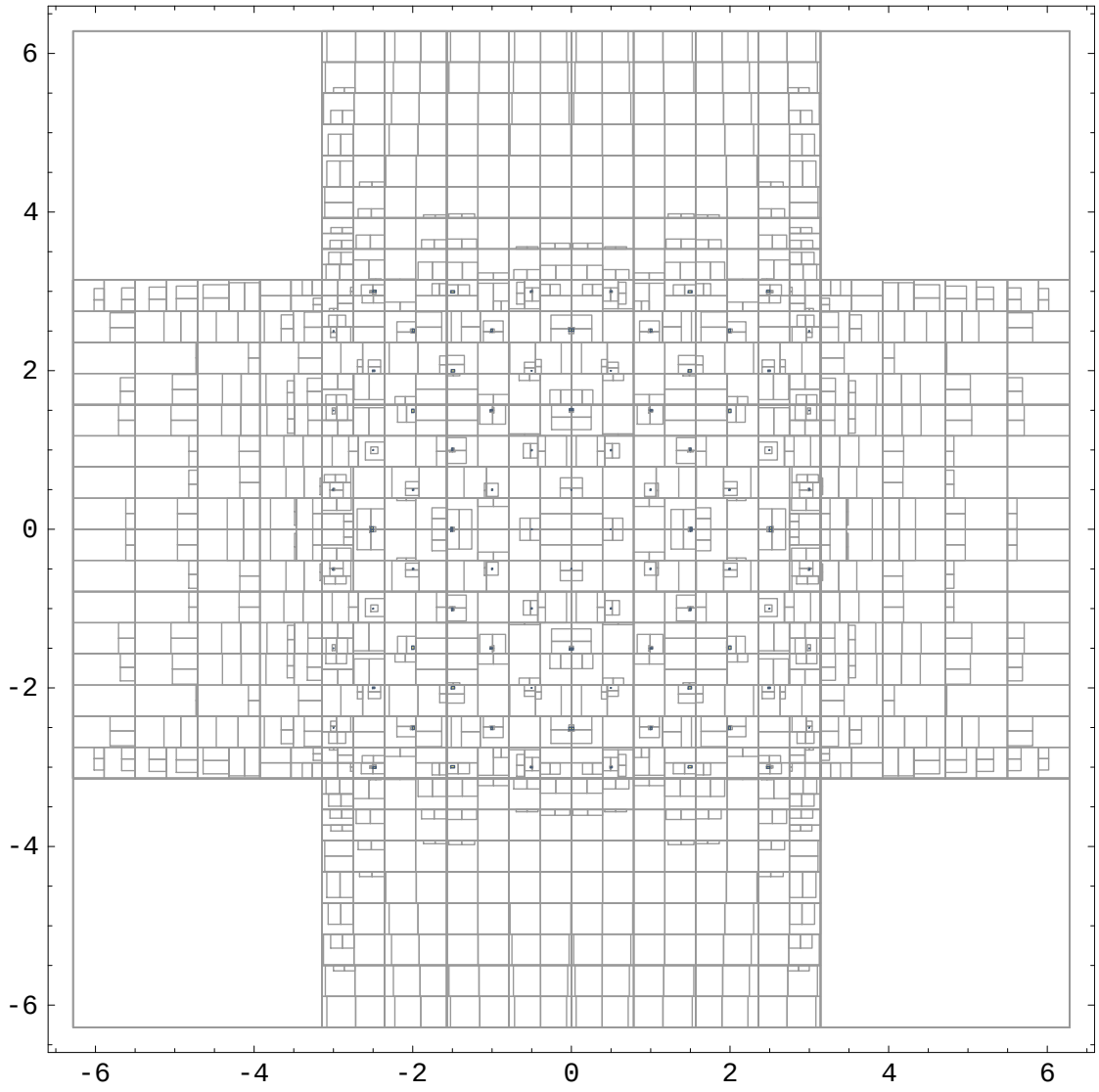


Figure 5.30: Remainder Interval Newton without tightening. Solutions of the system of polynomials 5.5.2 over $[-2\pi, 2\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 2,787 iterations which took 95.687 seconds (Mathematica 5, P4@3.06GHz.)

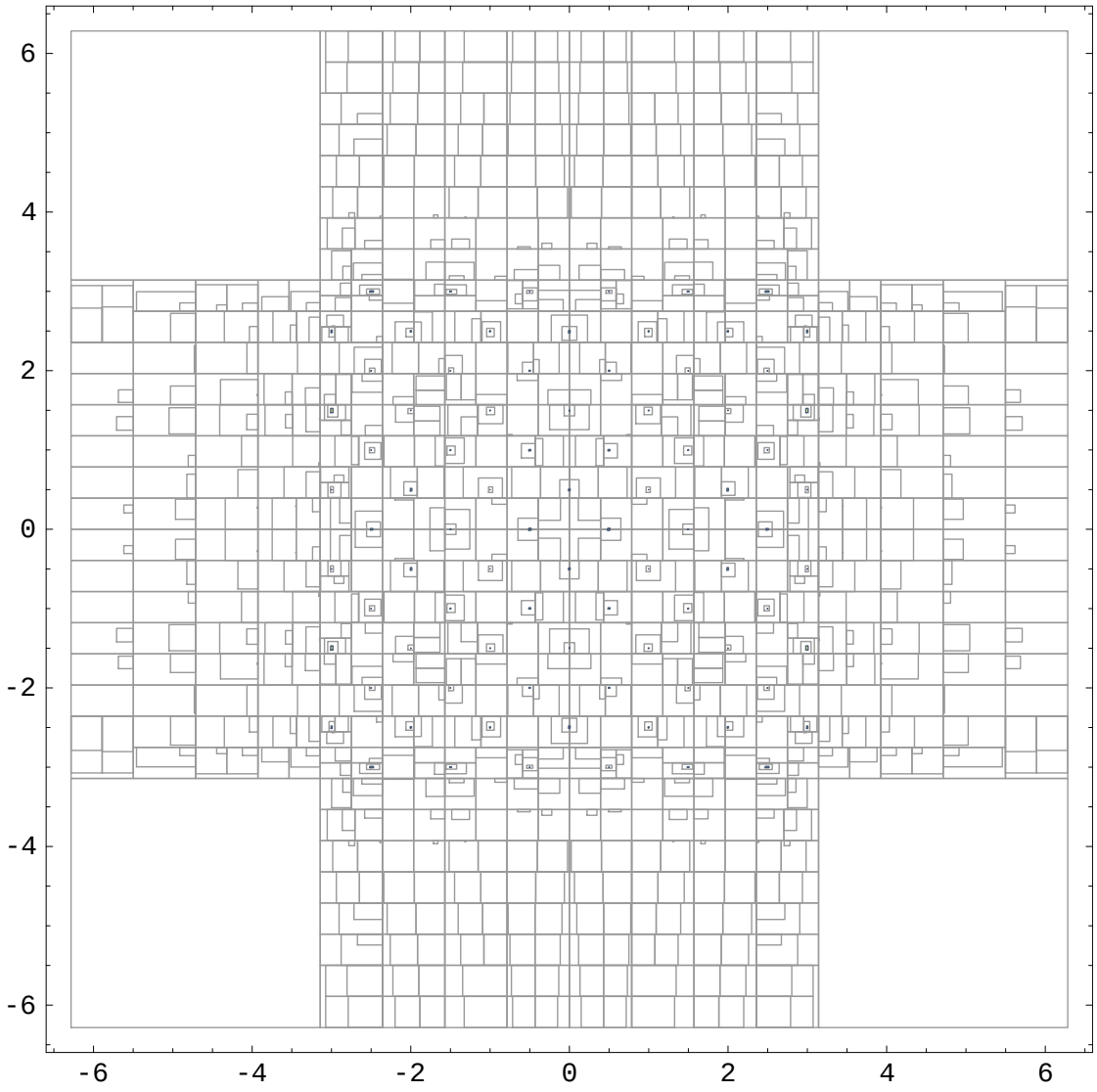


Figure 5.31: Remainder Interval Newton with tightening. Solutions of the system of polynomials 5.5.2 over $[-2\pi, 2\pi]^2$. Solution interval width is less than 2^{-4}. The algorithm found 96 solution regions in 2,099 iterations which took 71.891 seconds (Mathematica 5, P4@3.06GHz.)

Chapter 6

An Application: Beam Tracing for Implicit Surfaces

In this chapter we present an application of the methods introduced in the previous chapters to a computer graphics problem: reliable beam tracing of implicit surfaces.

6.1 Introduction

Over the past 20 years, ray tracing has become a popular computer graphics rendering method with many desirable characteristics. The ray tracing framework allows the creation of images with multiple light sources, hard and soft shadows, reflections, refractions and other important optical effects.

The ray tracing approach is based on the principles of geometric optics. It reduces to the problem of intersecting straight lines, or rays, with each of the objects in the scene.

Typically, several such rays are traced from the eye through each pixel on the screen. The first intersection point with an object in the scene is recorded. Rays connecting this intersection point with each of the light sources are traced to determine if the point is lit or is shadowed by other objects in the scene. The light intensity at the intersection point is then computed using one of several lighting models. Reflected and refracted child rays can be generated from this point and traced recursively, with their contributions added to the final result. The approach produces attractive images.

Like any method, however, ray tracing has its limitations. One source of problems is that ray

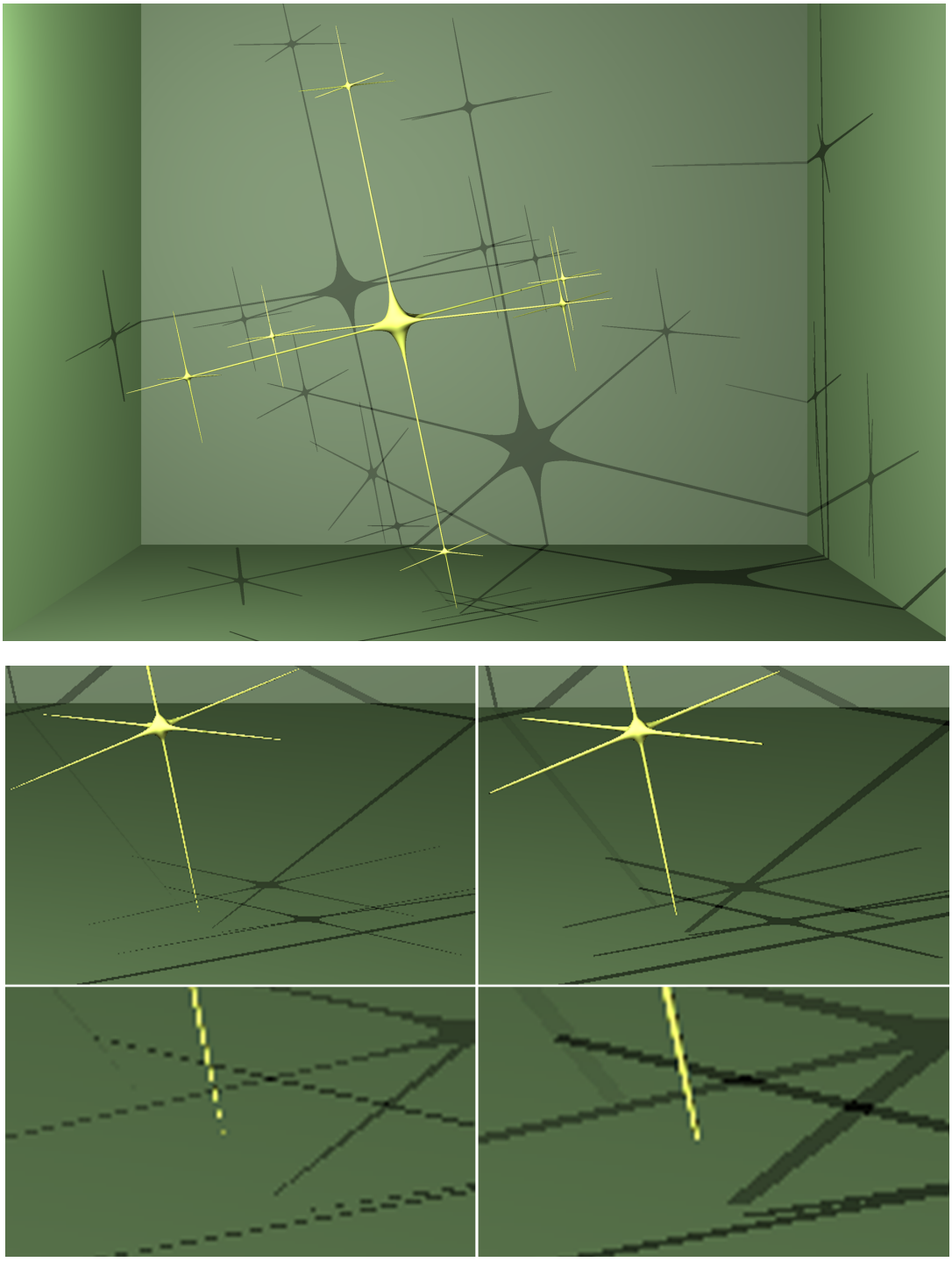


Figure 6.1: Rendering of a complex implicit model with thin, hair like features. *Top*, the whole scene. *Bottom*, detail views of one of the thin features of the surface. *Left*, ray traced images, above, antialiased and below, not antialiased; the rays sometimes miss the hair like features causing them and their shadows to appear discontinuous. *Right*, beam traced images; the thin features and the shadows they cast are always continuous and free of pixel dropouts.

tracing creates a point-sampling of an image function. We do not trace all of the mathematically possible rays (an infinite number) that go through the eye and reach the image plane, but only select a small finite subset to represent the image.

When the scenes contain objects with very thin features, thinner than the spacing between the rays we chose to sample, it is likely that we will miss some of the thin features. This is a well known limitation of any point sampling process. The resulting images will exhibit pixel dropouts, resulting in unpleasant visual artifacts, see figure 6.1. This effect does not fully anti-alias away, and the artifacts can be even more unpleasant if they occur in the frames of an animation, causing pixels to randomly flicker and distract the viewer.

The problem is made worse when the numerical methods used are not guaranteed to find the correct ray-object intersections.

To overcome some of these limitations, researchers have developed an approach which replaces the infinitely thin rays with thicker regions called *beams* ([Heckbert and Hanrahan 1984, Amanatides 1984]). Because the set of beams covers the image plane completely with no gaps, beam tracing avoids most of the sampling problems mentioned above. The geometric optics algorithm that uses beams has a very similar structure to the ray tracing algorithms.

We extend the original beam tracing approach and present a guaranteed interval method for intersecting beams with nonlinear implicit surfaces. By tracing beams instead of rays we cover the whole set of rays that go through the image plane and replace the sampling process with an averaging process. Beam tracing guarantees that the rendered images will be free of pixel dropouts, independently of the complexity of the scene.

In our current implementation, the intervals are *univariate*, with intervals just for the single variable of the ray parameter t which measures the distance from the eye.

Any of the previous interval root finding methods, such as conventional interval analysis, Interval Newton, centered forms, or affine arithmetic, can be used to perform beam tracing. The choice of method determines the overall efficiency of the beam tracer. Therefore, we use the RIN method with Corner Taylor Forms.

The RIN methods described here can also be used to perform basic ray tracing showing im-

proved performance over conventional interval methods and state of the art ray tracing algorithms for implicit surfaces, such as the well known LG-method.

An example of a particularly complex (very fine features) object rendered with our algorithm is shown in figure 6.1. For comparison, ray tracing the same image with the LG-method would have taken almost two years; our algorithm took less than six hours.

6.2 Previous Work

Beam tracing for polygonal objects was introduced by Heckbert and Hanrahan in 1984, see [Heckbert and Hanrahan 1984]. Simultaneously, Amanatides introduced a similar technique called *cone tracing*, see [Amanatides 1984]. Our algorithm extends these to intersect beams with any differentiable implicit surfaces.

We are not aware of any previous implementations of beam tracing for implicit surfaces. Beam tracing for polygonal models has been a subject of investigation by other researchers; see [Shinya et al. 1987, Watt 1990].

There are many algorithms for ray tracing implicit surfaces. We summarize a few that are most closely related to this work.

The algorithms are classified into two categories with respect to their ability to converge to the correct solution. The first category is comprised of algorithms that do not provide any guarantees, such as ray marching and straight applications of Newton-like methods, see 6.2.2. The second category is comprised of algorithms that guarantee the correct finite-precision ray surface intersection values, including the LG-method, interval analysis and affine analysis based methods, sphere tracing, and interpolatory methods, see 6.2.3.

6.2.1 Review: Ray/Implicit Surface Intersection in One Variable

Implicit surfaces are defined as the set of points in 3D where a scalar-valued function f of three variables takes on the value zero.

Mathematically, the surface is expressed as:

$$f : \mathbb{R}^3 \to \mathbb{R}, \quad f(\mathbf{x}) = 0 \tag{6.1}$$

A ray in three dimensions can be expressed in its parametric form

$$\mathbf{r} : \mathbb{R} \to \mathbb{R}^3, \quad \mathbf{r}(t) = (\mathbf{p} - \mathbf{c})t + \mathbf{c}. \tag{6.2}$$

where **c** is the 3D location vector of the camera and **p** is the 3D location vector of the pixel in world coordinates. The problem of determining intersection points between the ray **r** and the implicit surface $f(\mathbf{x}) = 0$ reduces to that of "plugging" the ray equation into the implicit equation, and solving for the smallest non-negative value of t that makes f be zero.

In other words, we are finding the smallest root of the univariate equation

$$(f \circ \mathbf{r})(t) = f((\mathbf{p} - \mathbf{c})t + \mathbf{c}) = 0 \tag{6.3}$$

that is closest to **c** and beyond **p**.

Very often these equations are nonlinear and do not have a closed form solution. A finite-precision numerical solution must be used in this case. It is important to be sure that the numerical solution method can always find the correct roots, within a satisfactory error tolerance.

6.2.2 Methods that Do Not Guarantee Solutions

The simplest method, *Ray Marching*, was introduced by Tuy and Tuy in 1984, see [Tuy and Tuy 1984]. It evaluates the implicit function at points along the ray, for successive values of t. The surface is detected when a change of sign occurs in equation 6.3. Ray marching is very general, only requiring a method for evaluating the value of the equation 6.3, the generating function at points in space, but may miss the first intersection or the whole surface entirely if the distance between the sampling points is too big. Reducing this distance improves accuracy at the cost of severely increasing the running time.

Another simple approach is to use *Newton's method* to find a root of equation 6.3. However, the convergence of Newton's method is notoriously dependent on the choice of starting point and it may converge to any of the solutions of equation 6.3 and not be able to locate the one we seek, or it may even diverge. There are other more stable relatives of Newton's method, such as *regula falsi*, but they require special initial conditions that are not easily achievable. (The LG-method provides the correct setting in which regula falsi can be used reliably.)

6.2.3 Methods that Guarantee Solutions Along a Ray

These are methods that reliably compute the solutions of the ray/surface equations. Note however, that this guarantee, even though desirable, does not prevent pixel dropouts from occurring in the rendered images. Dropped pixels can still occur because the guarantee only applies to the ray/surface intersection. If the ray "misses" the surface then the corresponding pixel will be colored with the background color. In the case of surfaces with fine detail, the rays can miss or hit in an inconsistent manner thus creating visually disturbing aliasing artifacts.

The first method that guarantees ray/surface intersections was described by Kalra and Barr in 1989, [Kalra and Barr 1989], and is know as the *LG-method*. It is still one of the standard algorithms for directly rendering implicit surfaces.

The LG-method is an interval analytic method in disguise. It requires the computation of guaranteed upper Lipschitz bounds: L, the upper bound of the Lipschitz constant of F in a region, and G, the upper bound of the Lipschitz constant of the gradient of F in a region. The L and G constants are used to robustly bound the function variation in those regions. The method uses binary subdivision to isolate the regions where only one solution is guaranteed to exist, followed by regula falsi to converge to that unique solution. The main problem with the LG-method is its lack of automatic generation of expressions for L and G — the user is required to provide the expressions of L and G for every primitive.

Interval analysis was first used to ray trace implicit surfaces by Mitchell in 1990, see [Mitchell 1990]. The method is similar to the LG-method, but it uses inclusion functions to bound the range of the ray/object functions without requiring the L and G constants. These

early applications of interval analysis suffered from the use of non-optimal inclusion functions which slowed down convergence considerably.

Affine arithmetic has been proposed as a replacement to conventional interval analysis, see [Stolfi and de Figueiredo 1997]. Applications to ray tracing implicit surfaces were shown in [de Cusatis Jr. et al. 1999]. The reported improvements over conventional interval methods have not been uniform — recent investigations show that affine arithmetic is a special case of the centered form.

We show that the combination of CTF inclusion functions and RIN solution methods can significantly improve the performance of interval based ray tracers, especially when rendering "difficult" surfaces.

Sphere tracing is a variant of the ray marching algorithm, see [Hart et al. 1989, Hart and DeFanti 1991, Hart 1993b]. It uses a signed distance function to the implicit surface to guarantee that the steps taken along the ray are always small enough not to penetrate the surface. Like the LG-method, this guarantee is achieved through the computation of Lipschitz bounds. It can be used with any surface for which the user can provide a signed distance function and the normal, including fractals and some surfaces without continuous derivatives.

Closed form solutions to equation 6.3 are desirable but usually unavailable. One approach is to approximate the function $F \circ \mathbf{r}$ with a simpler function $(F \circ \mathbf{r})^*$ for which closed form solutions exist. One such method has been introduced by Sherstyuk in 1998, see [Sherstyuk 1998]. The method uses interpolatory approximation with Hermite polynomials of 3rd degree and closed form solutions to achieve speedup. The accuracy of the rendered images can be somewhat limited by the accuracy of this approximation. Error can be reduced by increasing the number of Hermite patches used by the approximation, but no automatic method is provided. The algorithm achieves an approximate speedup factor of 3 when compared to the LG-method. Extending this algorithm to beam tracing requires the ability to compute with 4D Hermite approximations.

6.2.4 Methods that Guarantee Solutions Inside a Pixel

To completely and reliably eliminate all pixel dropout problems one needs to search for surface intersections for the set of all possible rays defined within the span of a pixel. This requirement can be accomplished through the use of beam tracing. Beam tracing for implicit surfaces has not been explored to date. In this paper we present a beam tracing method based on higher order interval analysis.

Although all variants of interval methods can be converted to perform beam tracing, the conventional methods may be unbearably slow. Higher order interval analysis is orders of magnitude faster than binary search based methods, and makes the approach much more tractable.

6.3 Beam Tracing Implicit Surfaces

6.3.1 Beams

Conventional rays are directed 3D lines defined by two points: the camera position (ray-origin) **c**, and the pixel position (ray-intercept) **p**:

$$\mathbf{r} : \mathbb{R} \rightarrow \mathbb{R}^3,\ \mathbf{r}(t) = (\mathbf{p} - \mathbf{c})t + \mathbf{c}. \tag{6.4}$$

Beams are generalized bundles of rays defined by two 3D intervals: A 3D interval for the camera position (beam-origin) $\underline{\overline{\mathbf{c}}}$, and a 3D interval for the pixel position (beam-intercept) $\underline{\overline{\mathbf{p}}}$:

$$\underline{\overline{\mathbf{b}}} : \mathbb{IR} \rightarrow \mathbb{IR}^3,\ \underline{\overline{\mathbf{b}}}(t) = (\mathbf{p} - \mathbf{c})t + \mathbf{c}. \tag{6.5}$$

Beams are classified in several categories. *Camera beams* have the origin in the camera location and the intercept on the image plane. *Shadow beams* have the origin at a light source and the intercept at a previously computed beam-surface intersection. *Reflected and refracted beams* have their origins at a beam-surface intersection and their intercepts are computed according to the rules of geometrical optics as a function of the surface normals at the origin and the direction interval of the incoming beam.

6.3.2 Beam-Surface Intersection

The beam casting equation is a generalization of the ray casting equation 6.3:

$$\left(\mathcal{I}(f) \circ \underline{\overline{\mathbf{b}}}\right)(t) = \mathcal{I}(f)\left((\underline{\overline{\mathbf{p}}} - \underline{\overline{\mathbf{c}}})t + \underline{\overline{\mathbf{c}}}\right) = 0. \tag{6.6}$$

The beam parameter t is restricted to an interval $\underline{\overline{t}}$ that depends on the beam type. For camera beams, the interval used is $\underline{\overline{t}} = [1, \infty]$, while shadow beams use an interval $\underline{\overline{t}} = [0, 1]$, and reflected and refracted beams use the interval $\underline{\overline{t}} = [0, \infty]$.

The RIN root finding algorithm described in the previous section is used to locate the "first" root $\underline{\overline{t}}_s$ of the beam casting equation 6.6. If $\underline{\overline{t}}_s$ is empty, no solutions exist. Otherwise the beam-surface intersection region $\underline{\overline{\mathbf{r}}}_s$ is computed by the equation:

$$\underline{\overline{\mathbf{r}}}_s = \underline{\overline{\mathbf{b}}}(\underline{\overline{t}}_s) = \underline{\overline{\mathbf{p}}}\ \underline{\overline{t}}_s + (1 - \underline{\overline{t}}_s)\,\underline{\overline{\mathbf{c}}}. \tag{6.7}$$

Beams are rotated into the frame of the object prior to computing the intersection to keep the expressions simple.

6.3.3 Computing the Illumination

The range of normals $\underline{\overline{\mathbf{n}}}_s$ to the surface in the region $\underline{\overline{\mathbf{r}}}_s$ is computed by dividing the range of the gradient $\underline{\overline{\mathbf{g}}}_s$ by the norm of the average gradient of f within $\underline{\overline{\mathbf{r}}}_s$, using the following formulas:

$$\underline{\overline{\mathbf{g}}}_s = \mathcal{I}(\nabla f)(\underline{\overline{\mathbf{r}}}_s) \tag{6.8}$$

$$\underline{\overline{\mathbf{n}}}_s = \frac{\underline{\overline{\mathbf{g}}}_s}{\|\mathrm{m}\left(\underline{\overline{\mathbf{g}}}_s\right)\|}. \tag{6.9}$$

Note that not all the vector elements in $\underline{\overline{\mathbf{n}}}_s$ are of unit length since it is impossible to represent a set of unit length vectors by simple intervals (boxes) in Euclidean space.

The illumination is computed in the form of interval values for each of the R, G, B components using interval versions of any of the standard illumination models and the interval vector $\underline{\overline{\mathbf{n}}}_s$. The contributions of all the beams in the tree are added and the average values are assigned

to the corresponding pixels.

6.3.4 Generating Reflected/Refracted Beams

Outgoing reflected/refracted beams are produced by plugging in the range of the normals at the beam-surface intersection region $\underline{\bar{\mathbf{r}}}_s$ into the interval form of the classic formulas of geometrical optics. The resulting beams can be very wide if the range of the normals is big. In this case the range of the normal is subdivided into a number of smaller intervals $\underline{\bar{\mathbf{n}}}_s^i$ and several outgoing beams are generated.

In cases where the user is not concerned with guaranteed reflections the outgoing beams can be narrowed or even be completely replaced with rays. We have opted for this optimization in our implementation, using beams for the camera and shadow rays, and conventional rays for the reflected rays. The reflected rays are computed with respect to the midpoint of the intersection region, $\mathrm{m}(\underline{\bar{\mathbf{r}}}_s)$, and the average normal $\mathrm{m}(\underline{\bar{\mathbf{n}}}_s)$.

6.3.5 Making Beam Tracing Work

There are a number of issues that should be considered when implementing our algorithm.

Rotations, a very common operation in all ray tracing algorithms, should only be applied once to expressions and objects that are defined using intervals. The reason for this is that each application of a rotation matrix to a higher (2 or more) dimensional interval can increase its size, sometimes by up to 100 percent, see figure 6.2. If a series of transformations need to be applied to such a higher dimensional interval it is imperative that the transformations be first composed together and the resultant applied only once. Some increase in size is unavoidable, for example when rotating the beams into the frame of the objects, thus artificially widening the beams. This effect is visible in the final images, as shadows are sometimes thicker than one might expect, see figure 6.1 bottom right. Such artifacts can be reduced by subdividing the beams appropriately, at the expense of increased computation.

It is important to use high quality inclusion functions - such as Taylor Forms - not only to speed up convergence but also to ensure the quality of the rendered images is high. As shown in

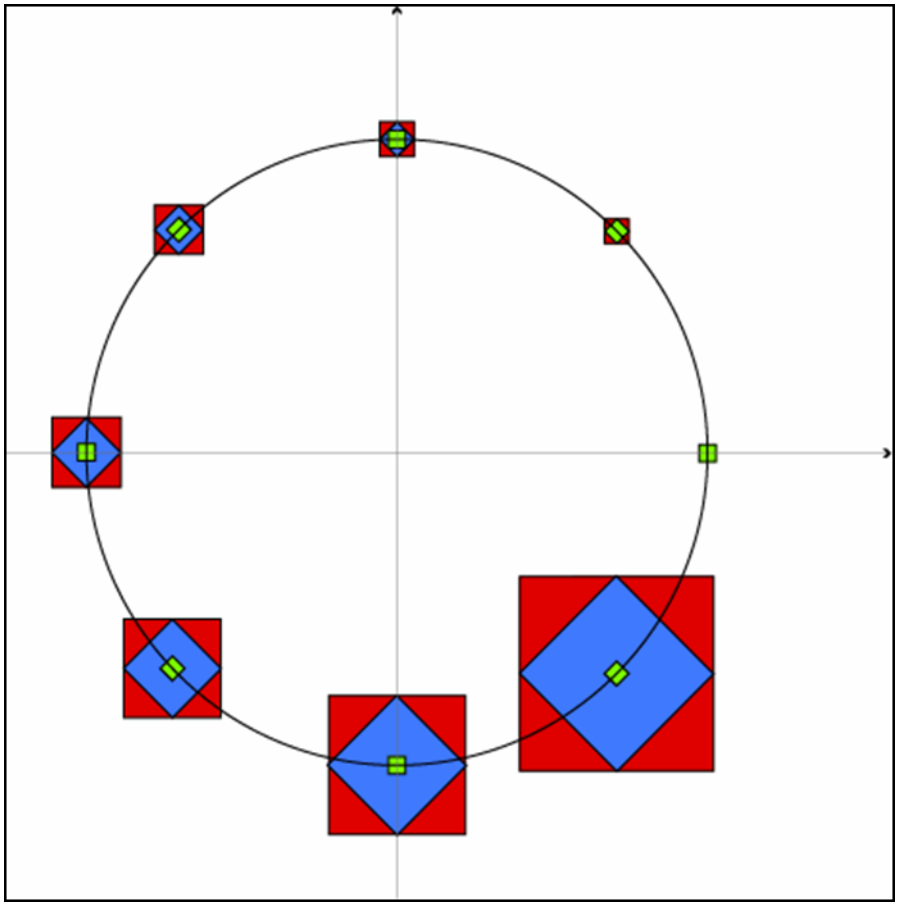


Figure 6.2: Applying multiple rotation transformations to a region can artificially increase its size. This artifact is known as the *wrapping effect*.

the previous chapters, too much excess width can produce solution sets with bad quality factors. When ray tracing a surface we pick only one of the solution boxes found along a ray; due to the omnipresent excess width, a single infinite precision solution may be covered by some number of adjacent boxes. How far the chosen box is from the exact solution depends on the quality factor, which in turn depends on the local amount of excess width. Therefore, use of inclusion functions with non-optimal excess width can result in the surfaces and their shadows becoming visibly distorted. In fact, use of the natural extension often produces images that are unrecognizable.

Finally, it is well known that interval arithmetic is correct and guaranteed only when the appropriate rounding modes are used in its implementation. The IEEE floating point standards guarantee that the floating point units on most current microprocessors implement the correct

rounding modes. These should always be used if any guaranteed results are to be expected. Frequently switching rounding modes, however, can be an expensive operation. There are techniques for computing the correct upper and lower bounds without ever changing rounding modes. In fact, when the fastest rendering time is desired, one may drop the use of directed rounding altogether. Of course, in doing so one loses the absolute guaranteed nature of interval analysis — however, the probability of pixel dropouts occurring in the rendered images is still very low.

6.4 Results and Conclusions

We implemented the beam tracing algorithm and the LG method in the same ray tracing program. This approach lets us compare the beam performance to the LG-method for computing beam-surface and ray-surface intersections.

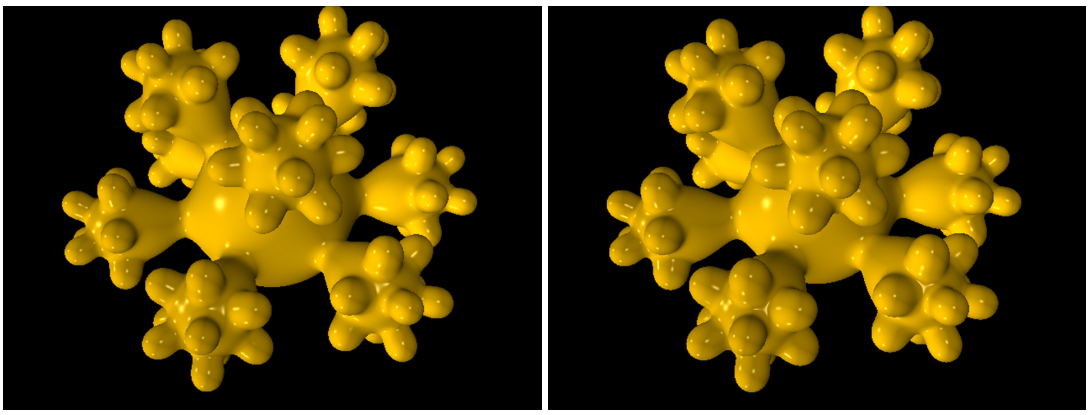


Figure 6.3: Rendering of a blobby flake. The model is comprised of 91 blended elliptic blobby primitives. *Left*, Gaussian blobbies. *Right*, polynomial blobbies.

For thin objects, the beams produced significantly improved results. Not only were pixel dropouts eliminated when we rendered with beams, as shown in figure 6.1, but the much-improved rate of convergence of the second order interval solver caused our algorithm to beat the rendering time of the LG-method by many orders of magnitude. This is expected to take place whenever rendering models with thin features and high curvature, such as the one shown in figure 6.4. The model is comprised of 76 blended elliptic Gaussian blobbies, 4 implicit planes, and 3 point light sources. Our method was able to ray trace the model in less than 20 hours, and beam

trace it in less than 50 hours, at a resolution of 2,880 by 1,944 with one ray/beam per pixel. We were not patient enough to render the same image using the LG-method, and rightly so, since we estimated it would have taken approximately 6 years to complete. We have done this estimation by rendering an 18 by 12 image of the model, which took 36 minutes, and multiplying by the ratio in resolutions (to simulate a 2,880 by 1,944 image). Ray tracing the same 18 by 12 image with our algorithm took less than a second.

For models with larger features and lower curvature, such as those shown in figure 6.4, the savings were not as significant. Times with beams were roughly equivalent to those of the LG-method. This is true because for those types of models the LG-method is able to find intervals that contain only one solution very quickly, at which point it switches to the regula falsi solver which has quadratic convergence. However, the beam tracing program is more robust near the boundaries and silhouette edges of the objects and is still much faster than conventional interval analysis.

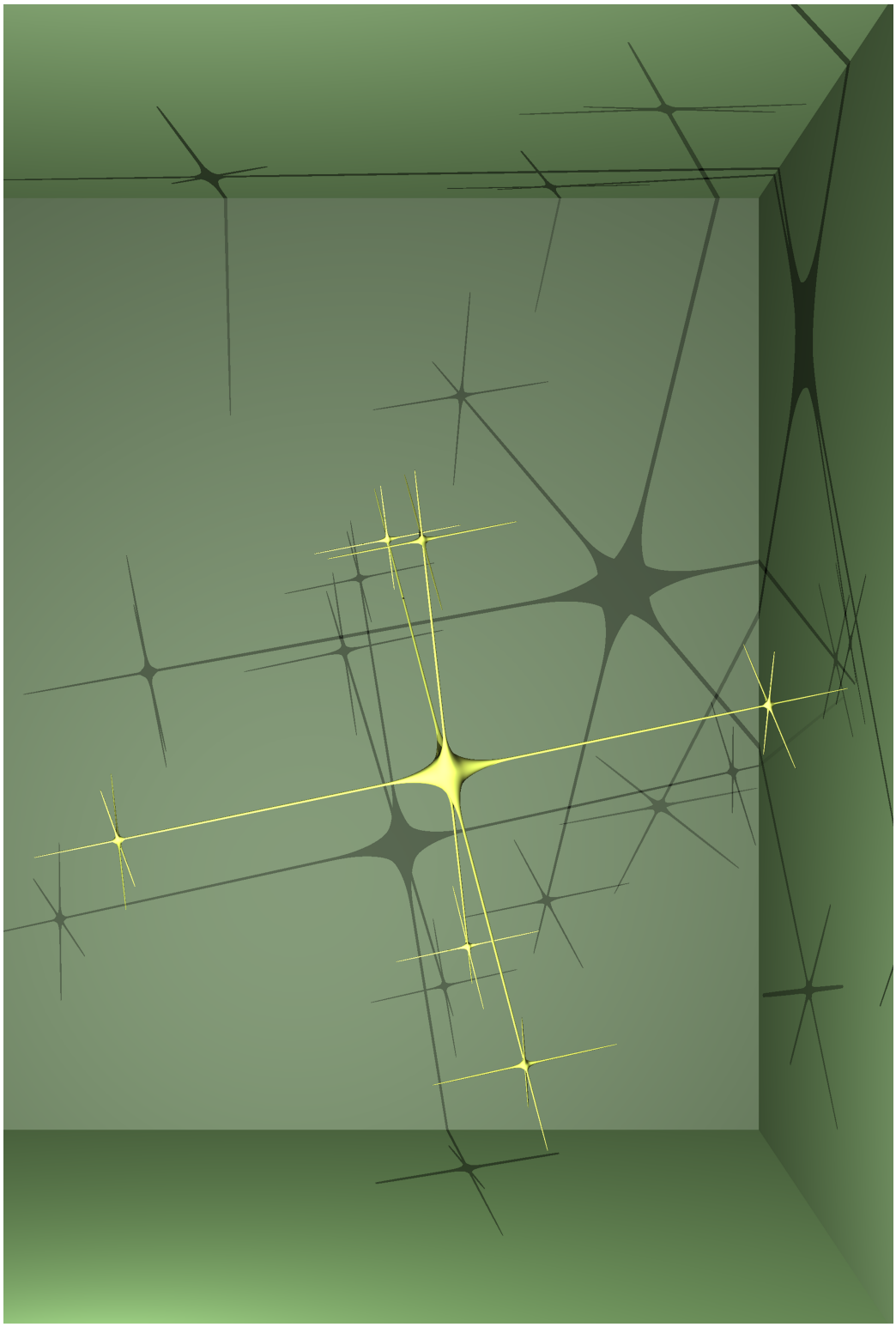


Figure 6.4: Rendering of a very complex implicit model with thin, hair like features. The model is composed of 76 super-thin Gaussian blobbies.

These results are encouraging and we envision applications of the method to render geometrically-modeled hair and other highly curved surfaces. The absence of pixel dropouts will be especially useful when making animations of these models.

6.5 Future Work

Replacing the current univariate interval solver with a multivariate version should improve both efficiency and quality by enabling adaptive spatial and temporal antialiasing. The multivariate framework could also be used to render caustics and simulate various wave phenomena, such as wavelength dependent scattering.

Chapter 7

Conclusion

Interval analysis is not as inefficient as some believe it to be. However, as the examples in this thesis show, efficient interval analysis requires consistent use of the most sophisticated methods available.

In this thesis we presented two new advancements in verified scientific computing using interval analysis that offer considerably improved efficiency:

1. **The Corner Taylor Form (CTF) interval extension.** We introduced the CTF, the first interval extension for multivariate polynomials that guarantees smaller excess width than the natural extension on any input interval, large or small, and has quadratic or better inclusion order. To help with the proofs we introduced the concept of *Posynomial Decomposition (PD)*. Using PD we developed simple and elegant proofs showing the CTF is isotonic and has quadratic or better (local) inclusion convergence order. We also developed closed form methods for computing the exact local order of convergence as well as the magnitude of excess width reduction the CTF produces over the natural extension. We presented practical examples and compared the CRF with other inclusion function types.

2. **The Remainder Interval Newton (RIN) method.** We also introduced the RIN method, which uses first order Taylor Models (instead of the Mean Value Theorem) to linearize nonlinear equations and systems. We showed that this linearization has many advantages, making RIN significantly more efficient than conventional Interval Newton (IN). In particular, for single multivariate equations, we introduced a new subdivision method based

on the RIN linearization. For this case, we showed that RIN requires only order of the square root as many solution regions as IN does. For square systems, we showed that RIN is able to isolate solutions faster than IN. For both types of problems, we presented examples where RIN methods realized savings in both time and memory for a sizable overall improvement.

As an application to computer graphics, we presented a novel algorithm for *Beam Tracing Implicit Surfaces.* We showed that beam tracing eliminates some of the shortcoming of conventional ray tracing, particularly the problem of *dropped* pixels due to its inherent sampling nature. We also showed that use of RIN can reduce running times significantly, both in the beam tracing and the conventional ray tracing settings.

Bibliography

[Alander 1985] Alander, J., 1985. “On interval arithmetic range approximation methods of polynomials and rational functions,” *Computers and Graphics*, 9(4):365–372.

[Alefeld and Herzberger 1983] Alefeld, G. and Herzberger, J. 1983. *Introduction to Interval Computations*. Academic Press Inc., New York, USA. Transl. by J. Rokne from the original German ‘Einführung in die Intervallrechnung’. 7

[Amanatides 1984] Amanatides, J., 1984. “Ray tracing with cones,” *Computer Graphics*, 18(3):129–135. 136, 137

[Bao and Rokne 1988] Bao, P. G. and Rokne, J. G., 1988. “Low complexity *k*-dimensional Taylor forms,” *Applied Mathematics and Computation*, 27(3 (part I)):265–280.

[Barth et al. 1994] Barth, W., Lieger, R., and Schindler, M., 1994. “Ray tracing general parametric surfaces using interval arithmetic,” *Visual Computer*, 10(7):363–371.

[Baumann 1988] Baumann, E., 1988. “Optimal centered forms,” *BIT*, 28(1):80–87. 40

[Berz and Hoefkens 2001] Berz, M. and Hoefkens, J., 2001. “Verified high-order inversion of functional dependencies and interval newton methods,” *Reliable Computing*, 7(5):379–398. 109

[Berz and Hofstätter 1998] Berz, M. and Hofstätter, G., 1998. “Computation and application of Taylor polynomials with interval remainder bounds,” *Reliable Computing*, 4(1):83–97. 28, 36, 94

[Bolin and Meyer 1995] Bolin, M. R. and Meyer, G. W., 1995. "A frequency based ray tracer," *Computer Graphics*, 29(Annual Conference Series):409–418.

[Corliss and Rall 1984] Corliss, G. F. and Rall, L. B. 1984. "Automatic generation of Taylor series in Pascal-SC: Basic operations and applications to differential equations," in *Trans. of the First Army Conference on Applied Mathematics and Computing (Washington, D.C., 1983)*, pages 177–209. ARO Rep. 84-1, U. S. Army Res. Office, Research Triangle Park, N.C.

[Csendes and Ratz 1997] Csendes, T. and Ratz, D., 1997. "Subdivision direction selection in interval methods for global optimization," *SIAM Journal on Numerical Analysis*, 34(3):922–938.

[de Cusatis Jr. et al. 1999] de Cusatis Jr., A., de Figueiredo, L. H., and Gattas, M. 1999. "Interval methods for ray casting implicit surfaces with affine arithmetic," in *SIBGRAPI '99*, pages 65–71. IEEE Computer Press. 140

[De Figueiredo and Stolfi 1996] De Figueiredo, L. H. and Stolfi, J., 1996. "Adaptive enumeration of implicit surfaces with affine arithmetic," *Computer Graphics Forum*, 15(5):287–296.

[Duff 1992] Duff, T. 1992. "Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry," in *SIGGRAPH '92: 19th International Conference on Computer Graphics and Interactive Techniques, Chicago, Illinois, July 26–31, 1992*, Catmull, E. E., editor, volume 26, pages 131–138, New York, NY 10036, USA. ACM Press.

[Eckmann et al. 1986] Eckmann, J.-P., Malaspinas, A., and Kamphorst, S. O. 1986. "A software tool for analysis in function spaces," Rep. Ugva-Dpt 1986/11-524, Department Theor. Phys., University Geneva, Geneva, Switzerland. 36

[Epstein et al. 1981] Epstein, C., Miranker, W. L., and Rivlin, T. J. 1981. "Ultra-arithmetic part 2: Intervals of polynomials," Research Report RC 8743, IBM Thomas J. Watson Res. Cent., Yorktown Heights, New York. Published In: Math. Comput. Simulation 24, 19–29, 1982. 36

[Epstein et al. 1982] Epstein, C., Miranker, W. L., and Rivlin, T. J., 1982. "Ultra-arithmetic I: function data types," *Mathematics and Computers in Simulation*, 24(1):1–18. 36

[Garloff 1985] Garloff, J. 1985. "Convergent bounds for the range of multivariate polynomials," in *Proceedings of the International Symposium on Interval Mathematics*, Nickel, K., editor, volume 212 of *LNCS*, pages 37–56, Freiburg, FRG. Springer. 44

[Garloff and Krawczyk 1986] Garloff, J. and Krawczyk, R., 1986. "Optimal inclusion of a solution set," *SIAM Journal on Numerical Analysis*, 23(1):217–226.

[Gascuel 1995] Gascuel, J.-D. 1995. "Implicit patches: An optimised and powerful ray intersection algorithm," in *Implicit Surfaces'95*, pages 143–160, Grenoble, France. Proceedings of the first international workshop on Implicit Surfaces.

[Hanrahan 1983] Hanrahan, P., 1983. "Ray tracing algebraic surfaces," *Computer Graphics*, 17(3):83–90.

[Hansen 1969] Hansen, E. 1969. "The centered form," in *Topics in Interval Analysis*, Hansen, E., editor, pages 102–106. Oxford University Press. 39

[Hansen 1978] Hansen, E., 1978. "Interval forms of Newton's method," *Computing*, 20(2):153–163. 47

[Hansen and Sengupta 1981] Hansen, E. and Sengupta, S., 1981. "Bounding solutions of systems of equations using interval analysis," *BIT*, 21(2):203–211. 50

[Hansen 1988] Hansen, E. R. 1988. "An overview of global optimization using interval analysis," in *Reliability in Computing*, Moore, R. E., editor, pages 289–307. Academic Press, New York.

[Hansen 1992] Hansen, E. R. 1992. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York. 7

[Hansen 1993] Hansen, E. R., 1993. "Computing zeros of functions using generalized interval arithmetic," *Interval Computations = Interval'nye vychisleniia*, 3:3–28.

[Hansen 1997] Hansen, E. R., 1997. "Preconditioning linearized equations," *Computing*, 58(2):187–196.

[Hansen and Greenberg 1983] Hansen, E. R. and Greenberg, R. I., 1983. "An interval Newton method," *Applied Mathematics and Computation*, 12(2–3):89–98.

[Hansen et al. 1990] Hansen, E. R., Patrick, M. L., and Wang, R. L. C., 1990. "Polynomial evaluation with scaling," *ACM Transactions on Mathematical Software*, 16(1):86–93.

[Hansen and Walster 2003] Hansen, E. R. and Walster, G. W. 2003. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York. 2, 3, 7

[Hart 1993a] Hart, J. 1993. "Ray tracing implicit surfaces," in *Modeling, Visualizing and Animating Implicit Surfaces*. SIGGRAPH Course Notes 25.

[Hart 1993b] Hart, J. 1993. "Sphere tracing: Simple robust antialiased rendering of distance-based implicit surface," in *Modeling, Visualizing and Animating Implicit Surfaces*. SIGGRAPH Course Notes 25. 140

[Hart and DeFanti 1991] Hart, J. C. and DeFanti, T. A., 1991. "Efficient antialiased rendering of 3-D linear fractals," *Computer Graphics*, 25(4):91–100. 140

[Hart et al. 1989] Hart, J. C., Sandin, D. J., and Kauffman, L. H., 1989. "Ray tracing deterministic 3-D fractals," *Computer Graphics*, 23(3):289–296. 140

[Heckbert and Hanrahan 1984] Heckbert, P. S. and Hanrahan, P., 1984. "Beam tracing polygonal objects," *Computer Graphics*, 18(3):119–127. 136, 137

[Herbison-Evans 1994] Herbison-Evans, D. 1994. "Solving quartics and cubics for graphics," Technical Report TR-94-487, Basser Department of Computer Science, University of Sidney, Sidney, Australia.

[Ito 1989] Ito, H., 1989. "Ray tracing meta-balls," *Pixel*, (77):76–80. In Japanese.

[Jaulin et al. 2001] Jaulin, L., Kieffer, M., Walter, O., and Didrit, O. 2001. *Applied Interval Analysis*. Springer Verlag. 7

[Kalra and Barr 1989] Kalra, D. and Barr, A. H., 1989. "Guaranteed ray intersections with implicit surfaces," *Computer Graphics*, 23(3):297–306. 139

[Kaucher and Miranker 1983a] Kaucher, E. and Miranker, W. L. 1983. "Iterative residual correction and self-validating numerics in functoids," Research Report RC 10260, IBM Thomas J. Watson Res. Cent., Yorktown Heights, New York. 36

[Kaucher and Miranker 1983b] Kaucher, E. and Miranker, W. L. 1983. "Numerics with guaranteed accuracy for function space problems," Research Report RC 9789, IBM Thomas J. Watson Res. Cent., Yorktown Heights, New York. 36

[Kaucher and Miranker 1984a] Kaucher, E. and Miranker, W. L., 1984. "Residual correction and validation in functoids," *Computing. Supplementum*, 5:169–192. 36

[Kaucher and Miranker 1984b] Kaucher, E. W. and Miranker, W. L. 1984. *Self-Validating Numerics for Function Space Problems — Computation with Guarantees for Differential and Integral Equations*. Academic Press, New York. 36

[Kay and Kajiya 1986] Kay, T. L. and Kajiya, J. T., 1986. "Ray tracing complex scenes," *Computer Graphics*, 20(4):269–278.

[Kearfott 1996] Kearfott, R. B. 1996. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht, Netherlands.

[Krawczyk and Neumaier 1984] Krawczyk, R. and Neumaier, A. 1984. "An improved interval Newton operator," Freiburger Intervall-Ber. 84/4, Universität Freiburg, Freiburg, Germany. Published in: J. Math. Anal. Appl. 118, 194–207, 1986. 49

[Krawczyk and Neumaier 1985] Krawczyk, R. and Neumaier, A., 1985. "Interval slopes for rational functions and associated centered forms," *SIAM J. Numer. Anal.*, 22:604–616.

[Krawczyk and Nickel 1981] Krawczyk, R. and Nickel, K. 1981. "Die Zentrische Form in der Intervallarithmetik, Ihre Quadratische Konvergenz und Ihre Inklusionsisotonie," Freiburger

Intervall-Ber. 81/9, Universität Freiburg, Freiburg, Germany. Published in: Computing 28, 117–137, 1982. 40

[Makino and Berz 2003] Makino, K. and Berz, M., 2003. "Taylor models and other validated functional inclusion methods," *International Journal of Pure and Applied Mathematics*, 4(4):379–456. 109

[Mayer 1992] Mayer, G., 1992. "Some remarks on two interval-arithmetic modifications of the Newton method," *Computing*, 48(1):125–128.

[Miranker 1983] Miranker, W. L. 1983. "Ultra-arithmetic: The digital computer set in function space," in *Parallel and Large-Scale Computers: Performance, Architecture, Applications*, Ruschitzka, M., Christensen, M., Ames, W. F., and Vichnevetsky, R., editors, volume 2 of *IMACS*, pages 275–279. North Holland, Amsterdam, Netherlands. 36

[Mitchell 1990] Mitchell, D. P. 1990. "Robust ray intersection with interval arithmetic," in *Graphics Interface 90, Halifax, Nova Scotia, 14–18 May 1990: proceedings*, pages 68–74, Toronto, Ont., Canada. Canadian Inf. Process. Soc. 139

[Mitchell and Hanrahan 1992] Mitchell, D. P. and Hanrahan, P. 1992. "Illumination from curved reflectors," in *SIGGRAPH '92: 19th International Conference on Computer Graphics and Interactive Techniques, Chicago, Illinois, July 26–31, 1992*, Catmull, E. E., editor, volume 26, pages 283–291, New York, NY 10036, USA. ACM Press.

[Moore 1962] Moore, R. E. 1962. *Interval arithmetic and automatic error analysis in digital computing.* PhD thesis, Dept. of Mathematics, Stanford University. 1, 36

[Moore 1966] Moore, R. E. 1966. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, USA. 1, 36, 39

[Moore 1979] Moore, R. E. 1979. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. 7, 18, 36

[Nataraj and Kotecha 2002] Nataraj, P. S. V. and Kotecha, K., 2002. "An algorithm for global optimization using the taylor-bernstein form as inclusion function," *Journal of Global Optimization*, 24(4):417–436. 36

[Neumaier 1983] Neumaier, A. 1983. "An interval version of the secant method," Freiburger Intervall-Ber. 83/10, Universität Freiburg, Freiburg, Germany. Published in: Bit 24, 366–372, 1984.

[Neumaier 1990] Neumaier, A. 1990. *Interval Methods for Systems of Equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, UK. 7, 15

[Neumaier 2002] Neumaier, A. 2002. "Taylor forms - use and limits,". 2002. 36

[Owen and Rockwood 1989] Owen, J. C. and Rockwood, A. P., 1989. "Intersection of general implicit surfaces," *ACM Transactions on Graphics*.

[Ratschek 1980] Ratschek, H., 1980. "Centered forms," *SIAM Journal on Numerical Analysis*, 17(5):656–662. 39

[Ratschek and Rokne 1984] Ratschek, H. and Rokne, J. 1984. *Computer Methods for the Range of Functions*. Ellis Horwood Ser.: Math. Appl. Ellis Horwood.

[Ratschek and Rokne 2003] Ratschek, H. and Rokne, J. 2003. *Geometric Computations with Interval and New Robust Methods*. Horwood Series in Computer Science. Horwood Publishing.

[Rivlin 1970] Rivlin, T. J., 1970. "Bounds on a polynomial," *J. Res. Nat. Bur. Standards Sect. B*, 74B:47–54. 43

[Rokne 1977] Rokne, J., 1977. "Bounds for an interval polynomial," *Computing*, 18(3):225–240. 44

[Rokne 1978] Rokne, J. 1978. "A note on the Bernstein algorithm for bounds for interval polynomials," Technical Report 78/29/8, University of Calgary. 44

[Rokne 1981] Rokne, J. 1981. “Optimal computation of the Bernstein algorithm for the bound of an interval polynomial,” Freiburger Intervall-Ber. 81/4, Universität Freiburg, Freiburg, Germany. Published in: Computing 28, 239–246, 1982. 44

[Savchenko and Pasko 1995] Savchenko, V. and Pasko, A. 1995. “Collision detection for functionally defined deformable objects,” in *Implicit Surfaces’95*, pages 217–222, Grenoble, France. Proceedings of the first international workshop on Implicit Surfaces.

[Shary 1999] Shary, S. P. 1999. “Interval Gauss-Seidel method for generalized solution sets to interval linear systems,” in *MISC ’99 — Workshop on Applications of Interval Analysis to Systems and Control (Girona, Spain, February 24-26, 1999)*, pages 51–65. Universidad de Girona.

[Sherstyuk 1998] Sherstyuk, A. 1998. “Fast ray tracing of implicit surfaces,” in *Implicit Surfaces ’98*, pages 145–153. 140

[Shinya et al. 1987] Shinya, M., Takahashi, T., and Naito, S., 1987. “Principles and applications of pencil tracing,” *Computer Graphics*, 21(4):45–54. 137

[Snyder 1992] Snyder, J. M., 1992. “Interval analysis for computer graphics,” *Computer Graphics*, 26(2):121–130.

[Stahl 1996] Stahl, V. 1996. “Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations,” Technical Report 96-17, RISC-Linz, Johannes Kepler University, Linz, Austria. Published in Ph.D. thesis. 36, 38, 40, 44, 52

[Stolfi and de Figueiredo 1997] Stolfi, J. and de Figueiredo, L. H. 1997. “Self-validated numerical methods,” in *Monograph for the 21st Brazilian Mathematics Colloquim*. IMPA, Rio de Janeiro. 140

[Tonnesen 1989] Tonnesen, D. 1989. “Ray-tracing implicit surfaces resulting from the summation of bounded polynomial functions,” Technical Report TR-89003, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, Ney York.

[Tuy and Tuy 1984] Tuy, T. H. and Tuy, L. T., 1984. "Direct 2-d display of 3-d objects," *IEEE Computer Graphics and Applications*, 4(10):29–33. 138

[Walster et al. 1985] Walster, G. W., Hansen, E. R., and Sengupta, S. 1985. "Test results for a global optimization algorithm," in *Numerical Optimization 1984*, Boggs, P. T., Byrd, R. H., and Schnabel, R. B., editors, pages 272–287, Philadelphia. SIAM.

[Watt 1990] Watt, M., 1990. "Light-water interaction using backward beam tracing," *Computer Graphics*, 24(4):377–385. 137

[Wyvill and Trotman 1990] Wyvill, G. and Trotman, A. 1990. "Ray-tracing soft objects," in *Computer Graphics International '90*, pages 469–475.