A 细分非线性方程数据集

JuanXu[1,2],赖辉龙[1,2],程莹莹[1,2], 杨文强[1,2], 和 Changbo Chen[*1,2]

[1]中国科学院重庆绿色智能技术研究院，中国重庆[2]中国科学院大学重庆学院，中国重庆

arXiv:2603.27499v1 [cs.SC] 29 Mar 2026

2026年3月31日

摘要

目的：细分法是一种基础且重要的可靠方法，可用于求解非线性方程组，包括多项式方程和超越方程。本研究旨在提供一个大型真实数据集，用于细分方法的基准测试与优化，并助力开发求解非线性方程的机器学习方法。

设计/方法/途径：为了构建该数据集，我们检索了现有的主要细分方法基准测试集。我们还浏览了 1000 多篇文献，并详细检索了 300 多篇与细分相关的文献，以寻找这些资源库未收录的示例。在仔细比较这些系统并删除重复示例后，我们从文献和公开数据集中整理出 451 个多项式系统和 130 个非多项式系统。为扩充数据集，我们进一步从实际应用中出现的 5 个族参数化非线性系统生成了 48000 个零维实例。我们采用两种细分求解器 IbexSolve 和 RealPaver，以及 Maple 中的符号求解器 RootFinding:-Isolate，为这些示例提供可靠的解。

研究发现：通过为数据集中的示例提供解信息，我们对细分求解器进行了大规模测试，并得出以下主要发现：从文献和代码库收集的示例之间存在大量重叠；总体而言，两种最先进的细分求解器 IbexSolve

*通讯作者。电子邮箱：chenchangbo@cigit.ac.cn

和 RealPaver 的效率高于符号求解器 RootFinding:-Isolate，而 IbexSolve 的效率又高于 RealPaver；不过，没有哪种求解器始终比其他求解器更快。尽管细分求解器被认为是可靠的，但我们的实验表明，在极少数情况下，IbexSolve 可能会漏掉一些解。

研究局限：相关研究一直着重于收集用于细分的零维系统。研究未涵盖利用细分法求解正维系统的相关文献或代码库，例如曲线或曲面的近似。虽然也收录了一些使用符号方法或同伦方法求解零维系统的文献和代码库，但本研究主要聚焦于细分方法，并未系统地收集非细分方法的示例。

实际意义：该数据集可用于对不同细分求解器进行基准测试。由于数据集包含了使用两种最先进的细分求解器的默认策略获得的解信息，因此在开发新的细分方法或为现有细分方法设计替代求解策略时，可将其作为可靠的基准。这个带标签的数据集也可用于助力细分方法相关机器学习技术的发展。

原创性/价值：我们提供了迄今为止规模最大的真实数据集，其中包含适合使用细分法求解的零维非线性系统。我们投入了大量精力，剔除来自不同来源的重复系统，并获取数据集中各系统可靠的解信息。除可作为细分求解器的基准测试集外，这个带标签的数据集还有望用于训练和评估机器学习模型。本研究的另一项价值在于比较细分方法与符号方法；此前很少有人对此进行系统比较，而这项比较揭示了两种方法各自的优势与不足。

关键词：细分；数据集；零维非线性方程；多项式方程；超越方程。

1 引言

求解非线性方程组是许多领域中的一项基础任务。细分方法是一类可靠的方法，用于在有界区域内寻找非线性方程组的实解。其基本思想是反复划分感兴趣区域，直至解被隔离或得到足够精确的近似。当然，细分并不局限于实根计算；其变体已用于复根隔离（Becker 等人，2018年）和全局优化（Neumaier，2004年；Chabert 和 Jaulin，2009年）。然而，本文仅关注用于计算实根的细分方法。这些方法自然

属于分支与剪枝框架：经证明不含解的区域会被剪除，包含唯一解（或足够小）的区域会被保留，而不确定的区域则会进一步分支为更小的区域。可以看出，该框架与用于全局优化的分支定界（B&B）框架有许多相似之处，因此常被放在一起讨论（Neumaier，2004年；Chabert 和 Jaulin，2009年）。

除细分法外，符号方法（Cox 等，1997年；Von Zur Gathen 和 Gerhard，2003年）以及同伦延拓（Sommese 等，2005年）是求解非线性方程的两种主要方法。与细分法不同，这些方法主要适用于多项式系统。符号方法，或者更准确地说，消元方法，会将多项式系统转化为更易求解的三角系统。随后，可以应用通常基于细分法的实根隔离技术，确定该系统的解。符号方法在理论上优雅而严谨，但由于依赖精确算术，通常会耗费大量内存和计算资源。相比之下，同伦延拓通过追踪从已知（或易于计算）解的起始系统到目标系统的解路径，来求解多项式系统。它通常效率较高，但本身并不具备认证能力，因为追踪过程中发生的路径跳跃可能导致遗漏解。

在许多应用中，人们只关注指定定义域内的解。在这种情况下，正如 Sherbrooke 和 Patrikalakis（1993年）指出的那样，符号方法和同伦方法往往会提供超出所需的信息：符号消元仍会将系统转化为三角形式，而同伦延拓仍会追踪所有复解路径；这些步骤通常占据了主要运行时间。相比之下，细分法将计算限制在指定区域内，因此区域越小，成本越低。另一个区别是：许多细分方法适用于一般非线性系统（包括含有初等函数的系统），而符号方法和同伦方法则主要限于多项式系统。

细分方法中使用的分支与剪枝框架与分支定界框架密切相关。启发式方法在分支定界算法中一直发挥着重要作用。要成功应用这些启发式方法，需要在有效性与效率之间取得平衡。例如，强分支启发式方法已被证明能有效缩小分支定界树的规模。另一方面，由于需要为所有分支变量求解线性规划问题，应用这类启发式方法的成本很高。为降低这类成本，可以应用监督式机器学习来模仿强分支的行为。仅仅依靠模仿，几乎不可能找到比人类设计的方法更有效的启发式方法。另一个重要方向是应用强化学习，以更自主的方式寻找求解策略，力求超越人类设计的最佳策略。尽管将机器学习融入分支定界算法已取得重要进展

（Scavuzzo 等，2024），但近期一项针对基于机器学习的组合优化求解器（包括混合整数线性规划求解器）的评估显示，基于学习的方法仍难以超越最佳求解器（Feng 等，2025），尤其是在困难的基准示例上。然而，无论是训练还是评估基于学习的方法，都必须收集高质量的基准示例。我们也有兴趣应用机器学习来加速细分。为此，必要的一步是拥有真实数据集，以评估不同的方法或启发式方法。

本文介绍了迄今为止构建的最大规模带标签数据集，该数据集用于通过基于细分的方法求解零维非线性系统。此外，我们还提供了一份简要但不求面面俱到的综述，重点介绍过去二十年的相关文献，以配合该数据集。我们通过对现有实现进行基准测试，以及将该数据集用于训练模型以分类参数系统的实根，展示了该数据集的价值。总之，我们的贡献有三方面：

- 我们综述了用于求解零维系统的细分算法的发展历程，可为理解该数据集提供指引，也可作为对细分优化感兴趣的初学者的入门教程。

- 我们从广泛的文献和细分基准数据集中收集了用于细分的零维示例。我们使用若干不同的求解器处理这些示例，并收集时间和解信息，从而构建了一个用于细分的带标签实数数据集。

- 我们通过对不同求解器进行基准测试、找出改进细分方法的潜在空间，以及训练机器学习模型来分类参数化非线性系统的实解数量，展示了该数据集的实用价值。

2 符号和预备知识

本节概述全文将使用的符号和预备背景。按照 Kearfott 等（2010）提出的标准化记法，我们用小写字母表示标量和向量，用粗体小写字母表示区间和盒（区间向量），用大写字母表示矩阵。为简化记号，默认向量均视为列向量；若上下文含义明确，则省略转置符号。

由 $[a,b]$ with $a \le b$ 表示的一个（实、闭、非空）区间是集合 $\{x \in \mathbb{R} \mid a \le x \le b\}$。我们还允许端点取自 $\mathbb{R} \cup \{-\infty, +\infty\}$，以纳入无界区间。令 $\mathbb{IR}$ 表示所有（可能无界的）实区间的集合。盒可以视为一个区间向量 $\boldsymbol{x} = (\boldsymbol{x}_1、\ .....、\ \boldsymbol{x}_n) \subseteq \mathbb{IR}^n$，它由所有满足以下条件的点组成

$x = (x_1、 \cdots、 x_n)$，其中 $x_k \in \boldsymbol{x}_k$ 对任意 $k = 1、.....、 n$ 均成立。一维区间可以视为盒的一种特殊情况。对于维度大于 1 的盒 $\boldsymbol{x}$ ，我们用 i 表示其第 i 个分量 $\boldsymbol{x}_i$。

给定区间 $\boldsymbol{x}$，分别用 $\boldsymbol{x}$ 和 $\underline{\boldsymbol{x}}$ 表示 $\boldsymbol{x}$的下界和上界（端点）。区间 $\boldsymbol{x} = [\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$ 的宽度和中心分别定义为 $\mathsf{wid}(\boldsymbol{x}) = \overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}$和 $\mathsf{mid}(\boldsymbol{x}) = \frac{\underline{\boldsymbol{x}} + \overline{\boldsymbol{x}}}{2}$。盒 $\boldsymbol{x} = （\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n）$的宽度定义为向量 $\overline{\mathsf{wid}}(\boldsymbol{x}) = (\mathsf{wid}(\boldsymbol{x}_1) \ldots, \mathsf{wid}(\boldsymbol{x}_n))$，最大宽度则定义为 $\max \mathsf{wid}(\boldsymbol{x}) = \max_{1 \le i \le n} \mathsf{wid}(\boldsymbol{x}_i)$。类似地，盒的中心定义为 $\mathsf{mid}(\boldsymbol{x}) = (\mathsf{mid}(\boldsymbol{x}_1) \ldots, \mathsf{mid}(\boldsymbol{x}_n))$，区间矩阵的中心也以类似方式按分量定义。除非另有说明，$\mathsf{wid}(\cdot)$ 和 $\mathsf{mid}(\cdot)$ 均针对有界区间（和盒）定义。用 $\mathsf{int}(\boldsymbol{x})$ 表示盒 $\boldsymbol{x}$的内部。给定集合$S \subset \mathbb{R}$，用 S 表示包含 S的最小区间。

□

对于任意运算 $\circ \in \{+, -, *\}$，两个区间 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 之间的运算定义为

$$\boldsymbol{a} \circ \boldsymbol{b} = \{a \circ b \mid a \in \boldsymbol{a}, b \in \boldsymbol{b}\}.$$

区间之间的除法和一些基本初等函数（不含复合）可以用类似的方式定义，不过，如果除数包含零，或者函数未在整个区间上都有定义，就需要特别谨慎。详细公式可参见 Hansen and Walster (2003年) 等经典文献。

在区间算术中，明确区分函数和表达式至关重要。函数表示从输入到输出的抽象数学映射，而表达式则指用于求值该函数的运算的特定语法排列。由于所谓的依赖问题，表示同一函数的不同表达式可能导致截然不同的区间求值。概念上，我们会保留这一区分；但为方便记号，通常对函数及其对应的某个固定表达式使用同一符号，例如 f；其具体含义可由上下文判断。我们用小写字母表示标量值函数，用大写字母表示向量值函数或表达式。粗体字母用于表示区间值或盒值函数或表达式。

给定表达式 f 和盒 $\boldsymbol{x}$，将 $\boldsymbol{x}$ 的各分量区间代入表达式并执行区间算术运算，所得的区间求值记为 $\boldsymbol{f}(\boldsymbol{x})$。注意， $\boldsymbol{f}(\boldsymbol{x})$通常不等于函数 f在盒 $\boldsymbol{x}$上的精确值域，该值域定义为 $\mathsf{range}(f, \boldsymbol{x}) := \{f(x) \mid x \in \boldsymbol{x}\}$。相反，由于依赖问题，精确值域始终包含在计算所得的区间求值之中：$\mathsf{range}(f, \boldsymbol{x}) \subseteq \boldsymbol{f}(\boldsymbol{x})$。以此方式构造的区间函数 $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{IR}^n \to \mathbb{IR}$称为由 f表示的函数的自然区间扩展 。这种扩展给出了 f真实值域的一个包含区间，其包含性保证计算所得的区间始终包含精确

值域。此外，随着输入盒的宽度趋于零，自然区间扩展会收敛到函数的精确值域（假设使用精确算术且不存在舍入误差）。由于一个函数可能有多个代数等价表达式，因此也可能对应多个区间扩展。不同的区间扩展可能得到不同的值域包含区间。一个值得注意的例子是霍纳扩展：它将多项式改写为嵌套的（霍纳）形式，然后对改写后的表达式应用自然区间扩展。这种扩展通常所需的区间运算较少，并能减轻依赖效应，往往得到更紧的包含区间。另一种常用的区间扩展是均值扩展。给定可微函数 f 和盒 $\boldsymbol{x}$，f 的均值扩展按以下公式构造：

$$\boldsymbol{f}_{\mathrm{MV}}(\boldsymbol{x}) = f(m) + \boldsymbol{J}_f(\boldsymbol{x})(\boldsymbol{x} - m)$$

其中，m是 $\boldsymbol{x}$的中心，而 $\boldsymbol{J}_f(\boldsymbol{x})$是$f$ 在 $\boldsymbol{x}$上的雅可比矩阵的区间扩展。

3 细分综述

本节综述用于认证实根求解的细分方法。与消元法和同伦方法不同，消元法和同伦方法通过代数变换或连续变形，将原系统与一个更容易求解的系统联系起来，从而求解原系统；细分方法则以一种相当自然的方式，直接在变量的指定定义域内搜索原系统的解。

我们首先介绍细分方法的一般框架，然后讨论关键的算法要素，包括验证算子和收缩算子。最后，我们回顾选择和组合这些要素的策略。

3.1 细分框架（算法）

细分算法通常遵循以下通用框架：给定一个感兴趣的盒$\boldsymbol{x}^{(0)} \subseteq \mathbb{R}^n$，算法首先尝试尽可能缩小该盒。然后检查该盒是否满足输出判据。如果满足判据，则将该盒添加到输出列表；否则，将该盒二分，并对得到的子盒递归地执行相同的过程。

算法 1： 框架

输入： 一个初始感兴趣盒 $\boldsymbol{x}^{(0)} \subseteq \mathbb{R}^n$。

输出： 一个盒列表 $\mathcal{L}$，其中每个盒都包含于 $\boldsymbol{x}^{(0)}$，并满足输出判据。

1 设定 $\mathcal{W} = \{\boldsymbol{x}^{(0)}\}$； $\mathcal{L} = \emptyset$;

2 **当** $\mathcal{W} \neq \emptyset$ **时**

3 通过**节点选择策略**从 $\mathcal{W}$ 中选取并移除一个盒 $\boldsymbol{x}^{(k)}$；

4 通过移除已证明不包含任何根的区域来减少 $\boldsymbol{x}^{(k)}$

收缩策略移除已证明不包含根的区域，从而缩减；

5 **如果** $\boldsymbol{x}^{(k)}$ 非空，**则**

6 **如果** $\boldsymbol{x}^{(k)}$ 通过**检查策略**满足输出判据 **then**

7 将 $\boldsymbol{x}^{(k)}$ 添加到 $\mathcal{L}$;

8 **else**

9 将 $\boldsymbol{x}^{(k)}$ 分成两个子盒 $\boldsymbol{x}^{(k,1)}$，$\boldsymbol{x}^{(k,2)}$ 通过**二分策略**;

10 将 $\boldsymbol{x}^{(k,1)}$ 和 $\boldsymbol{x}^{(k,2)}$ 添加到 $\mathcal{W}$;

11 **end**

12 **end**

13 **结束**

14 **返回** $\mathcal{L}$;

算法 1 概述了细分算法的通用框架，该框架遵循分支定界结构（Caro 等，2014）。实现上述框架还需要进一步明确并实例化几个关键分量：第 3 行的节点选择策略、第 4 行的收缩策略、第 9 行的二分策略，以及第 6 行的检查策略。策略的选择一方面取决于算法的目标，另一方面也受到效率方面考量的影响。

该算法的目标可以从两个角度进行描述：从输出数量来看，可分为三类：求得单个解、指定数量的解，或盒内的所有解；从输出要求来看，可分为两类：根隔离或根近似。

对于根隔离，检查策略是检测盒 B 是否恰好包含一个根，这通常通过验证算子实现。对于根近似，检查策略是检测 B 的大小是否低于预设容差 ε。在根近似的情况下，只要第 9 行的二分策略选择得当，就能保证算法终止。相比之下，对于根隔离，即使工作列表中的某个盒确实包含一个根，要证明它最终会通过验证算子的确认通常也很困难。因此，文献中的大多数算法都引入容差作为备用终止条件，并将足够小的盒接受为输出，尽管它们可能并未真正实现

隔离。作为例外，(Stahl, 1995年；Xu 和 Yap, 2019年)分析了存在单根时验证算子的成功条件，从而无需依赖此类容差即可终止。需要注意的是，验证算子无法确认多重根的存在。这是数值算法面临的常见难题。存在多重根时，通常会引入容差以确保算法终止。

文献中的大多数研究旨在找出给定盒中的所有根。在这种情况下，节点选择策略既不影响输出，也不影响效率，因此可以任意选择策略（Araya 和 Reyes，2016年）。通常更倾向于采用深度优先策略，因为这样可以在不切换区域的情况下完整处理一个盒。相比之下，如果目标是只找到一个或少数几个根，节点选择策略就至关重要，因为有些根更容易找到，而另一些则更难找到，尤其是在根是病态的情况下。文献中这类情况并不常见，只有少数例外。在 Huang 等人（2019）的研究中，目标是找到一个根，但仍采用深度优先选择策略；研究重点转而放在划分策略的选择上。在 Bublitz（2023）中，为了快速找到系统的一个根，节点选择策略会优先选择这样一个盒：其中心处函数求值的绝对值之和最小。同时，研究还采用了一种启发式求根过程，以加快向根的收敛。尽管针对零维系统的节点选择策略鲜有研究，但它们发挥着重要作用，并已在正维系统的分支剪枝算法（de Latour 等人，2021年；Chenouard 等人，2009年）以及全局优化中的分支定界算法（Neveu 等人，2016年）中得到讨论。

在任何情况下，收缩策略和二分策略对于算法的效率都至关重要。我们将在介绍收缩算子之后，于第 3.4 节讨论这些方面。

3.2 验证算子

验证算子用于检查盒内是否恰有一个解。在子分算法中，最常用的验证算子是区间牛顿型算子，例如 Krawczyk算子和 Hansen-Sengupta 算子。

Krawczyk算子源自不动点定理，因其简洁且易于实现而特别受欢迎。给定一个盒 $\boldsymbol{x}$，Krawczyk算子定义为

$$K(\boldsymbol{x}, \tilde{x}) := \tilde{x} - C\boldsymbol{f}(\tilde{x}) + (\mathbb{I} - C \cdot \boldsymbol{J}_f(\boldsymbol{x}))(\boldsymbol{x} - \tilde{x})$$

其中， $\tilde{x}$ 是 $\boldsymbol{x}$中的一个点， C 是预条件矩阵， $\boldsymbol{f}$ 和 $\boldsymbol{J}_f$ 分别是 f 和 f的雅可比矩阵的区间扩展。已有证明表明，当 $\tilde{x} = \text{mid}(\boldsymbol{x})$且 $C = \text{mid}(\boldsymbol{J}_f(\boldsymbol{x}))^{-1}$时，Krawczyk算子能得到最佳（最紧）的结果。Krawczyk

算子 $K(\boldsymbol{x},\ \tilde{x})$具有以下性质（Neumaier，2004年）：

1. f的任意零点 $x^* \in \boldsymbol{x}$ 都位于 $\boldsymbol{x} \cap K(\boldsymbol{x}, \tilde{x})$中。

2. 如果 $\boldsymbol{x} \cap K(\boldsymbol{x}, \tilde{x}) = \emptyset$，那么 $\boldsymbol{x}$ 不包含 f的零点。

3. 如果 $K(\boldsymbol{x}, \tilde{x}) \subseteq \text{int}(\boldsymbol{x})$，那么 $\boldsymbol{x}$ 包含 f的唯一零点。

这些性质表明，Krawczyk算子既可用作验证算子，也可用作收缩算子。虽然Krawczyk算子简单且在实践中广泛使用，但其能力不如Hansen-Sengupta 算子（Neumaier，1990）。为介绍Hansen-Sengupta 算子的定义，我们定义一个作用于区间的函数$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x} \in \mathbb{IR}$:

$$\Gamma(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}) := \square\big\{x \mid ax = b \text{ for some } a \in \boldsymbol{a}, b \in \boldsymbol{b}\big\}.$$

该函数可扩展到区间矩阵和向量，写作 $\Gamma(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}) := \boldsymbol{y}$其中$\boldsymbol{A} \in \mathbb{IR}^{n\times n}, \boldsymbol{b} \in \mathbb{IR}^n$, $\boldsymbol{x} \in \mathbb{IR}^n$，且 $\boldsymbol{y}$的第 i个分量 $\boldsymbol{y}_i$定义为

$$\boldsymbol{y}_i = \Gamma(\boldsymbol{A}_{ii}, \boldsymbol{b}_i - \sum_{j<i} \boldsymbol{A}_{ij}\boldsymbol{y}_i - \sum_{j>i} \boldsymbol{A}_{ij}\boldsymbol{x}_i, \boldsymbol{x}_i), \quad i = 1, \ldots, n. \tag{1}$$

算子 $\Gamma(\boldsymbol{A},\ \boldsymbol{b},\ \boldsymbol{x})$ 称为区间 Gauss-Seidel 算子，用于求解区间线性系统。它提供了下列集合的一个包含区间

$$\big\{x \mid Ax = b \text{ for some } A \in \boldsymbol{A}, b \in \boldsymbol{b}\big\}. \tag{2}$$

给定一个盒 $\boldsymbol{x}$，Hansen-Sengupta 算子定义为

$$H(\boldsymbol{x}, \tilde{x}) := \tilde{x} + \Gamma(C \cdot \boldsymbol{J}_f(\boldsymbol{x}), -C\boldsymbol{f}(\tilde{x}), \boldsymbol{x} - \tilde{x}) \tag{3}$$

其中，$\tilde{x}$ 是 $\boldsymbol{x}$ 中的一个点，C 是一个预条件矩阵。该算子具有与前文所述Krawczyk算子相同的优良性质。此外，作为收缩算子使用时，Hansen–Sengupta 算子在实践中通常更为有效，因为它始终能为解集生成更紧，或至少同样紧的包含区间。

将式 (3) 中的区间 Gauss-Seidel 算子替换为区间版高斯消元法（Neumaier，2004年，1990），即可得到区间 Newton 型算子的另一种变体。这种新算子保留了区间牛顿型方法的关键性质，实验表明，它比 Hansen-Sengupta 算子更强大（Jaulin 等人，2001）。然而，由于计算成本较高，它在实践中较少使用。

如上所述，区间 Newton 类型方法可用作验证算子，用于确认盒内根的唯一性。除此之外，还存在其他检验，例如 Miranda 检验和 Borsuk 检验，它们只能验证解的存在性，

而不能验证盒内解的唯一性。Miranda 检验通过检查盒的相对面上的符号变化，确认盒内连续函数的零点存在性。它可以被视为高维空间中介值定理的一种应用。
Borsuk 检验是一种拓扑方法，通过利用连续函数在对称域上的对跖性质来验证根的存在性。该方法在理论上功能强大，但实现更为复杂，因此在算法中很少使用。已有研究表明，它至少与 Miranda 检验同样强大（Frommer and Lang，2005年）。在实践中，Miranda 检验通常在预条件化之后应用（Moore and Kioustelidis，1980年）。研究表明，Miranda 检验的一种标准实现至少与基于 Krawczyk算子的存在性检验同样强大（Frommer et al.，2004年），但不如基于 Hansen–Sengupta 算子的检验强大（Goldsztejn，2007年），而计算成本相近。

3.3 收缩算子

如前所述，区间牛顿型算子可以用作收缩算子。然而，这类算子的计算成本很高，因为它们需要对整个系统进行运算，包括计算区间雅可比矩阵，以及同时求解涉及所有变量的区间线性系统。在搜索早期将其应用于较大的盒时，区间雅可比矩阵的各元素往往具有较大的宽度，这可能导致算子性能不佳。在这种情况下，所实现的约简可能不足以抵偿其计算成本。一致性技术为盒约简提供了另一种方法。与作用于整个系统的区间 Newton 型方法不同，一致性技术作用于单个约束（即方程）。每个约束都独立用于约简其中涉及变量的定义域。与区间牛顿法相比，一致性技术的计算成本更低，更适合在搜索早期快速剪除较大的盒。

这里介绍连续约束满足问题中常用的一些一致性技术，其中最广泛使用的是凸包一致性和盒一致性。这些技术每次应用于一个约束。一般而言，约束可以是方程或不等式；本文聚焦于方程，即形式为 $f(x) = 0$ 的约束。

凸包一致性，也称为 2B一致性，旨在计算包围给定方程所有解的最小盒。盒 $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ 关于约束 $c(x_1, \ldots, x_n)$ 满足凸包一致性，当且仅当 $\forall i = 1, \ldots, n$

$$\boldsymbol{x}_i = \square\{r_i \in \boldsymbol{x}_i \mid \exists r_1 \in \boldsymbol{x}_1, \ldots, r_{i-1} \in \boldsymbol{x}_{i-1}, r_{i+1} \in \boldsymbol{x}_{i+1}, \ldots, r_n \in I_n : c(r_1, \ldots, r_n)\}.$$

换言之，如果在每个维度上，对应的区间都无法进一步缩小，因为解集在该维度上的投影

覆盖两个端点，那么该盒就是凸包一致的。凸包一致性通常通过前向-后向传播来实现（Benhamou 等人，1999年）。

给定一个约束，前向-后向传播会沿两个方向处理变量域：前向求值计算约束函数在输入变量当前盒上的区间像。后向求值则尝试通过与能够产生所计算输出区间的原像求交，来缩小涉及的每个变量的区间。

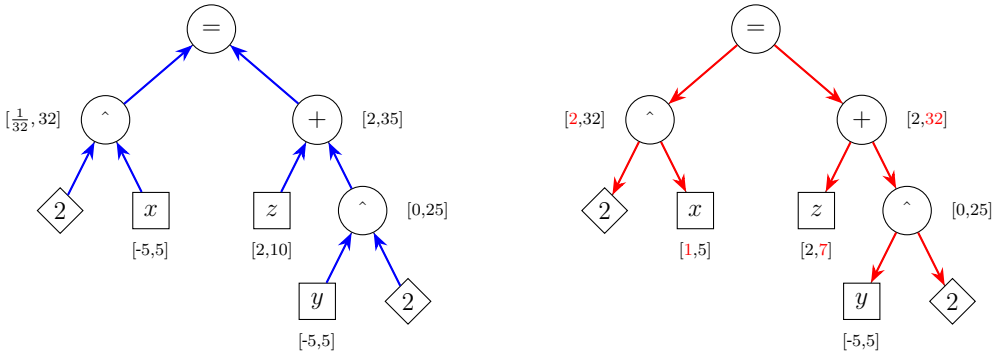


图 1：约束 $2^x = z + y^2$ 的前向传播（左）和后向传播（右）。

请注意，前向-后向传播会尝试同时缩小所有涉及的变量的范围。如果每个变量在约束中只出现一次，前向-后向传播会得到最优结果，也就是说，所得盒满足包一致性。然而，当变量多次出现时，在求值过程中，每次重复出现都会被视为一个新的独立变量。这破坏了固有的依赖结构，可能导致显著的过高估计，从而降低该方法的剪枝效率。

当变量多次出现时，盒一致性可能会得到更好的结果。与凸包一致性不同，盒一致性技术通过将原始多元约束转化为一元约束，并用其他所有变量的区间代替这些变量，从而分别细化每个变量的区间。

考虑约束 $f(x_1, \ldots\ldots, x_n) = 0$ 以及 f 的区间扩展 $\boldsymbol{f}$。当且仅当对于 $\forall i = 1, \ldots\ldots, n$，下列条件成立时，盒 $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n) \in \mathbb{IR}^n$ 相对于 $\boldsymbol{f}$ 具有盒一致性：

$$\begin{cases} 0 \in \boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, [\underline{\boldsymbol{x}_i}, \underline{\boldsymbol{x}_i}^+], \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n) \\ 0 \in \boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, [\overline{\boldsymbol{x}_i}^-, \overline{\boldsymbol{x}_i}], \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n) \end{cases}$$

其中，x^+（分别为 x^-）表示 x 的浮点后继（分别为 x 的浮点前驱）。注意，盒一致性是相对于 f 的区间扩展 $\boldsymbol{f}$ 定义的。对于同一函数 f，不同的区间扩展可能会产生不同形式的盒一致性（Van Hentenryck 等人，1997年）。为了强制实现盒一致性，

缩减盒 $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ 时，我们将每个变量 x_j（对于 $j \neq i$）固定为其当前区间 $\boldsymbol{x}_j$，并考虑由此得到的区间函数

$$\boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n).$$

由此得到关于 $\boldsymbol{x}$ 的一元区间函数。要对区间 $\boldsymbol{x}_i$进行剪枝，就需要收紧 $\boldsymbol{x}_i$，直至其端点接近 $\boldsymbol{f}_i$ 的最左和最右零点。为此，可以递归地分割区间，剪去不可行的子区间，并在尚未被排除的最左和最右子区间中搜索所需的零点。可以使用单变量区间牛顿法提高这一过程的效率。Goldsztejn 和 Goualard（2010）详细讨论了基于盒一致性的收缩算子的高效实现。

与通过存在量化定义的凸包一致性不同，盒一致性会将其他变量的区间代入约束的区间扩展进行求值。按照定义，凸包一致性似乎比盒一致性更严格。然而在实践中，当变量在约束中多次出现时，盒一致性技术比前向-后向传播得到的结果更紧。这是因为盒一致性一次只关注一个变量，并通过搜索该变量的最左和最右零点，有效减小了该维度上变量依赖的影响（Collavizza 等，1999）。另一方面，凸包一致性的计算成本较低，并且有望同时缩小所有相关变量的定义域。

由于凸包一致性和盒一致性是针对单个约束定义的，因此称为局部一致性。在细分算法中，必须在所有约束之间反复传播这些一致性技术，直到进一步缩减的幅度变得可以忽略。然而，由于各个约束是分别处理的，即使处理完所有约束，盒也可能无法收敛到解，如图2所示。在这种情况下，可以应用更强的一致性技术，进一步缩小盒。

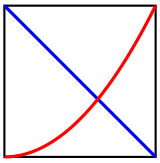


图2：盒 $([-1,1],[-1,1])$ 相对于系统 $\{x_2 = -x_1, 2x_2 = (x_1+1)^2 - 2\}$ 满足凸包一致性和盒一致性。

若盒 $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ 相对于约束集$\mathcal{S}$满足 3B 一致性，当且仅当对于每个 $i = 1, \ldots, n$，盒 $\boldsymbol{x}$ 相对于约化系统满足 2B 一致性（即凸包一致性）。

$$\mathcal{S} \cup \left\{x_i = \underline{\boldsymbol{x}_i}\right\} \quad \text{and} \quad \mathcal{S} \cup \left\{x_i = \overline{\boldsymbol{x}_i}\right\}.$$

很容易看出，图2中的盒对于这些约束并不满足 3B 一致性，因为例如，该盒对于系统 $\{x_2 = -x_1, 2x_2 = (x_1+1)^2 - 2, x_1 = -1\}$ 并不满足 2B一致性。根据 3B 一致性的定义，要在第 i 维上缩小盒 $\boldsymbol{x}$ ，可以从区间 $\boldsymbol{x}_i$ 的两端进行探测，并尝试使用前向-后向传播等 2B一致性技术，证明较小的边界子区间不可行。对于图2中的例子，可以利用 3B 一致性证明子盒 $([-1,0],[-1,1])$ 不可行。与盒一致性技术和凸包一致性技术相比，3B 一致性技术预计更强大，但代价也更高。如果在 3B 一致性的定义中用盒一致性替代凸包一致性，即 $\boldsymbol{x}$ 相对于约化系统满足盒一致性，那么由此得到的概念称为界一致性。有关上述一致性技术能力的比较，请参阅 Collavizza 等人（1999）。

通过归纳法，可将 3B 一致性的定义推广为 kB 一致性：若且唯若对于每个 $i = 1, \ldots, n$，盒 $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$ 相对于约束集 $\mathcal{S}$ 满足 kB 一致性，则称该盒相对于

$$\mathcal{S} \cup \left\{x_i = \underline{\boldsymbol{x}_i}\right\} \quad \text{and} \quad \mathcal{S} \cup \left\{x_i = \overline{\boldsymbol{x}_i}\right\}.$$

Ho然而， kB 技术在实际中很少用于 $k > 3$ ，因为它开销过高。 e.

在 Trombettoni 和 Chabert（2007）中，作者提出了两种改进的收缩算子：CID 和 3BCID。与标准的基于 3B 一致性的收缩算子不同，这些方法不仅尝试移除所选维度两端的边界子盒，还会检查内部的子盒。它们对这些中间区域应用凸包一致性收缩算子，从而能够在多个维度（甚至可能是所有维度）上进一步缩减内部子盒，进而缩减整个盒 $\boldsymbol{x}$。Neveu 等人（2015）随后提出了对 3BCID 的进一步改进。

线性松弛是缩小搜索盒范围的另一种有效方法。其核心思想是用一组线性不等式定义的多面体包围每个非线性方程的解集。通过求解由此得到的线性问题，可以改进每个维度上的界。仿射算术提供了一种构造系统线性松弛的方法。作为区间算术的扩展（Kolev，2001，2004；Messine，2002），仿射算术将每个变量表示为噪声符号的仿射组合，以保留变量之间的依赖关系，因此比区间算术能得到更紧的界。对约束中的所有表达式应用仿射算术运算后，这些约束会被转换为仿射形式。由此得到的仿射表示可自然地导出约束的线性松弛（Ninin 等，2015）。

Lebbah 等人（2005a、b）提出了一种用于多项式系统的线性松弛技术，该技术改编自重构线性化技术。此方法用辅助变量替代非线性项，并在

原系统中引入相应的恒等约束。随后构造线性不等式，以近似这些非线性项的含义。通过利用二次项的特定结构，该技术能够得到解集的紧包含区间。Domes 与 Neumaier Domes 和 Neumaier（2010）针对二次约束提出了一种替代松弛方法，并可自然地将其扩展到更一般的多项式约束。这两种松弛方法的主要区别在于处理双线性约束的方式。Domes 和 Neumaier（2012）介绍了对这些线性松弛技术的整合与比较，并提供了实验评估。

最近，Araya 等（2012）提出了一种线性松弛技术，该技术基于在盒的一对对置顶点处求值的区间泰勒展开。在 Araya 等（2025）的研究中，作者提出了一种混合方法，将仿射松弛技术与极值泰勒方法相结合，从而在受约束全局优化实例上取得了更好的性能。

如前所述，区间牛顿型算子（例如 Krawczyk算子和 Hansen-Sengupta 算子）凭借其第一项性质，也可作为收缩算子。作为收缩算子使用时，Krawczyk算子保持不变，而 Hansen-Sengupta 算子则需要稍作修改：在每次迭代（1）中，将 $\boldsymbol{y}_i$ 与 $\boldsymbol{x}_i$ 相交，以获得更紧的包络。此外，区间 Gauss-Seidel 算子可以递归应用，直到进一步改进变得可以忽略不计。可以看出，区间牛顿型算子作用于整个系统，并有可能缩小每个方向上的盒。与前面描述的、不对系统形状施加限制的收缩算子相比，区间牛顿型方法要求系统为方阵。

3.4 求解器策略

上一小节介绍了细分框架的关键组成部分。然而，实际求解器的性能在很大程度上取决于协调这些组成部分所采用的策略。我们已经在第 3.1 节讨论了节点选择策略和检查策略。这里我们重点介绍收缩策略和二分策略。

对于收缩，需要决定以何种顺序应用收缩算子、以多大力度迭代传播，以及何时从收缩切换到分支。对于二分，需要决定如何划分一个盒。虽然二分操作看似简单，但它可能会显著影响整体性能，而且其对效率的影响通常比收缩策略的影响更难预测。

文献中常指出，应尽可能避免使用二分，尤其是在变量数量较多时（Jaulin 等人，2001）。通常只有在万不得已时才会使用二分，即现有收缩算子几乎无法缩小盒，或缩小效果不显著时。

因为沿一个方向进行二分后，往往还需要沿其他方向继续二分，导致复杂度呈指数级增长。

3.4.1 收缩算子策略

正如 Jaulin 等人（2001）所指出的，没有哪种收缩算子能够被视为普遍优于其他收缩算子。每种收缩算子都是针对特定条件设计的。例如，后向-前向传播（基于凸包一致性）通常效果良好，并且有可能沿所有维度缩小一个盒。然而，当某个变量在约束中多次出现时，其效果可能会显著降低。相比之下，基于盒一致性的收缩算子代价较高，并且每次只能缩小一个变量。不过，它们的效果可能优于前向-后向传播。如上所述，这两类收缩算子都是局部的，适合在算法初期使用。然而，仅依赖它们可能导致死锁，即盒无法再进一步缩小。为克服这一局限，通常需要将它们与局部性较弱的收缩算子结合使用，例如 3B 收缩算子、基于区间牛顿法的收缩算子或基于线性松弛的收缩算子。因此，后三类收缩算子的计算开销更大。基于区间牛顿法的收缩算子和基于线性松弛的收缩算子只有在盒足够小或系统近似线性时才有效，因为它们通常需要使用区间雅可比矩阵或类似结构。如果该区间矩阵各元素的宽度较大，收缩算子的效力就会大幅降低。

鉴于这些收缩算子的适用性和计算开销各不相同，采用有效的收缩算子组合策略对于实现较高的算法效率至关重要。一种直接且常用的启发式方法是先应用成本较低的收缩算子，然后再调用成本较高的收缩算子（Vu 等人，2009a；Beelitz 等人，2005年；Granvilliers 和 Benhamou，2001；Benhamou 等人（1999））。具体而言，Benhamou 等人（1999）提出了一种执行约束传播的经典算法：首先采用前向-后向传播，然后仅对约束中多次出现的变量应用盒一致性技术。现有求解器通常也采用类似的启发式方法来组合收缩算子。 `RealPaver`（Granvilliers 和 Benhamou，2006年）中的默认策略将 Benhamou 等人（1999）提出的方法与 Hansen–Sengupta 算子相结合，以提高剪枝效率。在 `IbexSolve` （Chabert 和 Jaulin，2009年）中，默认策略采用一系列收缩算子，包括前向-后向传播器、ACID 收缩算子、区间牛顿型算子和基于线性松弛的收缩算子，通常按上述顺序应用。不同于以往研究中采用的固定收缩算子组合，Araya 等人（2015）提出了一种自适应策略，用于微调每个收缩算子的调用频率。

其核心思想是减少成本高昂的收缩算子的使用，除非它们能够有效缩小最近处理的盒。

除了选择和调度各种收缩算子之外，各个收缩算子内部的实现参数或细节也会对算法效率产生显著影响，尤其是约束传播收缩算子。基于包络一致性或盒一致性的约束传播收缩算子一次处理一个约束。应用于约束集时，它们必须在所有约束间反复传播，直到达到不动盒，也就是无法再进一步缩减为止。实践中，为平衡计算成本与剪枝力度，如果观察不到明显改进，传播过程可能会提前终止。通常根据经验性能选择合适的终止阈值（Baharev 等，2011）。盒一致性收缩算子则更加灵活，因为它们针对约束-变量对进行操作，而且每个约束可能涉及多个变量。不过，将收缩算子应用于所有变量可能代价高昂。为降低这一成本，Goualard 和 Jermann（2008）提出了一种动态选择策略，根据约束-变量对过去在定义域缩减方面的表现，为其分配并持续更新权重，优先处理权重较高的对象。针对单个盒一致性收缩算子，Goldsztejn 和 Goualard（2010）提出了一种自适应剃刀过程，从上下界向内收紧定义域，其表现比原始实现更稳健。Trombettoni 和 Chabert（2007）开展了实验，评估 CID 和 3BCID 收缩算子中不同参数设置对基准测试集整体效率的影响，并据此选择合适的参数。

3.4.2 二分算子策略

到目前为止，我们尚未单独讨论二分算子，因为该操作本身相对简单——沿选定的维度将一个盒分成两个子盒。然而，选择拆分哪个维度仍会对算法效率产生重大影响，而如何确定最优选择仍是一个具有挑战性且尚无定论的问题（Baharev 等，2011；Granvilliers，2012）。

轮询（RR）、MaxDom（或最大优先）和基于 Smear 的启发式方法是最常用的启发式策略。轮询（RR）策略按循环顺序选择变量；它简单、确定，且不依赖任何问题特定信息或启发式方法。MaxDom 策略选择当前盒中定义域最大的变量，旨在尽可能减少变量依赖，从而通过局部一致性技术实现更强的收缩。基于 Smear 的策略利用系统和当前盒的信息，找出对系统影响最大的变量。给定一个盒 $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$，

相对于 f_j ， x_i 的涂抹值为

$$\text{smear}(x_i, f_j) = |\boldsymbol{A}_{ij}| \cdot wid(\boldsymbol{x}_i)$$

其中， $\boldsymbol{A}_{ij}$ 是 $\frac{\partial f_j}{\partial x_i}$ 在 $\boldsymbol{x}$上的区间形式的取值。最大涂抹启发式选择使 $\max_{j=1,\ldots,n}$ (smear(x_i, f_j)) 最大化的变量 x_i ； 涂抹值求和启发式选择使 $\sum_{j=1}^{n}$(smear(x_i, f_j)) 最大化的变量 x_i 。作为改进，Trombettoni 等人（2011）提出了基于相对涂抹准则的启发式方法，即用与 f_j相关的所有涂抹值之和对 smear(x_i, f_j) 进行归一化。研究表明，改进后的启发式方法比原始方法效率更高。Araya 等人（2013）介绍了更复杂的基于涂抹值的启发式方法。

正如 Granvilliers（2012）指出的，细分算法的效率取决于二分策略与收缩过程的配合。基于这样的观察：当盒的尺寸较小时，最大涂抹启发式更有用，Granvilliers（2012）提出了一种自适应策略，在搜索早期优先采用 RR，随后逐渐转向最大涂抹启发式，以期兼顾两种方法的优势。

文献中采用的另一种策略是根据收缩算子的输出选择分裂变量。在应用 Hansen-Sengupta 算子时，区间雅可比矩阵的对角元素可能包含零。除以这样的区间会得到两个不相连、被间隙隔开的区间。这种现象可能表明存在多个解。因此，出现这种情况时，一种合理的分裂策略是利用该间隙进行进一步分支（Hansen 和 Greenberg，1983）。此外，Batnini 等人（2005）指出，由于除以包含零的区间，凸包一致性和盒一致性技术中可能会出现间隙。因此，可以据此设计利用这些间隙的分裂策略。Trombettoni 和 Chabert（2007）也提出了一种类似的分裂策略，该策略利用 CID 收缩算子产生的间隙。

除了经典策略及其变体之外，人们还进行了一些探索性尝试，以设计新颖的二分策略。除了 MaxDom 和基于 Smear 的启发式方法所采用的局部信息外，Reyes 和 Araya（2014年）还尝试利用过去的信息来引导变量选择。这些信息是在预处理阶段通过沿不同方向进行探索性运行收集的。此阶段结束后，会为每个变量分配一个权重，并在搜索过程中将其纳入变量选择启发式方法。Huang 等人（2019年）提出了前瞻分支启发式，试图做出局部最优选择。该方法对每个变量进行穷尽式分裂，以估算不同方向二分所带来的收益，并保留在接下来几步中进展最大的分支。实现中将该策略与 MaxDom 和基于 Smear 的启发式方法进行了比较。有趣的是，实验结果表明，不同的分支方法在.....上的表现大不相同

l较大的实例；然而，没有任何一种启发式方法能够始终优于其他方法 rs.

3.5 其他技术

除了作为细分算法主要组成部分的各种收缩算子外，还有其他技术可以提高其效率。这里我们简要讨论约束系统的有向无环图（DAG）表示，以及符号技术。

如前所述，前向–后向传播是细分算法中的一种基本收缩算子。然而，逐一传播每个约束可能会限制剪枝效率。为解决这一问题，Schichl 和 Neumaier（2005）提出将约束系统表示为有向无环图（DAG），而非若干独立的树。不同约束间的公共子表达式在 DAG 中只出现一次，从而减少依赖效应，进而降低区间高估。这种表示方式能使前向–后向传播产生更紧的收缩。Vu 等人（2009b，a）进一步发展了基于这种表示方式的技术，并证明了其效率。

另一种提高效率的技术是符号计算。正如 Beelitz 等人（2005）以及 Benhamou 和 Granvilliers（2006）所指出的，可以通过为不同约束中共有的每个子项引入一个共享辅助变量，来改写非线性系统；这一思路类似于 DAG 表示，旨在减轻依赖效应并降低区间高估。更进一步，还可以引入额外变量，以减少每个约束中各变量的出现次数（Granvilliers 等人，2001），这通常能带来更紧的收缩，但代价是更高的计算开销。符号计算还可以通过 Gröbner 基或其他消元方法，将多项式系统规范化为零点集相同的等价系统，其思路类似于针对线性系统的高斯消元。所得系统通常更易于计算。然而，Gröbner 基计算代价高昂（最坏情况下具有双指数复杂度），因此如何确定符号预处理的力度本身就是一项不容忽视的设计选择（Benhamou 和 Granvilliers，2006）。

4 数据集描述

我们的数据集由解集为零维的非线性方程方阵系统组成，这类系统尤其适合使用基于细分的求解器。数据集可分为两部分：第一部分包含取自文献和公开数据集的基准测试；第二部分则由一些示例组成，这些示例基于实际应用中出现的若干精选参数系统构造而成。数据集的第一部分包含 581 个系统，其中包括 451 个多项式系统和 130 个非多项式系统。第二部分由根据 5 个族的参数系统生成的示例组成；通过对每个族中的参数进行赋值，我们得到每个族 2,000–30,000 个零维系统。接下来，我们将介绍数据集的结构，然后分别介绍两部分的统计信息，包括系统规模、解的数量和其他显著特征。

通过为每个系列实例化参数，我们可以为每个系列获得 2,000–30,000 个零维系统。接下来，我们将介绍数据集的结构，然后展示各部分的统计数据，包括系统规模、解的数量以及其他显著特征。

4.1 数据集概述

细分数据集（SDD）按以下方式组织。在 SDD 根目录中，有两个目录：非参数和参数。非参数目录包含非多项式目录和多项式目录。每个收集到的非参数系统都分配到一个具有唯一标识符的子目录中，并根据其类型放入上述两个目录之一。每个系统的子目录中包含四个文件：sys.txt 存储输入系统的信息；output.txt 存储三个求解器的计算时间、已认证解和未认证解的数量，以及单元数量（适用于细分求解器）；solution.txt 存储输入系统的解以及用于获得这些解的求解器；info.txt 包含系统的来源信息。

在参数目录中，每个参数系统都分配到一个唯一的子目录，其中包含三个文件：parameter.txt 列出参数的不同取值，parametricSys.txt 描述参数系统，而文件夹实例则存储参数取值对应的特化系统。每个特化系统现在都是一个非参数系统，其组织方式与非参数目录中的系统完全相同。

在台式计算机上，使用两个细分求解器和一个符号求解器求得这些系统的解。软件和硬件的具体信息见表 1。所有求解器的时间限制均设为 1000 秒。IbexSolve 使用选项“ibexsolve -e 1e-6”运行，该选项将二分容差设为 10^{-6}。RealPaver 使用选项“-number +oo -precision 1e-6”运行，使用该选项的原因将在 5.1 节中详细说明。

表 1：硬件和软件信息。

计算机	Intel i9-11900K@3.50GHz，内存 128GB，Ubuntu 20.04
IbexSolve	Release 2.8.9，选项为 “-e 1e-6”
RealPaver	版本 0.4 (c) LINA 2004年，选项为 “-number +oo -precision 1e-6”
RootFinding	Maple 2025

4.2 文献中现有示例的收集

我们报告了数据集第一部分的详细信息和汇总统计数据。这部分由从现有文献中收集的基准测试组成。示例收集自

多种来源，包括 IbexSolve（Chabert 和 Jaulin，2009年）和 Real- Paver（Granvilliers 和 Benhamou，2006年）软件套件、COCONUT（Shcherbina 等人，2002年）、PHCpack（Verschelde，2011年）和 ALIAS 基准测试，以及众多已发表的参考文献，例如 Boulier 等人（2014年）；Lukˇsan 和 Vlcek（1999年）；Stuber 等人（2010年）；Van Hentenryck 等人（1997年）；Wang（1996年）；Vu（2005年）；Xia 和 Zhang（2006年）；Abbott 和 Brent（1975年）；Hou 和 Zhang（2019年）；Mantzaflaris 等人（2011年）；Sherbrooke 和 Patrikalakis（1993年）；Verschelde 等人（1994年）；Kuri-Morales（2002年）；Moore 等人（2009年）；Gawali 等人（2022年）等。收集示例时，我们首先进行了筛选，剔除了明显重复的示例。由此得到表2所列的770个示例。使用细分方法求解这些示例后，我们检查了解的数量相同的示例，并仔细比较输入系统，进一步找出近200个重复示例，其中包括变量名称被改动的示例。最终，我们得到一个包含451 个多项式系统和130 个非多项式系统的数据集。我们认为，这些示例涵盖了文献中报告的绝大多数情况。它们的来源涉及广泛的应用领域，包括经济学、化学反应动力学、电路分析、机械设计、计算机视觉、神经生理学等。

表2：示例来源。

来源		来源	
Name	**示例数**	**名称**	**#Ex.**
IbexSolve 基准测试集[1]	260	PHCpack 数据库[2]	85
COCONUT 基准测试集[3]	101	ALIAS 基准测试集[4]	104
galaad[5]	9	Boulier 等人（2014年）	51
Lukˇsan 和 Vlcek（1999年）	22	RealPaver 基准测试集[6]	18
Stuber 等人（2010年）	17	Van Hentenryck 等人（1997年）	21
Wang（1996年）	16	Vu（2005年）	16
Xia 和 Zhang（2006年）	6	其他	44
		总计	**770 (581)**

表 3 总结了基准测试集中多项式系统按变量数量的分布情况。对于每个变量范围，表中列出了示例数量，以及方程数量和项数的范围、总次数和偏次数、系数大小，以及各求解器求解的实例数量。

表4概述了这130 个非多项式系统。它根据所涉及的非多项式函数类型对系统进行分类，并针对每一类

[1]https://ibex-team.github.io/ibex-lib/[2]https://homepages.math.uic.edu/~jan/demo.html[3]https://arnold-neumaier.at/glopt/coconut/Benchmark/Library3_new_v1.html[4]https://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/benches.html[5]https://www-sop.inria.fr/galaad/data/BASE/2.multipol/centralpos.html[6]https://realpavergroup.univ-nantes.io/realpaver/

表 3：多项式问题的统计数据。

问题特征							已求解实例		
变量	示例	方程	项	总计次数	偏次数	系数范围	IbexSolve	RealPaver	Maple
1–3	93	1-3	1-61	1-21	1-21	$[-1.38 \times 10^{19},\ 1.38 \times 10^{19}]$	92	108	93
4–6	104	4-6	2-32	2-11	1-6	$[-1.92 \times 10^{7},\ 1.59 \times 10^{14}]$	86	75	93
7–10	100	7-10	2-64	1-13	1-7	$[-1.21 \times 10^{15},\ 8.26 \times 10^{14}]$	87	64	70
11–25	76	11-25	2-96	2-30	1-30	$[-200, 220]$	55	32	18
26–50	41	26-44	1-121	2-30	1-3	$[-400, 293.0]$	24	10	4
51–100	13	51-100	1-83	2-3	2-3	$[-80.0, 12.8]$	9	6	0
101–500	13	101-482	1-403	3-4	2-3	$[-400.0, 12.8]$	6	4	0
501–1000	10	501-1000	3-901	2-3	2-3	$[-900.0, 11.8]$	0	1	0
> 1000	1	1001	5-1001	3	3	$[-1000.0, 11.8]$	0	0	0

报告示例数量，以及变量数和项数的范围。

表4：非多项式系统的信息。

Type	# 例题编号	变量数	项数
有理函数	5	1-9	2-7
根式（代数函数）	3	1	3-7
指数函数	38	1-1000	2-6
对数函数	1	1	2
三角函数	49	1-1001	2-1001
绝对值函数	1	2	2
双曲函数	3	1-2	3-4
上述函数的混合形式	30	1-100	2-12

表 5 报告了多项式系统和非多项式系统的已认证解数量分布。只有当 IbexSolve 或 Maple 至少有一个求解器返回精确根数、且不存在可疑区域时，该系统才会列入此表。在基准测试集中，451 个多项式系统和 130 个非多项式系统中，求解器分别成功确定了 358 个和 91 个系统的精确解数。对于其余实例，所有求解器都未能在 1000 秒内认证根数：它们要么都达到时间限制、崩溃或报错，要么输出中仍包含可能隐藏根的可疑区域。

4.3 参数系统

本节介绍数据集中的五类参数系统。表 6 总结了它们的关键信息，包括系统类型以及变量和参数的数量。对于每一类，我们通过为参数赋值，生成一组零维方形方程组。表 6 报告了每一类中 IbexSolve 未能在 1000 秒时间限制内求解的实例数量。对于其余实例，表 6 还报告了平均求解时间。

表 5：解的数量分布。

# 零点	# 多项式	# 非多项式	# 零点	# 多项式	# 非多项式
0	8	11	11–20	35	2
1	80	33	21–50	28	5
2	71	24	51–100	4	2
3	25	5	101–200	8	1
4	26	3	201–1000	3	0
5	8	0	> 1000	1	0
6–10	61	5			

表 6：参数系统摘要。

应用	系统	Type	数量 变量 （方程）	数量 参数	数量 实例	平均值 时间（秒）	数量 超时 实例
多关节 机械臂	平面- 三角函数	非多项式	30–42	8–10	5000	9.33	65
	平面- 多项式	多项式	20–28	8–10	5000	2.84	99
	空间- 三角函数	非多项式	9–15	3–7	3000	9.55	303
	空间- 多项式	非多项式	15–25	3–7	3000	15.6	389
Stewart 平台	Stewart	多项式	9	29	10000	1.04	0
Kuramoto 模型	Kuramoto	多项式	10	5	10000	0.351	0
闪光单元	闪光单元	非多项式	28	4	10000	7.80	0
轨道 确定	轨道	多项式	14	30	2000	358	277

4.3.1 多关节机械臂

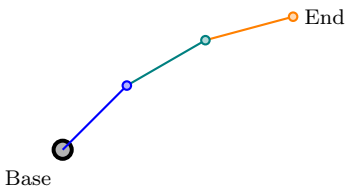


图 3：多关节机械臂。

多关节机械臂广泛应用于机器人学，执行操作、装配和精确定位等任务。分析这类机械臂时，一个基本问题是平面或空间多关节机械臂的定位：给定端点（末端执行器）的坐标，需要确定中间关节的可行构型。这种定位问题自然会导出一组由机械臂运动学约束推导出的非线性方程。

我们关注简单串联多关节机械臂的定位问题。假设该机械臂由 m 个串联连接的连杆组成 L_1，……，L_m 从基座延伸至末端执行器，各连杆的长度分别为 l_1，……，l_m。基座固定在原点。

首先，我们考虑定位问题的平面情形。给定坐标 s

末端连杆端点的坐标（x_m， y_m），目标是确定所有中间关节的可行位置和方向。在此设置下，所得运动学方程系统共有 $3m-2$ 个未知数：对于 $i=1$、.....、 $m-1$，有（x_i， y_i），表示第 i 个连杆的端点坐标；对于 $i=1$、.....、 m，有 θ_i，表示第 i 个连杆相对于正 x 轴逆时针测量的方向。该系统由 $2m$ 个方程组成：

$$P_1 = \begin{cases} x_j - \sum_{i=1}^{j} \ell_i \cos(\theta_i) &= 0, \\ y_j - \sum_{i=1}^{j} \ell_i \sin(\theta_i) &= 0, \end{cases} \qquad j = 1, \ldots, m.$$

坐标 (x_m, y_m) 固定后，该系统包含 $3m-2$ 个变量和 $2m$ 个方程，因此对于 $m \geq 3$，解集通常是正维的。

要得到一个零维系统，我们需要固定 $m-2$ 个变量。随机为这些变量赋值，很可能会导致系统不相容而无解。因此，必须谨慎构造该系统。下面详细介绍系统的生成方式。首先，固定连杆数量 m，并将连杆长度随机选为 1 到 10 之间的整数。接着，通过随机指定 $\theta_i \in [-\pi, \pi]$（对于 $i=1,\ldots,m$），并计算 $i=1,\ldots,m$ 对应的坐标 (x_i, y_i)，构造一个相容系统。现在我们已经得到一个已知的相容系统，接下来通过将最后一根连杆的端点坐标 (x_m, y_m) 以及 $\{x_i, y_i\}_{i=1,\ldots,m-1}$ 中的 $m-2$ 个坐标固定为常数，并将其余变量视为未知数，从而构造一个零维方阵系统。未知数由 $\{\theta_i\}_{i=1,\ldots,m}$ 和 $\{x_i, y_i\}_{i=1,\ldots,m-1}$ 中的 m 个变量组成。共有 $2m$ 个未知数和 $2m$ 个方程。变量的关注区间设为 $x_j, y_j \in [-\sum_{i=1}^{j} \ell_i, \sum_{i=1}^{j} \ell_i]$，以及对于 $j=1,\ldots,m$ 的 $\theta_j \in [-\pi, \pi]$。

在固定 $m-2$ 坐标 $\{x_i, y_i\}_{i=1,\ldots,m-1}$的过程中，必须谨慎选择要固定的变量，以确保所得系统具有零维且非空的解集。基本思路是确保由连续一组连杆构成的任何子系统都既非超定，也非欠定。具体实现如下：1）从集合 $\{x_i, y_i\}_{i=1,\ldots,m-1}$中随机选择 $m-2$ 个坐标。2）将所有选定的坐标按顺序$x_1 < y_1 < x_2 < y_2 < \cdots < x_{m-1} < y_{m-1}$ 排序，排列成有序列表 $\mathcal{L}$。3）对于满足 x_j和 y_j均出现在 $\mathcal{L}$中的每个索引 j，检查 $\mathcal{L}$ 中位于 x_j 之前的元素数量是否等于 $j-2$。如果满足此条件，则由从 1 到 j 的连杆构成的子系统是一个方阵系统；否则，舍弃当前的随机选择。我们重复这一随机选择与检查过程，直到选定的坐标满足上述条件。我们使用 Python 脚本总计生成 5000 个实例，其中 m 的范围为 10 到 14，并且每个 m值生成 1000 个实例。选择 m 的范围时，确保系统的求解时间既不会过长，也不会短得无关紧要。

除了将问题表示为上述非线性系统外，我们还

将同一问题表示为多项式系统，以便比较。只需分别用辅助变量 s_i 和 c_i替换 P_1 中的正弦(θ_i)和余弦(θ_i)，并添加约束 $c_i^2 + s_i^2 = 1$，其中 $i = 1$、.....、 m。生成随机实例的过程与上述相同：首先构造一个相容系统，然后将坐标 x_m、 y_m 和 $m - 2$ 固定为常数，其中坐标取自 $\{x_i, y_i\}_{i=1.....、\ m-1}$ 。唯一的区别是，未知数 $\{\theta_i\}_{i=1.....、\ m}$ 被替换为 $\{s_i, c_i\}_{i=1.....、\ m}$，并添加约束 $\{c_i^2 + s_i^2 = 1\}_{i=1.....、\ m}$。由此得到一个包含 $3m$ 个方程和 $3m$ 个变量的多项式系统。我们生成了 5000 个实例，它们表示与上述相同的问题，只是采用多项式系统的形式。在求解这些系统时，将 c_i 和 s_i 的关注区间设为 $[-1, 1]$。

基准测试集中的每个三角实例都有一个对应的多项式版本：这两种表述形式编码的是同一个底层问题，区别仅在于表示方式。我们使用 IbexSolve 和“ibexsolve -e1e-6”分别求解两个版本，该命令将二分容差设为 10^{-6}。在此配置下，我们分别运行 IbexSolve 求解这两种表述形式，并记录结果。

在这 5,000 个实例中，IbexSolve 报告了 196 种不同的根数，范围为 2 到 1,920。表7汇总了不同连杆数量 m对应的根数分布。我们汇总了三角函数表述和多项式表述的结果：只要 IbexSolve 至少对一种表述形式返回精确根数，该实例就会纳入统计。如果对于两种表述形式，IbexSolve 均未能在 1,000 秒的时间限制内终止，或返回可疑区域，则排除该实例。对于这类实例，所有被排除的情况都是由于两种表述形式均超时。表中，“Exact”列报告可获得精确根数的实例数量，“Min”（分别为“Max”）列报告根的最小（分别为最大）数量，“C”列报告观测到的不同根数的数量。其余各列则报告落在相应根数范围内的实例数量。

表7：平面情形下精确根数的分布。

m	精确	Min	Max	C	1–10	11–50	51–100	101–200	201–500	501–1000	> 1000
10	999	2	272	73	150	583	186	69	11	0	0
11	998	2	556	87	102	498	247	115	34	2	0
12	997	2	896	108	44	440	254	175	71	13	0
13	993	4	1664	107	34	307	271	213	137	28	3
14	998	4	1920	116	21	251	242	233	186	57	8

从表7可以看出，对于大多数实例，IbexSolve 都能可靠地隔离出正确数量的实根。随着 m 增大，多关节机械臂的不同可行构型数量显著增加。

除了平面多关节机械臂，我们还考虑空间多关节机械臂的定位问题。其设定与上述相同，不同之处在于

连杆是在三维空间中而非平面内运动。该问题可表述为

$$P_1' = \begin{cases} x_j - \sum_{i=1}^{j} \ell_i cos(\theta_i) cos(\phi_i) &= 0, \quad for\ j = 1, \ldots, m \\ y_j - \sum_{i=1}^{j} \ell_i cos(\theta_i) sin(\phi_i) &= 0, \quad for\ j = 1, \ldots, m \\ z_j - \sum_{i=1}^{j} \ell_i sin(\theta_i) &= 0, \quad for\ j = 1, \ldots, m \end{cases}$$

其中，(x_i, y_i, z_i) 表示连杆 i 端点的坐标，适用于 $i = 1, \ldots, m$。实际中，(x_m, y_m, z_m) 通常是固定的。于是，该系统有 $5m - 3$ 个变量和 $3m$ 个方程，因此对于 $m \geq 2$，$2m-3$，必须固定其中若干个变量才能得到零维系统。为避免生成很可能不一致的系统，我们首先通过随机赋值给变量 θ_i 和 ϕ_i，构造一个已知系统，适用于 $i = 1, \ldots, m$。接着，我们固定 $i = 1, \ldots, m-1$ 对应的 (x_i, y_i, z_i) 中的 $2m - 3$ 个坐标，并固定端点坐标 $(x_{\text{end}}, y_{\text{end}}, z_{\text{end}})$。由此得到一个含 $3m$ 个变量和 $3m$ 个方程的系统。我们使用 Python 脚本总共生成 3000 个实例，其中 m 的取值范围为 3 到 5，每个 m 值对应生成 1000 个实例。生成实例的流程以及筛选 m 的标准，基本遵循上述方法。

与平面系统一样，我们也为空间系统构造了多项式表述。具体而言，我们用四个变量替换 $sin(\theta_i)$ $cos(\theta_i)$ $sin(\phi_1)$ 和 $cos(\phi_i) st_i$、ct_i、sp_i、cp_i，并添加约束 $st_i^2 + ct_i^2 = 1$、$sp_i^2 + cp_i^2 = 1$，其中$i = 1$、.....、m。我们生成了 3000 个多项式系统，每个系统编码的问题与其三角函数对应形式所编码的问题相同。

与平面情形一样，我们使用 IbexSolve 求解三角函数表述和多项式表述，并记录结果。表8总结了这些系统的根数分布。我们以与平面系统相同的方式汇总两种表述形式的结果。表中，“精确”列报告了可获得精确根数的实例数量。对于 IbexSolve 在两种表述形式下均未能给出精确根数的实例，几乎所有情况都是两种表述形式下均超时。唯一的例外是 $m = 5$ 的一个实例：一种表述形式超时，另一种则正常结束，但产生了未认证输出（即输出中包含未知盒）。右侧各列报告了达到相应根数的实例数量。对于这组实例，根数取值的种类较少，因为与平面情形相比，连杆数量 m 更少。

表 8：空间情形下精确根数的分布。

m	精确	16	32	64	96	128	192	256	320	384	448	512
3	1000	586	414	0	0	0	0	0	0	0	0	0
4	943	0	333	426	48	136	0	0	0	0	0	0
5	844	0	0	148	0	370	51	205	13	20	7	30

我们比较了 IbexSolve 在平面情形和空间情形下求解三角函数表述与多项式表述的性能。在 IbexSolve 成功返回结果的实例中，我们发现，平面情形下多项式表述的平均求解时间更短，而空间情形下三角函数表述的平均速度更快。然而，在这两种情形下，多项式表述出现超时和未认证结果（即运行结束时仍有未知盒）的情况都更多。详情见表9。

表9：IbexSolve 对机器人系统的结果。每个条目从上到下依次报告：**TO** = #(超时)，**NC** =#(未认证)，以及**Avg** =对未超时实例计算的平均时间（秒）。

		平面情形					空间情形		
		m=10	**m=11**	**m=12**	**m=13**	**m=14**	**m=3**	**m=4**	**m=5**
三角函数。	**TO**	8	12	12	22	11	0	84	219
	NC	1	0	0	1	0	1	1	0
	Avg	3.598	5.357	7.942	9.941	19.834	1.230	10.862	18.656
多项式。	**TO**	18	13	18	24	26	23	129	237
	NC	1	2	0	0	1	2	3	4
	Avg	0.630	2.024	2.657	3.903	5.013	6.673	19.626	22.517

由于三角函数实例和多项式实例编码的是同一个底层问题，我们通过比较两种表述形式生成的经认证的解箱数量，交叉验证求解器输出。对于上表中报告的所有实例，两种表述形式的结果一致。

然而，我们发现有少数平面实例（未列入表中）的两种表述形式会得出不一致的结果。例如，对于一个实例，三角函数表述会生成 64 个经认证的解盒，且报告没有可疑区域；而相应的多项式表述既未生成已认证解，也未生成可疑盒。此外，多项式求解在搜索一开始便通过剪枝终止了初始域，期间没有进行任何二分。有趣的是，如果我们手动将初始域二分为两个子盒，并分别求解这两个子问题，那么每个子问题都会返回已认证解，合并后的解数量也与三角函数表述所得结果一致。这种现象表明，求解器在多项式表述上的剪枝存在不稳定性：仅仅将初始盒拆分为两个子盒，就可能改变求解器的结果，尽管实解集合本身并未改变。不过，我们发现，禁用线性松弛组件后，IbexSolve 会得出稳定的结果；这些问题实例的经认证解数量也会与三角函数表述所得结果一致。

4.3.2 斯图尔特平台

如图 4 所示，斯图尔特平台是一种并联机器人机构，由一个刚体构成的动平台通过六根长度可调的连杆连接到固定基座，广泛应用于飞行模拟器、精密加工和运动控制。给定连杆的长度，问题是确定所有可能的构型

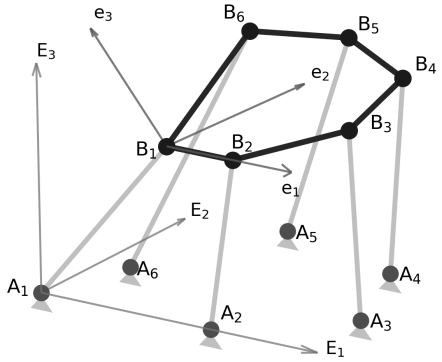


图4：斯图尔特平台。

动平台的姿态。这是机器人学中一个广为人知且十分重要的问题，因为它对并联机构的设计、控制和运动规划至关重要。Dietmaier（1998年）指出，该问题最多可有40个解，因此在相关文献中被用作一项具有挑战性的基准测试集（Baharev等人，2017年）。

这个经典问题可以用多种方式表述。本文采用Dietmaier（1998年）的表述，因为它提供了相对清晰的框架，并便于指定参数。

图 4 展示了斯图尔特平台的结构。基座上固定了一个正交坐标系，其原点为 A_1 ，坐标轴为 E_1、E_2、E_3；另一个坐标系则固定在平台（刚体）上，其原点为 B_1 ，坐标轴为 e_1、e_2、e_3。这里，E_1、E_2、E_3 和 e_1、e_2、e_3 是向量。文献中通常用粗体表示这些符号，但为与前文保持一致，这里仍采用普通（非粗体）记法。选择 E_1 和 e_1 的方向，使点 A_2 和 B_2 分别位于这两个坐标轴上； E_2 和 e_2 的方向则设置为使点 A_3 和 B_3 分别位于由（E_1， E_2）和（e_1， e_2）定义的平面内。当每一对对应点 A_j 和 B_j 之间的距离均为 L_j （即连杆的长度）时，该构型是可行的。为简便起见，我们假设 $L_1 = 1$。图中标出了一个从 A_1指向 B_1 的单位向量 n 。令 a_{ij} （b_{ij}）分别表示 A_i（B_i）在基座（平台）坐标系中的坐标。连杆施加的约束可表述为：

$$\begin{cases} \|n + b_{21}e_1 - a_{21}E_1\|^2 = L_2^2, \\ \|n + (b_{31}e_1 + b_{32}e_2) - (a_{31}E_1 + a_{32}E_2)\|^2 = L_3^2, \\ \|n + (b_{41}e_1 + b_{42}e_2 + b_{43}e_3) - (a_{41}E_1 + a_{42}E_2 + a_{43}E_3)\|^2 = L_4^2, \\ \|n + (b_{51}e_1 + b_{52}e_2 + b_{53}e_3) - (a_{51}E_1 + a_{52}E_2 + a_{53}E_3)\|^2 = L_5^2, \\ \|n + (b_{61}e_1 + b_{62}e_2 + b_{63}e_3) - (a_{61}E_1 + a_{62}E_2 + a_{63}E_3)\|^2 = L_6^2. \end{cases} \tag{4}$$

可令 $E_1 =$(1，0，0) $E_2 =$(0，1，0) $E_3 =$(0，0，1)，然后将 n、e_1 和 e_2表示为基准标架向量 E_1，E_2，E_3的线性组合，其坐标为 $n_j = n \cdot E_j$和 $e_{ij} = e_i \cdot E_j$。通过代入 $e_3 = e_1 \times e_2$消去向量 e_3。在这种表述下，我们得到以下附加约束：

$$\begin{cases} n_1^2 + n_2^2 + n_3^2 = 1, \\ e_{11}^2 + e_{12}^2 + e_{13}^2 = 1, \\ e_{21}^2 + e_{22}^2 + e_{23}^2 = 1, \\ e_{11}e_{21} + e_{12}e_{22} + e_{13}e_{23} = 0. \end{cases} \tag{5}$$

结合 (4) 和 (5)，我们得到一个由 9 个方程组成的多项式系统。我们将

$$(L_2, L_3, L_4, L_5, L_6, a_{21}, b_{21}, a_{31}, a_{32}, b_{31}, b_{32}, a_{41}, a_{42}, a_{43}, b_{41}, b_{42}, b_{43},$$
$$a_{51}, a_{52}, a_{53}, b_{51}, b_{52}, b_{53}, a_{61}, a_{62}, a_{63}, b_{61}, b_{62}, b_{63})$$

视为几何参数，并将

$$(n_1, n_2, n_3, e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23})$$

视为变量。由此得到一个包含 9 个方程和 9 个未知数的系统。

生成随机实例时，我们按如下方式赋予参数值。这里，根据假设，L_1已固定为 1。对于 $i = 2$，.....，6，杆长 L_i从 $[0.5, 2]$中均匀抽样。坐标 a_{21}和 b_{21}分别从 $[0, 2]$，和 $[0, 1.5]$中均匀抽样。这些取值范围旨在涵盖 Dietmaier（1998）中的一种特殊参数配置，该配置下系统有 40 个实解，但在我们随机生成的 10，000 个实例中，并未观察到这一特定配置。

IbexSolve 成功求解了全部 10，000 个生成的实例，以及 Dietmaier（1998）报告的具有 40 个实解的实例，并在 1，000 秒内为每个实例都生成了经认证的结果。表 10 汇总了这 10，001 个实例的根数分布。

表 10：精确根数的分布。

根数	0	2	4	6	8	10	12	14	40
实例	7521	1365	658	303	126	23	3	1	1

4.3.3 Kuramoto 模型

Kuramoto 模型是一种经典模型，用于描述由具有不同固有频率的耦合振子组成的系统，展现了同步如何在振子之间的相互作用中涌现。一般的 Kuramoto 模型如下：

$$\frac{d\theta_i}{dt} = \omega_i - \frac{1}{N}\sum_{j=1}^{N} K_{i,j}\sin(\theta_i - \theta_j), \quad \text{for } i = 1, \ldots, N, \tag{6}$$

其中， N 是振荡器的数量，$K_{i,j}$ 是第 i个与第j个振荡器之间的耦合强度。由于该系统在全局相位偏移下保持不变，通常会固定一个参考角，例如设定 $\theta_N = 0$，然后去掉相应的方程$\frac{d\theta_N}{dt} = 0$。在几种等价的平衡方程表述形式中（Chen 等人，2019），我们采用多项式表述，因为它较为简单。具体而言，我们设定$K_{i,j} = 1$，这相当于假设耦合均匀，是一种常见的简化。然后，对于 $i = 1, \ldots, N-1$,

$$\begin{cases} \omega_i - \dfrac{1}{N}\displaystyle\sum_{j=1}^{N}(s_i c_j - s_j c_i) = 0, \\ c_i^2 + s_i^2 - 1 = 0, \end{cases}$$

其中 $c_N = 1, s_N = 0$。这是一个包含 2($N-1$) 个方程和 2($N-1$) 个未知数的多项式系统。可以通过多种方法求解（Coss 等人，2018）；在本文中，我们重点采用细分法。生成示例时，需要为 $i = 1, \ldots, N-1$ 固定参数 ω_i。为反映实际场景，我们从均值为零、标准差为 0.3 的高斯分布中对每个 ω_i 进行采样，即 $\omega_i \sim \mathcal{N}(0, 0.3_2)$。在此设置下，我们生成了 5,000 个包含 $N = 6$ 的随机实例。每个实例都是一个含有 10 个方程和 10 个变量的方多项式系统。c_i 和 s_i 的关注区间设为 $[-1, 1]$。IbexSolve 成功求解了全部 10,000 个系统，并为每个系统返回经认证的结果。表 11 报告了根数的分布。

表 11：精确根数的分布。

根数	10	12	16	18	20	22	24	26	289	30	32
实例	46	4	15	3	10	809	65	143	464	4180	4261

由于这些实例规模相对较小，且在符号方法的能力范围内，我们使用 Maple 的 RootFinding 包来交叉验证实根的数量。

RootFin ding 对表 1 中报告的各个实例给出的结果一致 1.

不过，我们还发现了另外四个实例（未列于表 11），在这些实例中，Ibex-Solve 报告的已认证解比 RootFinding 少一个，但没有报告可疑/未知区域。这表明这两个求解器之间存在不一致：在这些情况下，IbexSolve 返回的已认证包含区间似乎并不完整。

为调查这一差异，我们通过修改 IbexSolve 的默认收缩算子开展了一系列受控实验。具体而言，禁用线性松弛分量后，IbexSolve 找回了缺失的解，其已认证解数量也与 Maple 在全部四个实例中的结果一致。

一种合理的解释是，线性松弛步骤依赖浮点线性规划计算。尽管这类松弛在实践中可以加强剪枝，但线性规划分量依赖浮点求解器，而浮点舍入误差在少数情况下可能导致数值不稳定的剪枝。文献中也曾指出浮点线性规划与严格区间推理之间的相互作用可能引发类似问题（例如，参见 Trombettoni 等人（2011））。因此，上述四个实例具体表明，在少数情况下，启用基于线性规划的线性松弛可能导致过度激进的剪枝，进而丢失已认证的解包含区间。

4.3.4 闪蒸器

闪光单元是一种分离设备，通过使液体混合物部分汽化来生成汽液平衡物流，常见于化工和石化行业。我们考虑 Bublitz (2023) 中的闪光单元模型，该模型描述了乙醇–水混合物的部分汽化。模型假设液相和气相充分混合并处于热力学平衡状态，且通过供热使单元温度保持恒定。进料条件、压降和操作温度均为已知条件，未知数则包括相组成、流量和热负荷。该问题中的约束涵盖多种衡算，包括相平衡、物料和能量守恒、热力学性质关系，以及闪光单元的几何条件。该问题包含 28 个方程；详情请参阅 Bublitz (2023)。完整的稳态模型由相平衡 (7)、活度系数关系 (8)、物料和能量衡算 (9)、热力学体积关系 (10) 以及几何约束 (11) 组成。

$$
\begin{cases}
K_i x_i - y_i = 0 \ (i = 1, 2), y_1 + y_2 = 1, x_1 + x_2 = 1, x_1^F + x_2^F = 1 \\
h^F - \left(x_1^F h_1^F + x_2^F h_2^F + h^{EF} \right) = 0, \\
h^L - \left(x_1 h_1^L + x_2 h_2^L + h^{EL} \right) = 0, \\
h^V - \left(y_1 h_1^V + y_2 h_2^V + h^{EV} \right) = 0, \\
\Delta p - (p^F - p) = 0, K_i - \gamma_i \, p_i^{\mathrm{sat}} / p = 0 \quad (i = 1, 2).
\end{cases} \tag{7}
$$

$$\begin{cases} \gamma_i - \dfrac{\exp\left[(1-x_i)\left(\frac{\alpha_i}{x_i+\alpha_i(1-x_i)} - \frac{\alpha_j}{\alpha_j x_i+(1-x_i)}\right)\right]}{x_i+\alpha_i(1-x_i)} = 0 \ (i=1,2; j=3-i), \\ \alpha_i - \dfrac{(v_1+v_2)-v_i}{v_i}\exp\left(-\dfrac{\lambda_i}{T}\right) = 0 \ (i=1,2). \end{cases} \tag{8}$$

$$F^F h^F - F^V h^V - F^L h^L + Q = 0, F^F x_i^F - F^V y_i - F^L x_i = 0 \ (i=1,2). \tag{9}$$

$$\begin{cases} U - n^L\left(h^L - p\left(10^5\left(x_1 v_1 + x_2 v_2 + v^{EL}\right)\right)\right) - n^V\left(h^V - R\left(T\, z^V\right)\right) = 0, \\ V - \left(V^L + V^V\right) = 0, \\ n_i - \left(x_i\, n^L + y_i\, n^V\right) = 0, \quad i = 1,2, \\ V^L - \left(\left(v_1 x_1 + v_2 x_2 + v^{EL}\right) n^L\right) = 0, \\ V^V - \dfrac{n^V\, R\left(T\, z^V\right)}{p \cdot 10^5} = 0. \end{cases} \tag{10}$$

$$H^L - \frac{4V^L}{\pi D^2} = 0, A - \frac{\pi D^2}{4} = 0, V - AH = 0, r - \frac{D}{H} = 0. \tag{11}$$

该系统中的变量为 $K_1, K_2, x_1, x_2, x_2^F, y_1, y_2, \gamma_1, \gamma_2, \alpha_1, \alpha_2, p, h^F, h^L, h^V, F^L, F^V, Q, n_1, n_2, n^L, n^V, U, V^L, V^V, V, A, r.$。其中，$K_i$为气液平衡比；$x_i$和 y_i 分别为分量 i在液相和气相中的摩尔分数；x_2^F 为分量 2 的进料组成。量 γ_i 和 α_i 分别为活度系数及其温度相关参数。标量 p 为操作压力。符号 h^F、h^L、h^V分别表示进料、液相和气相的比焓；F^L和 F^V分别为液相和气相出口流量；Q为热负荷。变量 n_i 表示容器中分量 i 的持液量；n^L 和 n^V 分别为液相和气相总持液量。量 U 为总内能。变量 V^L、V^V和 V 分别表示液相、气相和总体积。最后，A 为容器横截面积，r 为直径高度比。

系统中的其余符号均视为参数。为生成实例，我们根据 Bublitz (2023) 中的数据，固定以下与操作条件、热物性常数或几何数据相关的参数：闪蒸温度$T = 353.15\,\mathrm{K}$ 和进料温度 $T^F = 353.15\,\mathrm{K}$；焓偏移项 h^{EF}, h^{EL},h^{EV}（在当前设定中取 0）；容器直径 $D = 0.16$、特征高度$H = 0.5$和参考液位高度 $H^L = 0.25$；过量液体体积贡献$v^{EL} = 0$；以及气相压缩因子 $z^V = 1.0$。热力学关联式通过各组分的焓和饱和压力参数进行编码。对于每个组分 $i \in \{1,2\}$，液相、气相和进料相焓 h_i^L, h_i^V, h_i^F根据表达式计算，其系数在实现中以数值形式给出

（详情见 Bublitz (2023) 第187页），饱和压力 p_i^{sat}也由此计算。参数 $(\lambda_1, \lambda_2) = (95.68, 506.7)$ 为经验常数，液体摩尔体积 $(v_1, v_2) = (5.869 \times 10^{-5},\ 1.807 \times 10^{-5})$ 均采用一致的单位。最后，气体常数为 $R = 8.314$，它参与蒸气持液量和体积关系的计算。

除固定参数外，还有若干操作输入由用户指定：压降 Δp、进料压力 p^F、组分1的进料组成 x_1^F 以及进料流量 F^F。为生成基准实例，我们在预定义范围内以固定步长系统地改变这些用户指定参数。通过在网格上组合这些变化，我们总共获得了 10,000 个问题实例。参数范围的选择参考了 Bublitz (2023) 中报告的数值，并扩展至涵盖具有实际意义的操作条件。每个变量的感兴趣区域设定为 $[-10^9, 10^9]$，与 Bublitz (2023) 中的设定一致。

IbexSolve 成功为全部 10,000 个实例生成了输出。然而，这些输出均未得到完全认证：每个实例都至少包含一个未知盒。除未知盒外，10000 个实例中还有 3436 个包含已认证的解盒，其中 1956 个实例生成了一个解盒，1290 个生成了两个，190 个生成了三个。所有实例中普遍存在未知盒，似乎主要源于两个因素。首先，初始搜索域极其宽泛。其次，这些系统具有高维度和强非线性，同时包含多元多项式、指数项和有理表达式。当定义域较宽、使分母可能接近零时，这些特征会使严格的基于区间的认证变得更加困难。从经验来看，一旦将初始定义域收紧至具有物理意义的范围，未知盒的出现频率就会大幅降低。

4.3.5 初始轨道确定

初始轨道确定（IOD）根据最少的观测数据估计天体的开普勒轨道——通常只有角度（视线），且没有先验信息。从几何上看，这归结为寻找一条具有指定焦点、并与观测视线相交的圆锥曲线。这里我们考虑以下问题：在 $\mathbb{R}^3$ 中寻找一条焦点位于原点、且经过五条给定的一般位置直线的圆锥曲线。该问题可表述为一个非线性方程方阵系统。与其他实际问题一样，该问题有多种等价的代数重构形式（Duff 等人，2022；Huang 等人（2025））；这里我们采用 Huang 等人（2025）的表述形式。

这里简要介绍已知数据、变量和约束。五位观测者的位置由已知矩阵 3×5 matrix P 给出，其视线（单位方向向量）由已知矩阵 3×5 matrix U 给出。我们将 $w = (w_1,\ w_2,\ w_3) \in \mathbb{R}^3$作为轨道平面的单位法向量，并基于对称性假设 $w_1 \geq 0$。由此引入三个变量 $w_1,\ w_2,\ w_3$，其感兴趣区域为 $w_1 \in [0, 1],\ w_2,\ w_3 \in [-1, 1]$，以及一个

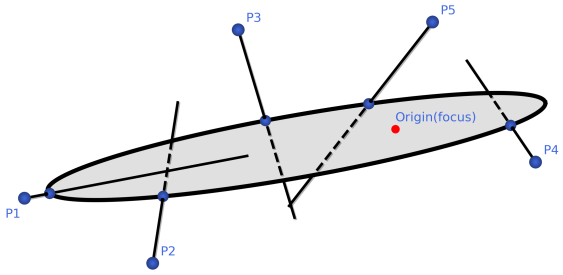


图5：几何设置：点 P_i 表示观测者位置，线条表示观测到的视线。

约束 $\|w\|^2 = 1$。为了在轨道平面上构成局部正交规范标架，我们还需要添加两个单位向量 v， $v' \in \mathbb{R}^3$。按照 Huang 等人（2025）的做法，令 $e_1 = (1, 0, 0)$，设 $v' = \lambda'(w \times e_1)$，其中 $\lambda' = \frac{1}{\|w \times e_1\|}$，并定义 $v = v' \times w$。此处引入 λ' 作为一个变量，其感兴趣区域为 $[1, \infty)$；相应的约束为 $\lambda' = \frac{1}{\|w \times e_1\|}$。

我们为每条视线与轨道平面的交点引入标量变量 ρ_i。令 p_i 和 u_i 分别为 i的第 i 列，并定义

$$r_i = p_i + \rho_i u_i, \quad i = 1, \ldots, 5$$

满足共面条件 $r_i \cdot w = 0$，其中 $i = 1, \ldots, 5$，因为轨道平面内的向量与法向量 w正交。这引入了五个变量 $\{\rho_i\}_{i=1}^5$ 和五个线性约束。

在局部正交规范标架中，圆锥曲线（归一化后）的形式为 $ax^2 + bxy + cy^2 + dx + ey + 1 = 0$，包含五个变量 a, b, c, d, e。将交点r_i （在此标架中表示）代入圆锥曲线方程，可得到五个约束，每个$i = 1, \ldots, 5$对应一个约束。

焦点位于原点这一要求会带来两个额外的约束（Semple 和 Kneebone，1998年）：

$$e^2 - d^2 + 4a - 4c = 0, \quad de - 2b = 0.$$

这些方程合在一起，构成如下包含 14 个变量的方多项式系统：

$$\begin{cases} \|w\|^2 = 1, \\ \lambda'^2 \cdot \|w \times e_1\|^2 = 1, \\ w \cdot (p_i + \rho_i u_i) = 0, \quad i = 1, \ldots, 5, \\ a\,x_i^2 + b\,x_i y_i + c\,y_i^2 + d\,x_i + e\,y_i + 1 = 0, \quad i = 1, \ldots, 5, \\ e^2 - d^2 + 4a - 4c = 0, \\ de - 2b = 0, \end{cases} \tag{12}$$

其中 $w = (w_1, w_2, w_3)$， $e_1 = (1, 0, 0)$， p_i和 u_i分别是 i阶的列 o f

矩阵 P和 U，坐标 $x_i = (p_i + \rho_i u_i) \cdot v$, $y_i = (p_i + \rho_i u_i) \cdot v'$ 与 $v' = \lambda'(w \times e_1)$，以及 $v = v' \times w$。在这个系统中，矩阵 P和 U 是已知数据，未知数为 w_1，w_2，w_3，λ'，$\rho_i (i = 1, \ldots, 5)$ a，b，c，d，e。

生成实例时，我们通过从均匀盒 $[-10, 10]^3$ 中独立同分布地抽取 $p_i \in \mathbb{R}^3$，并将其作为列堆叠起来，构造位置矩阵 $P \in \mathbb{R}^{3\times5}$ P。方向矩阵 $U \in \mathbb{R}^{3\times5}$ 的构造方法是从标准正态分布中独立抽取向量$z_i \in \mathbb{R}^3$，将其归一化为 $u_i = z_i / \|z_i\|$，再将单位向量 u_i 作为列堆叠成矩阵 U。采用此方法，我们生成了 2,000 个实例。

这些实例颇具挑战性：使用 IbexSolve 求解时，大多数实例都会超时。为加快搜索，我们针对每个实例 $b^2 - 4ac < 0$，在原方阵系统中增加一个不等式，以限制轨迹为椭圆形。在此设置下，IbexSolve 为 1,723 个实例求得了经认证的结果，其余 277 个实例则超时。表 12 报告了这 1,723 个实例的根数分布。

表 12：根的数量分布。

根数	0	1	2	3	4	5	6	7	8	9	10	11	13
实例	118	182	315	275	294	222	145	87	45	26	11	2	1

5 数据集的应用

5.1 作为基准数据集

为了比较不同求解器的性能并收集基准测试的解信息，我们在 IbexSolve、RealPaver 和 Maple 上运行了这些示例。我们将报告并比较这些求解器的结果。

我们首先考察两个基于细分的求解器 IbexSolve 和 RealPaver。这两个求解器虽然采用相同的细分框架，但在多个方面有所不同，例如收缩策略、二分策略、节点选择策略以及其他实现细节，这些差异会影响效率，有时也会导致结果不同。
IbexSolve 默认采用深度优先搜索。其默认收缩算子由多种技术组合而成，包括包一致性技术、Hansen–Sengupta 算子、ACID 收缩算子以及基于线性松弛的收缩算子。默认的二分策略是相对扩散和（RSS）策略。对于零维方形方程组，IbexSolve 会细分整个感兴趣区域，并输出解盒（经验证包含唯一解）以及可疑盒（既无法排除，也无法确认其唯一性）。RealPaver 使用一种复合收缩算子，

该算子结合了凸包一致性和盒一致性技术、Hansen–Sengupta 算子以及 3B 一致性方法。默认情况下，RealPaver 执行深度优先搜索，并采用一种将轮询与最大收缩规则相结合的二分策略（选择雅可比列绝对值之和最大的变量；如果雅可比矩阵不可用，则退回轮询）。然而，在我们的实验中，如果不限制盒的数量，该策略会导致 18 个实例产生超过 1 GB 的输出，表明在此设置下发生了输出爆炸。因此，我们改用铺砌模式，该模式采用广度优先搜索和最大优先二分策略。除这些异常情况外，铺砌模式与默认模式的整体性能相当。综合考虑后，我们报告采用铺砌模式得到的结果。除非另有说明，以下所有 RealPaver 结果均采用铺砌模式。RealPaver 会输出所有达到规定精度且无法被其收缩算子剪枝的盒；与 IbexSolve 不同，在大多数情况下，它不会确认并标记盒中包含唯一解，即使内部可能使用验证型算子（例如 Hansen–Sengupta 算子）作为收缩算子。为确保公平，我们将两个求解器的二分精度设为相同值，即 10^{-6}，并将超时时间设为 1000 秒。

在 581 个实例中，RealPaver 有 1 次运行因语法无法识别而失败，IbexSolve 有 2 次运行因段错误（可能是软件缺陷）而失败。在其余实例中，IbexSolve 在 1000 秒内求解了 459 个；其中，393 个仅生成了已认证盒（即没有可疑区域）。RealPaver 在相同的时间限制内求解了 385 个实例。

表 13：不同时间区间内求解的示例数量。

时间（秒）	≤ 1	1-10	10-100	100-1000	超时（>1000）
IbexSolve	288	83	47	41	120
RealPaver	268	49	37	31	195

表 13 统计了在不同时间区间内求解成功的示例数量。图 6 进一步比较了 IbexSolve 与 RealPaver 的累计运行时间，其中不包括超时的实例。从表 13 和图 6 可以看出，从统计结果来看，IbexSolve 的速度明显快于 RealPaver。

在 581 个实例中，只有 IbexSolve 求解成功的有 102/581（17.6%），只有 RealPaver 求解成功的有 28/581（4.8%）——即一个求解器成功求解，而另一个超时或出错。两者均求解成功的有 357/581（61.4%）；在这些实例中，排除双方均在 1 秒内求解成功的 241 个简单案例后，IbexSolve 在 85 个案例中（74.6%）速度更快，RealPaver 则在 29 个案例中（25.4%）速度更快。

除了基于细分的求解器外，我们还使用基于符号计算的 `RootFinding[Isolate]`在 Maple 中运行了这些实例。由于符号方法仅适用于多项式系统，我们将 Maple 实验限制在基准测试集的多项式子集中。 `RootFinding[Isolate]`的输出是

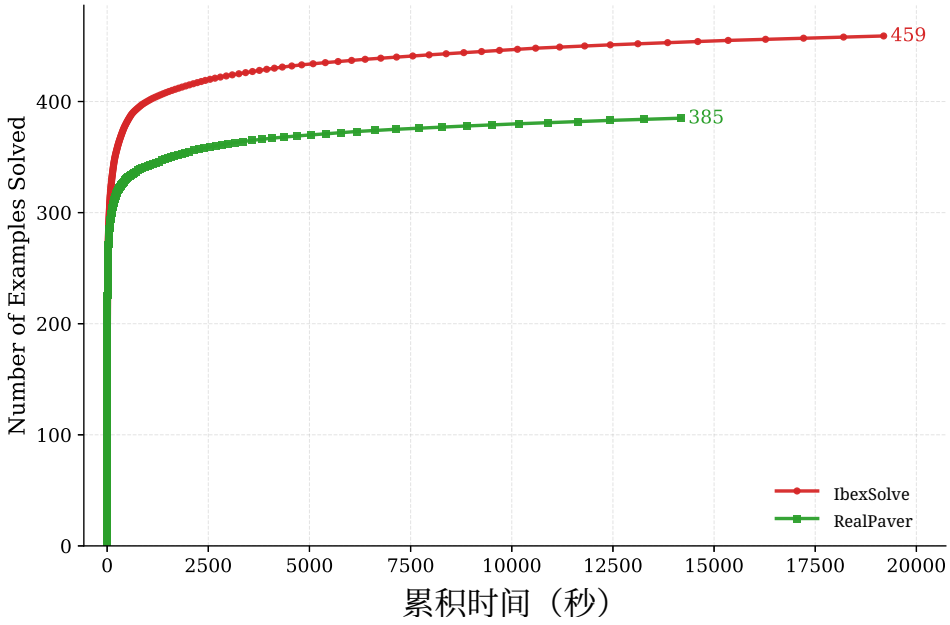


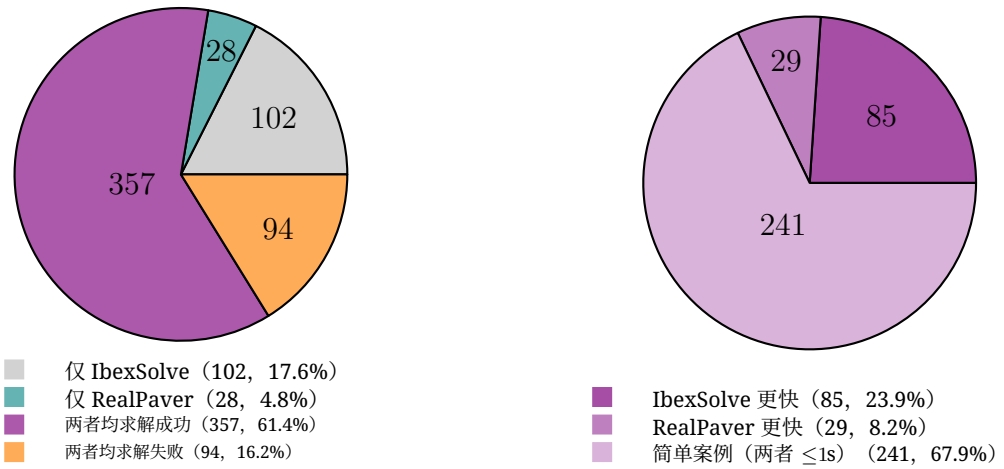
图 6：IbexSolve 与 RealPaver 的累计运行时间比较。

输入多项式系统实解的隔离区间/盒列表。该程序旨在计算所有实解，不支持将搜索限制在用户指定的感兴趣区域内。在 451 个多项式实例中，Maple 为 278 个实例求得了解；其余实例要么超时，要么因内核崩溃而终止，要么因系统定义了正维复代数簇而不适用。

表 14：Maple 中 RootFinding 的时间分布（秒）。

时间	≤ 1	1-10	10-100	100-1000	超时（>1000）	错误
数量	201	40	19	18	156	17

图8展示了 IbexSolve、RealPaver 和 Maple 在多项式子集上的累计运行时间比较。总体而言，在我们的实验中，IbexSolve 的表现优于 RealPaver 和 Maple 的 `RootFinding[Isolate]`，而后两者的表现相当。另一方面，正如 IbexSolve 与 RealPaver 的比较所示，没有任何一个求解器在所有实例上都始终快于另一个求解器。因此，我们分别将 Maple 的 `RootFinding[Isolate]` 与 IbexSolve 和 RealPaver 进行比较。在 451 个实例中，只有 IbexSolve 成功求解的有 104/451 个（23.1%），只有 Maple 成功求解的有 23/451 个（5.1%），即一个求解器成功求解，而另一个超时或出错；在仅由 IbexSolve 求解的 104 个实例中，80 个结果为精确，24 个为非精确。两者都成功求解了 255/451 个（56.5%），其中 IbexSolve 在 222 个案例中得到精确结果，在 33 个案例中得到非精确结果。排除双方均在 1 秒内求解的 146 个简单案例（IbexSolve 对其中 119 个得到精确结果，27 个得到非精确结果）后，IbexSolve 在 51 个案例中更快（46.8%；全部为精确结果），Maple 在 58 个案例中更快（53.2%；其中 IbexSolve 对 52 个得到精确结果，对 6 个得到非精确结果）。



（a）581 个实例的覆盖率　　（b）两者均求解成功案例的细分（355）

图7：IbexSolve 与 RealPaver：时间比较。

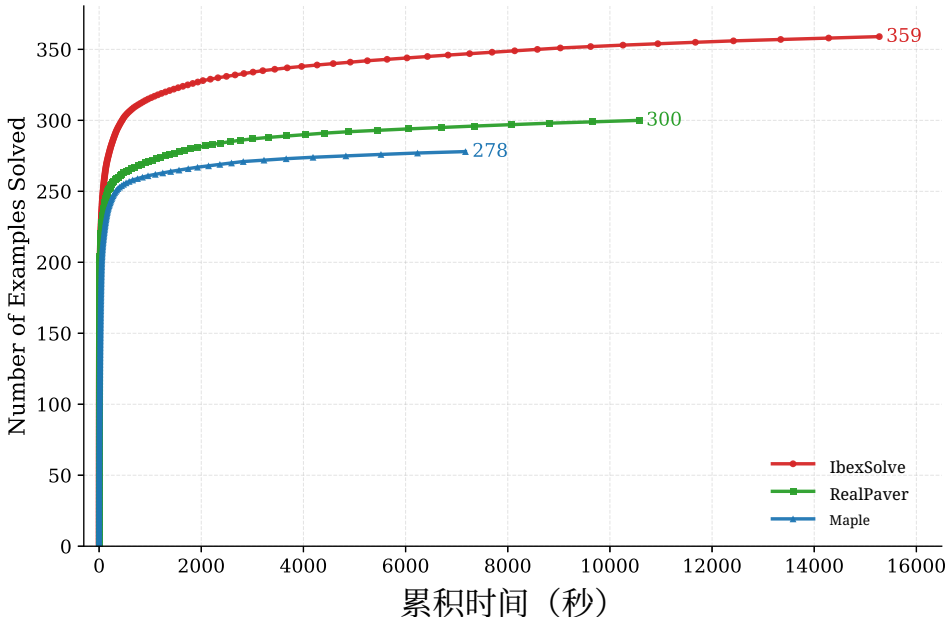
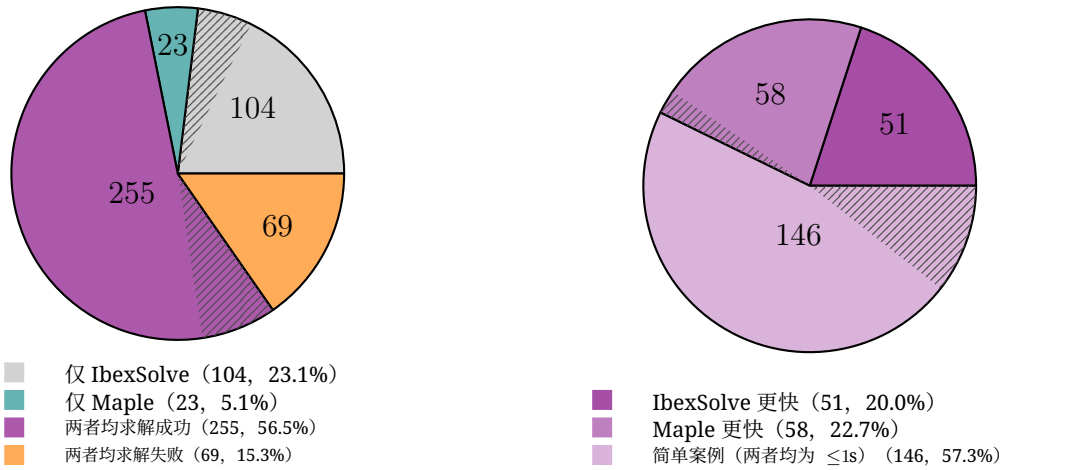


图8：累计运行时间比较：IbexSolve、RealPaver 和 Mapl　　e

在 451 个实例中，只有 RealPaver 成功求解的有 88/451 个（19.5%），只有 Maple 成功求解的有 66/451 个（14.6%）。两者都成功求解了 212/451 个（47.0%）；排除双方均在 1 秒内求解的 126 个简单案例后，RealPaver 在 38 个案例中更快（44.2%），Maple 在 48 个案例中更快（55.8%）。

为评估不同方法之间的一致性，我们比较了 IbexSolve 和 Maple 返回的解。此阶段不考虑 RealPaver，因为它不会认证根（例如，它无法证明唯一性）。根据我们的实验，我们发现有 31 个实例中，Maple 在感兴趣区域内分离出的解比 IbexSolve 更多。不过，这并不意味着 IbexSolve 的搜索漏掉了解：在每个此类实例中，Maple 额外找到的解都落在 IbexSolve 的可疑（未知）盒内。对这些未经认证的解处的



（a）451 个实例的覆盖率　　（b）两者均求解成功案例中的细分情况（255）

阴影区域表示 IbexSolve 输出未经认证的实例

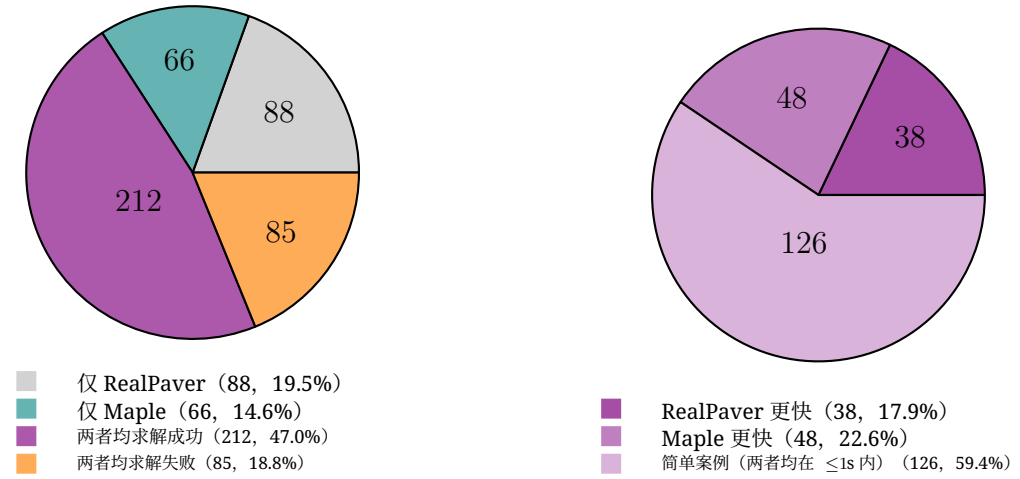
图9：IbexSolve 与 Maple：时间对比

雅可比矩阵求值可得 $\det(J) \approx 0$，表明存在多重根。因此，IbexSolve 未能认证唯一性，最可能是由于根的重数，这与其认证过程相符。从这个意义上说，这两种工具的输出彼此一致。

从上述对比可以看出，IbexSolve 通常比 RealPaver 更高效，而 RealPaver 往往比 Maple 更快。不过，这种排序仅体现统计趋势：对于各个单独的实例，没有任何一个求解器始终比其他求解器更快。这一观察也印证了我们先前关于细分策略的讨论。对于两种基于细分的工具 IbexSolve 和 RealPaver，总体而言 IbexSolve 通常更快。不过，在数据集中的一些高维系列中，例如 Yamamura 和 Trigo系列（变量数多达数千个），RealPaver 表现出色，往往能成功求解 IbexSolve 无法求解的问题。与符号工具 RootFinding 相比，基于细分的求解器在变量数较多、但解集相对规整的问题上通常更具优势，尤其是在解的数量较少时。相反，对于代数结构复杂的低次系统，特别是存在多重根时，符号方法往往更为适用。总体而言，这些结果表明，求解器性能高度依赖于具体实例，这说明采用特征驱动策略可以有效提升效率。

通过在我们收集的基准测试集和生成的实例上运行 IbexSolve，我们发现了若干问题，这些问题表明 IbexSolve 可能存在实现错误，或者至少还有改进空间。总体而言，我们将这些观察结果分为四种类型：其中两种已找到合理的原因，另外两种仍需进一步仔细调查。

第一个问题与线性松弛技术的使用有关，它可能



（a）451 个实例的覆盖率　　（b）两者均求解案例的细分（212）

图 10：RealPaver 与 Maple：时间比较

导致结果未能得到完全认证。我们已在 4.3.3 和 4.3.1 中讨论过这一点，而我们的实验也为此问题提供了具体证据。第二个问题涉及少数几个基准测试集，在这些测试中，IbexSolve 报告的解比 Maple 更多。仔细检查后发现，这在很大程度上是由于 IbexSolve 在最终验证阶段进行的盒膨胀所致。如果两个相邻的盒都经过检查，且某个解靠近它们的公共边界，那么这两个盒都可能被认证为包含一个（唯一）解，从而导致同一个解被重复计数。幸运的是，通常可以通过简单的后处理程序纠正这种重复计数。第三个问题涉及 4.3.1 中的空间系统。对于少数几个实例，尽管三角函数表述和多项式表述描述的是同一个底层问题，但两者得出的解并不一致。排除上述前两种解释后，这种不一致仍然存在，我们尚未找出其原因。最后，我们还发现了第四个问题：IbexSolve 在两个基准测试集上因段错误而崩溃，其中一个测试集相当简单，本应很容易求解。尽管这些只是个别情况，但它们可能表明实现中潜藏着一个错误。

另一项观察来自于在 4.3.1 节的机器人系统上运行 IbexSolve。我们发现，如果将二分容差放宽至 10^{-3}（而非前面实验中始终采用的默认值 10^{-6}），那么在相同的 1000 秒时间限制内，IbexSolve 实际上能为更多实例给出经认证的结果，不包括那些超时或仍包含可疑区域的实例。对于 4.3.1 节中的全部四类系统，这一现象都一致存在。这个结果多少有些违背直觉，因为人们往往认为更高的精度能够提供更多信息，从而更有助于认证。一种可能的解释是，较严格的容差会显著增加二分和传播的次数，导致留给最后认证步骤的预算减少；相比之下，稍微宽松一些的容差可能会降低搜索开销，使更多实例能够得到

经认证的结论。这一发现或许能为通过选择合适的二分容差来提高求解器效率提供实用启示。

5.2 用于训练机器学习模型

在本节中，我们使用 Kuramoto 模型来展示该数据集在训练实根分类模型方面的实用性。为此，我们采用了通过 Python 的 scikit-learn 库实现的三种机器学习方法：K近邻（KNN）、支持向量机（SVM）和随机森林。这些模型以给定的参数值作为输入，并预测 Kuramoto 模型的实解数量。Kuramoto 模型的数据集包含 10,000 个示例，并按 8:1:1 的比例划分为训练集、验证集和测试集。测试集上的实验结果如图 11 所示。为了便于可视化，结果被投影到参数空间（w_1，w_4）中。从图中可以看出，KNN 模型能够以很高的准确率（93.3%）对参数化 Kuramoto 模型的实根数量进行分类。我们在 Kuramoto 系统的验证集上对三种基线机器学习模型进行了超参数调优。KNN 模型在 k=2 时取得了最佳表现，准确率达到 93.3%。SVM 模型使用径向基函数核并设置惩罚参数 C=10 时，取得了最高的准确率 55.1%；其中 C 决定了间隔最大化与训练误差惩罚之间的平衡。对于随机森林，最优设置为 n estimators=40；其中 n estimators 表示集成中的树木数量，该配置的准确率达到 89.8%。

6 结论与未来工作

在本研究中，我们全面搜索了有关细分的文献和代码仓库，收集具有有限个实解的非线性方程。我们识别并删除了来自不同资源的重复示例，调用最先进的细分求解器计算实解，并进一步使用符号求解器验证这些结果。因此，我们构建了迄今规模最大的、用于细分的零维非线性方程带标签实数数据集。我们还选取了来自不同应用领域的 5 个族非线性参数系统，从参数范围内随机抽样取值，并将这些参数值代入系统，生成了超过 48K 个零维非线性方程；随后调用 IbexSolve 计算实解，以扩充数据集。

我们证明，该数据集既可用于对不同求解器进行基准测试，也可用于训练机器学习模型。在构建该数据集时，大多数示例都已由最先进的细分求解器 IbexSolve 求解。在开发

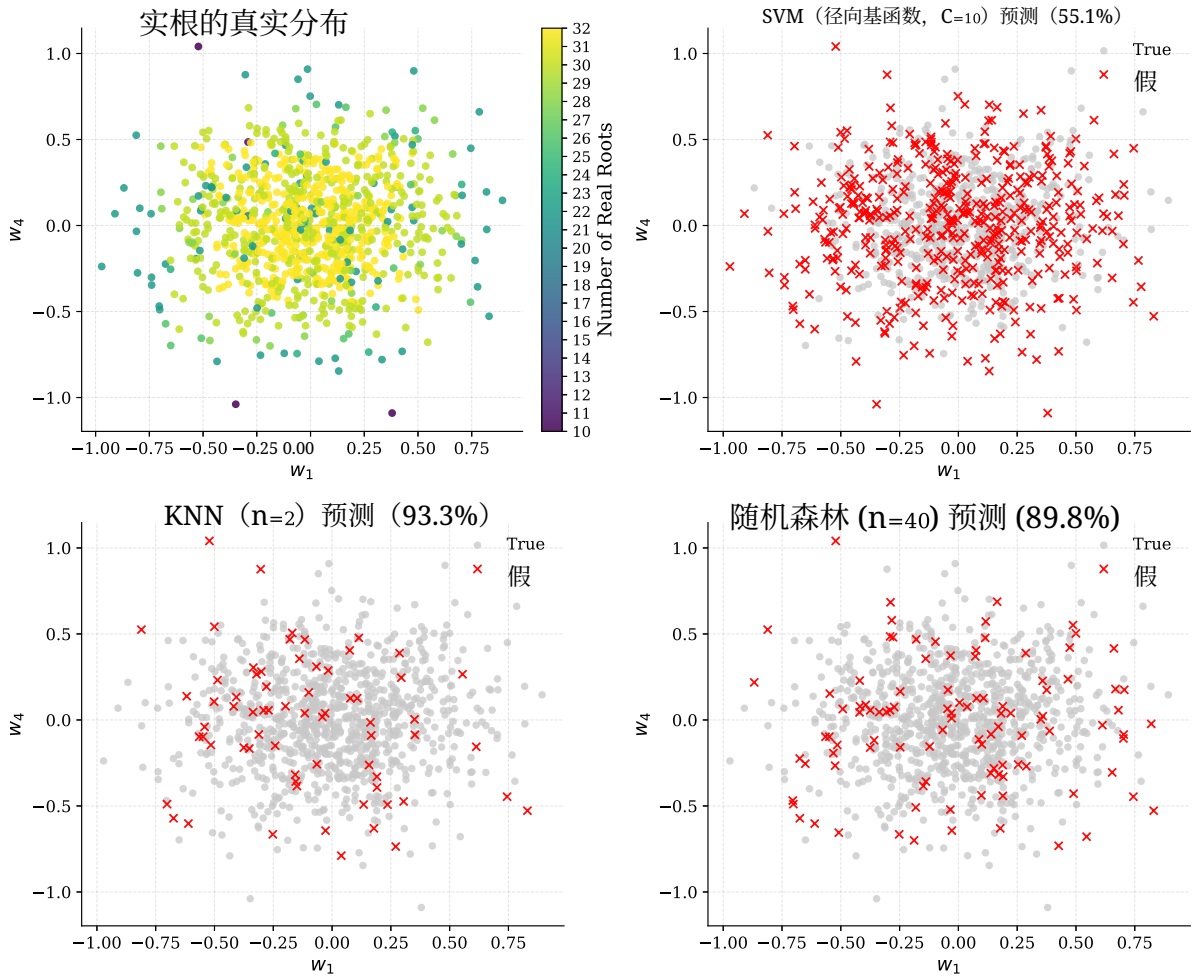


图11：模型在测试集上的性能可视化。

该数据集的过程中，我们还发现 IbexSolve 默认策略存在若干弱点，这为未来开发更高效的启发式策略或基于学习的策略创造了机会。

致谢

作者　谨此感谢 J¨urgen Gerhard 帮助我们获得 Maple 许可证。

资助信息

本研究得到以下项目资助：中国国家重点研发计划项目（项目编号：2023YFA1009402）、国家自然科学基金委员会项目（项目编号：12301650）以及重庆英才计划青年拔尖项目（项目编号：2021000263）。

数据可用性声明

支持本研究结论的数据已在 Science Data Bank 上公开，网址为 `https://doi.org/10.57760/sciencedb.34211`。使用这些数据时，请注明数据来源。

参考文献

James P Abbott 和 Richard P Brent. 非线性方程的单步和多步方法的快速局部收敛。*ANZIAM*期刊，19(2):173–199，1975年。

Ignacio Araya 和 Victor Reyes. 用于优化和约束满足的区间分支定界算法：综述与展望。*Journal of Global Optimization*，65:837–866，2016年。

Ignacio Araya、Gilles Trombettoni 和 Bertrand Neveu。基于凸区间泰勒的收缩算子。载于约束编程中的人工智能（*AI*）与运筹学（*OR*）技术集成国际会议，第1–16页。施普林格，2012年。

Ignacio Araya、Victor Reyes 和 Cristian Oreallana。用于数值约束满足问题的更多基于 smear 的变量选择启发式方法。载于*2013*年第*25*届*IEEE*人工智能工具国际会议，第1004–1011页。IEEE，2013年。

Ignacio Araya、Ricardo Soto 和 Broderick Crawford。数值约束满足问题的自适应过滤策略。《专家系统与应用》，42(21)：8086–8094，2015年。

Ignacio Araya、Fr´ed´eric Messine、Jordan Ninin 和 Gilles Trombettoni。在基于区间的求解器中混合两种线性松弛技术。*Journal of Global Optimization*，91(3)：437–456，2025年。

Ali Baharev、Lubomir Kolev 和 Endre R´ev。计算均相共沸蒸馏和理想双产品蒸馏中的多个稳态。《*AIChE*期刊》，57(6)：1485–1495，2011年。

AliBaharev、Ferenc Domes 和ArnoldNeumaier。寻找稀疏非线性方程组所有良好分离解的稳健方法。数值算法，76 (1)：163–189，2017年。

Heikel Batnini、Claude Michel 和 Michel Rueher。注意间隙：一种用于一致性技术的新型分裂策略。载于< शैली id='1'>约束编程原理与实践国际会议，第77–91页。施普林格，2005年。

Ruben Becker、Michael Sagraloff、Vikram Sharma 和 Chee Yap。一种基于Pellet检验和牛顿迭代的近最优复根隔离细分算法。符号计算杂志，86：51–96，2018年。

Thomas Beelitz、Andreas Frommer、Bruno Lang 和 Paul Willems。求解非线性系统的符号数值技术。载于*PAMM*：应用数学与力学论文集，第 5 卷，第 705–708 页。Wiley在线图书馆，2005年。

Fr´ed´eric Benhamou 和 LaurentGranvilliers。连续约束与区间约束。人工智能基础，2：571–603，2006年。

Fr´ed´eric Benhamou、Fr´ed´eric Goualard、Laurent Granvilliers 和 Jean-Fran˛cois Puget。修订包一致性和盒一致性。载于逻辑编程：*1999*年国际逻辑编程会议论文集，第 230 页，1999年。

Fran˛cois Boulier、Changbo Chen、Fran˛cois Lemaire 和 Marc Moreno Maza。正则链的实根隔离。载于 Ruyong Feng、Wen-shin Lee 和 Yosuke Sato 编，《 style id='3'>计算机数学，第 33–48 页，柏林、海德堡，2014年。Springer Berlin Heidelberg。

Saskia Bublitz。 针对化学过程模型测试的非线性代数方程组求解改进：初始化与重构策略。博士论文，柏林工业大学（德国），2023年。

St´ephane Caro、Damien Chablat、Alexandre Goldsztejn、Daisuke Ishii 和 Christophe Jermann。用于计算并联机器人的广义构型的分支剪枝算法。人工智能，211：34–50，2014年。

Gilles Chabert 和 Luc Jaulin。收缩算子编程。人工智能，173(11)：1079–1100，2009年。

Tianran Chen、Jakub Mareˇcek、Dhagash Mehta 和 Matthew Niemerg。Kuramoto 模型作为多项式方程系统的三种表述。发表于*2019*年第*57*届*Allerton*通信、控制与计算年会（*Allerton*），第810–815页。IEEE，2019年。

Raphael Chenouard、Alexandre Goldsztejn 和 Christophe Jermann。《分支剪枝算法的随时可用搜索策略》。载于 国际人工智能联合会议，第 p–468 页，2009年。

H´elene Collavizza、Fran˛coisDelobel 和 Michel Rueher。比较部分一致性。*Reliable Computing*，5(3)：213–228，1999年。

Owen Coss、Jonathan D Hauenstein、Hoon Hong 和 Daniel K Molzahn。定位并计数秩一耦合 Kuramoto 模型的平衡点。*SIAM*应用代数与几何期刊，2(1)：45–71，2018年。

David Cox、John Little、Donal O'shea 和 Moss Sweedler。《理想、簇与算法》。施普林格，1997年。

Thomas Richard de Latour、Rapha¨el Chenouard 和 Laurent Granvilliers。数值约束满足问题的随时求解策略。载于法语 ´约束编程会议，2021年。

Peter Dietmaier。一般几何构型的 Stewart-Gough平台可能有40种实数姿态。载于《机器人运动学进展：分析与控制》，第7–16页。施普林格，1998年。

Ferenc Domes 和 Arnold Neumaier。二次约束上的约束传播。《Constraints》，15：404–429，2010年。

Ferenc Domes 和 Arnold Neumaier。使用线性松弛进行严格筛选。《Journal of Global Optimization》，53：441–473，2012年。

Timothy Duff、Michela Mancini、Anton Leykin 和 John A Christian。《基于方位测量的几何初始轨道确定》。载于*AAS/AIAA*轨道动力学专家会议，北卡罗来纳州夏洛特，2022年。

Shengyu Feng、Weiwei Sun、Shanda Li、Ameet Talwalkar 和 Yiming Yang。《基于机器学习的组合优化当代求解器综合评估》，2025年。网址 https://arxiv.org/abs/2505.16952。

Andreas Frommer 和 Bruno Lang。《利用Borsuk定理检验非线性方程解的存在性》。*SIAM* 数值分析杂志，43(3)：1348–1361，2005年。

Andreas Frommer、Bruno Lang 和 Marco Schnurr。《Moore存在性检验与Miranda存在性检验的比较》。*Computing*期刊，72(3)：349–354，2004年。

DD Gawali、Ahmed Zidna 和 PSV Nataraj。《采用多项式B样条展开和B样条Krawczyk算子分析非线性电路的综合方法》。国际应用与计算数学杂志，8(1)：1，2022年。

Alexandre Goldsztejn。Hansen-Sengupta 方法与 Frommer-Lang-Schnurr 稳定性测试的比较。*Computing*期刊，79:53–60，2007年。

Alexandre Goldsztejn 和 Fr´ed´eric Goualard。通过自适应削减实现盒一致性。载于*2010*年 *ACM* 应用计算研讨会论文集，第2049–2054页，2010年。

Fr´ed´eric Goualard 和 Christophe Jermann。基于强化学习的区间约束传播方法。*Constraints*，13(1):206–226，2008年。

Laurent Granvilliers。区间约束求解器的组合。***Reliable Computing***，7(6):467–483，2001年。

Laurent Granvilliers。数值约束满足问题的自适应二分。载于约束编程原理与实践国际会议，第290–298页。施普林格，2012年。

Laurent Granvilliers 和 Fr´ed´eric Benhamou。电路设计问题求解的进展。*Journal of Global Optimization*，20:155–168，2001年。

Laurent Granvilliers 和 Fr´ed´eric Benhamou。算法852：Realpaver：一种采用约束满足技术的区间求解器。*ACM Transactions on MathematicalSoftware (TOMS)*，32(1):138–156，2006年。

Laurent Granvilliers、Eric Monfroy 和 Fr´ed´eric Benhamou。约束编程中的符号—区间协作。载于*2001*年符号与代数计算国际研讨会论文集，第 150–166 页，2001年。

Eldon Hansen 和 G William Walster。利用区间分析进行全局优化：修订扩展版，第 264 卷。CRC出版社，2003年。

Eldon R Hansen 和 Ronald I Greenberg。区间牛顿法。应用数学与计算，12(2-3):89–98，1983年。

Guoliang Hou 和 Shugong Zhang。基于 Krawczyk算子的非线性方程组改进验证算法。计算与应用数学杂志，359:145–152，2019年。ISSN 0377-0427。doi: https://doi.org/10.1016/ j.cam.2019.03.034。URL https://www.sciencedirect.com/science/article/pii/S0377042719301591。

Calvin Huang、Soonho Kong、Sicun Gao 和 Damien Zufferey。在可满足性区间约束传播中评估分支启发式方法。载于数值软件验证：第 *12* 届国际研讨会，*NSV 2019*，纽约市，纽约州，美国，*2019* 年 *7* 月 *13–14* 日，论文集 *12*，第 85–100 页。施普林格，2019年。

Ruiqi Huang、Anton Leykin 和 Michela Mancini。用于初始轨道确定的射影平面细分法。载于科学计算中的计算机代数国际研讨会，第98–117页。施普林格，2025年。

Luc Jaulin、Michel Kieffer、Olivier Didrit 和 Eric Walter。《应用区间分析：参数与状态估计、鲁棒控制和机器人学示例》，2001年。RB Kearfott、MT Nakao、S Neumaier、SM Rump、SP Shary 和 P van Hentenryck。区间分析中的标准化记法。*Reliable Computing*，15(1):7–13，2010年。Lubomir V Kolev。线性区间包络的自动计算。*Reliable Computing*，7(1):17–28，2001年。Lubomir V Kolev。用于求解非线性问题的改进区间线性化。数值算法，37:213–224，2004年。´Angel Fernando Kuri-Morales。使用遗传算法求解非线性方程组。2002年。URL https://api.semanticscholar.org/CorpusID:1721781。Yahia Lebbah、Claude Michel 和 Michel Rueher。用于二次约束的严格全局过滤算法。*Constraints*，10:47–65，2005a。Yahia Lebbah、Claude Michel、Michel Rueher、David Daney 和 Jean-Pierre Merlet。用于处理数值约束系统的高效安全全局约束。*SIAM 数值分析杂志*，42(5):2076–2097，2005b。L Lukˇsan 和 Jan Vlcek。无约束优化与等式约束优化的稀疏及部分可分测试问题，1999年。Angelos Mantzaflaris、Bernard Mourrain 和 Elias Tsigaridas。多项式系统实根的连分数展开、复杂度与条件数。理论计算机科学，412(22):2312–2330，2011年。JP Merlet。Alias-c++。在线获取：*https://wwwsop. inria. fr/coprin/logiciels/ALIAS/ALIAS-C++/ALIAS-C++. html*，2007 年。Fr´ed´eric Messine。仿射算术的扩展：在无约束全局优化中的应用。*J. Univers. Comput. Sci.*，8(11):992–1015，2002年。Ramon E Moore 和 John B Kioustelidis。用于检验非线性（或线性）系统近似解精度的简单测试。*SIAM 数值分析杂志*，17 (4):521–529，1980年。Ramon E. Moore、Ralph Baker Kearfott 和 Michael J. Cloud。《区间分析导论》，2009年。URL https://api.semanticscholar.org/CorpusID:20069281。Arnold Neumaier。方程组的区间方法。剑桥大学出版社，1990年。

Arnold Neumaier。连续全局优化和约束满足中的完整搜索。*Acta numerica*，13:271–369，2004年。

Bertrand Neveu、Gilles Trombettoni 和 Ignacio Araya。自适应构造性区间析取：算法与实验。*Constraints*，20:452–467，2015年。

Bertrand Neveu、Gilles Trombettoni 和 Ignacio Araya。区间分支定界算法中的节点选择策略。*Journal of Global optimization*，64:289–304，2016年。

Jordan Ninin、Fr´ed´eric Messine 和 Pierre Hansen。用于全局优化的可靠仿射松弛方法。*4OR*，13(3):247–277，2015年。

Victor Reyes 和 Ignacio Araya。用于数值约束满足问题的基于探测的变量选择启发式方法。载于*2014*年 *IEEE* 第*26*届人工智能工具国际会议，第16–23页。IEEE，2014年。

Lara Scavuzzo、Karen Aardal、Andrea Lodi 和 Neil Yorke-Smith。用于混合整数线性规划的机器学习增强分支定界。*MathematicalProgramming*，2024年8月。ISSN 1436-4646。doi：10.1007/s10107-024-02130-y。URL https://doi.org/10.1007/s10107-024-02130-y。

Hermann Schichl 和 Arnold Neumaier。用于全局优化的有向无环图区间分析。*Journal of Global Optimization*，33(4)：541–562，2005年。

John Greenlees Semple 和 Geoffrey Thomas Kneebone。代数射影几何。牛津大学出版社，1998年。

Oleg Shcherbina、Arnold Neumaier、Djamila Sam-Haroud、Xuan-Ha Vu 和 Tuan-Viet Nguyen。全局优化和约束满足程序的基准测试。在全球优化与约束满足国际研讨会上，第211–222页。施普林格，2002年。

Evan C Sherbrooke 和 Nicholas M Patrikalakis。非线性多项式系统的解的计算。*Computer Aided Geometric Design*，10(5):379–405，1993年。

Andrew J Sommese、Charles W Wampler 等。工程与科学中产生的多项式系统的数值求解。世界科技出版公司，2005年。

Volker Stahl。用于限定多项式值域和求解非线性方程组的区间方法。博士论文，林茨约翰·开普勒大学，1995年。

MD Stuber、V Kumar 和 PI Barton。《非光滑排除检验：求解非线性方程的全部解》。《BIT Numerical Mathematics》，50(4):885–917，2010年。

Gilles Trombettoni 和 Gilles Chabert。构造性区间析取。载于《约束编程原理与实践–*CP 2007*：第*13*届国际会议，*CP 2007*，美国罗得岛州普罗维登斯，*2007*年*9*月*23*日至*27*日。会议论文集》第*13*卷，第635–650页。施普林格，2007年。

Gilles Trombettoni、Ignacio Araya、Bertrand Neveu 和 Gilles Chabert。用于全局优化的内部区域与区间线性化。载于人工智能 *AAAI* 会议论文集，第 25 卷，第 99–104 页，2011 年。

Pascal Van Hentenryck、David McAllester 和 Deepak Kapur。使用分支剪枝方法求解多项式系统。*SIAM* 数值分析杂志，34 (2):797–827，1997年。

JanVerschelde。使用 PHCPack 进行多项式同伦延拓。*ACM*计算机代数通讯，44(3/4):217–220，2011 年。

JanVerschelde、Pierre Verlinden 和 RonaldCools。利用牛顿多胞体的同伦来求解稀疏多项式系统。*SIAM* 数值分析杂志，31 (3):915–930，1994 年。doi: 10.1137/0731049。URL `https://doi.org/10.1137/0731049`。

Joachim Von Zur Gathen 和 J¨urgen Gerhard。《现代计算机代数》。剑桥大学出版社，2003年。

Xuan-HaVu. 数值约束满足问题的严格求解技术。博士论文，洛桑联邦理工学院，2005年。

Xuan-Ha Vu、Djamila Sam-Haroud 和 Boi Faltings。利用多重包含表示增强数值约束传播。数学与人工智能年刊，55：295–354，2009a。

Xuan-Ha Vu、Hermann Schichl 和 Djamila Sam-Haroud。用于数值约束求解的有向无环图上的区间传播与搜索。*Journal of GlobalOptimization*，45(4)：499–531，2009b。

Dongming Wang。求解多项式方程：特征集与三角系统。数学与计算机模拟，42(4)：339–351，1996年。ISSN 0378-4754。doi: https://doi.org/10.1016/S0378-4754(96)00008-0。URL `https://www.sciencedirect.com/science/article/pii/S0378475496000080`。符号计算：新趋势与发展。

Bican Xia 和 Ting Zhang。使用区间算术进行实解隔离。计算机与应用数学，52(6-7):853–860，2006年。

Juan Xu 和 Chee Yap。具有复杂度分析的有效细分算法，用于隔离实方程组的零点。载于*2019*年国际符号与代数计算研讨会论文集，ISSAC '19，第355–362页，美国纽约州纽约，2019年。计算机协会。