ÉCOLE POLYTECHNIQUE

École Doctorale de l'École Polytechnique

Laboratoire d'Informatique de l'École Polytechnique

Centre de Mathématiques Appliquées de l'École Polytechnique

Inria Saclay – Île-de-France

THÈSE DE DOCTORAT

par **Victor Magron**

soutenue le **9 décembre 2013**

Pour obtenir le grade de **Docteur de l'École Polytechnique**
Discipline : **Informatique**

Formal Proofs For Global Optimization
Templates and Sums of Squares

DIRECTEURS DE THÈSE

M. GAUBERT Stéphane	*Directeur de Recherche, Inria Saclay*
M. WERNER Benjamin	*Professeur, École Polytechnique*

RAPPORTEURS

M. BERTOT Yves	*Directeur de Recherche, Inria Sophia Antipolis*
M. HALES Thomas	*Professor, University of Pittsburgh*
M. HENRION Didier	*Directeur de Recherche, LAAS CNRS*

EXAMINATEURS

M. ALLAMIGEON Xavier	*Chargé de Recherche, Inria Saclay*
Mme LAURENT Monique	*Professor, CWI and University of Tilburg*
M. SCHWEIGHOFER Markus	*Professor, Doctor, University of Konstanz*

Contents

List of Figures

List of Tables

Remerciements

Je tiens à commencer cette thèse[1] en remerciant chaleureusement mes trois chefs: Benjamin, Stéphane et Xavier. Merci pour votre encadrement, votre gentillesse, votre patience et votre réactivité sur tous les aspects de mon travail de thèse. Votre motivation ainsi que votre goût pour l'enseignement ont contribué à rendre cette recherche agréable.

Je remercie aussi Didier Henrion pour avoir accepté de rapporter ma thèse. Tes commentaires ont grandement contribué à améliorer ce manuscrit. Je te remercie aussi pour ton invitation au LAAS, ainsi que pour tous les évènements que tu as organisés et auxquels j'ai pu participer au cours de ces trois années.

Un grand merci à Yves Bertot d'avoir accepté de rapporter pour ma thèse et pour cette fabuleuse école de printemps qui m'a fait découvrir SSREFLECT.

Thank you very much Thomas Hales for reviewing my dissertation and for your detailed comments and corrections that helped me improving it.

Je tiens aussi à remercier vivement Markus et Monique d'avoir accepté de faire partie de mon jury. Merci pour tous vos efforts à organiser des évènements remarquables (conférences, séminaires, écoles d'été) au cours desquels j'ai énormément appris.

Je souhaite remercier Jean-Bernard Lasserre du LAAS. Tes précieux conseils m'ont souvent éclairé sur des aspects aussi bien théoriques que pratiques.

Je tiens aussi à remercier tous ceux qui ont éveillé mon intérêt pour la preuve formelle avant que je commence cette thèse : Gilles Dowek, Guillaume Melquiond, Sylvie Boldo, Eric Goubault, Germain Faure et Stéphane Lengrand.

Thank you Hayato Waki for your invitation to the ISMP conference in Berlin and for the nice discussions about sparse SOS relaxations.

Je remercie également Assia et Enrico de Mathematical Components. Merci d'avoir pris le temps de me guider au quotidien dans ma pratique de COQ.

Merci à David Monniaux et Mohab Safey El Din pour les intéressantes discussions que nous avons eues lorsque j'ai débuté ma thèse.

Merci à toutes celles et à tous ceux qui ont relu mon manuscrit: Chantal et Jean-Philippe.

Merci à tous les chercheurs avec qui j'ai eu l'occasion d'enseigner et d'échanger : Steve, Luca, Stéphane, Jonathan, Yvan et Amaury. Un remerciement spécial à Jean-Christophe pour sa faculté à débugger mes programmes OCAML de tête pendant les TD de Java.

[1] The research leading to these results has received funding from the European Union's 7th Framework Programme under grant agreement nr. 243847 (ForMath).

Merci à Nicolas Brisebarre pour les séminaires auxquels j'ai eu la chance de participer à Lyon. Merci à Cyril, Maxime, Arnaud et Erik pour leurs conseils concernant Coq.

Thank you Alexey for your interesting discussions about the Flyspeck project.

Merci à tous les collègues grâce à qui les discussions de cafés et repas ont été aussi animées, en particulier à tous ceux qui m'ont dévoilé les secrets de l'ENS Lyon: Julien, Jonas, Amaury, Chantal, Olivier, Bruno et tous ceux que j'oublie. Merci à tous les François et Mi(c)kaël du labo. Merci aux doctorants et postdocs de l'équipe Crypto pour avoir rendu mon été de rédaction plus supportable. Merci à Pascal et à tous les doctorants du CMAP que j'ai côtoyés. Merci à tous les camarades d'XDoc : Blaise, Barbara, Marc.

Merci à mes chers amis toulousains Julien, Xavier et Romain pour avoir toujours été fidèles à eux-mêmes.

Merci à mes amis Centraliens avec qui j'ai pu partager tant de choses au cours de ces dernières années : Chris, Eugénie, Etienne, Florian, Loulou, Nicolas, Pierre, Thibal, Yvonnick et à tous ceux que j'oublie honteusement.

Merci aux PCéens et affiliés : Manu et Hélène, Amir, les Thomas, Romain, Luc, Céline.

Merci à tous mes Japonais et assimilés : Naoya, Satoko, Johji, les David, Arthur, Rita, Aurélien, Alex, Hugo.

Merci à Alexandre et à tous ses amis pour me rappeler régulièrement que j'ai raté ma carrière de cycliste.

Merci à Margaux pour les Ramen, le vélo, ses stratégies et ses principes.

Merci à Djo, pour tous ses principes. Merci à Clém de l'avoir choisi.

Merci à Noémie pour m'avoir fait découvrir les joies de l'homme barbu. Merci à Julius qui a su y mettre un terme.

Merci à Jean-Noël et à tous ceux du Taekwondo qui m'ont aidé à progresser lors des entrainements au cours de ma thèse, notamment Rémi, Tiphaine, Xavier, Jérôme, Willy, Julia et Luc. Merci à tous les maitres avec qui j'ai eu la chance de pratiquer lors de ces trois années. Merci aux sœurs Pochard pour la voltige.

Merci à Ben, Nico, Florent et Elsa pour les débats inoubliables sur la nature de l'homme, Dieu et l'écolo-fascisme.

Merci à José pour ses encouragements de dernière minute.

Merci à Charles, Paul, Maxime et Philippe pour toutes leurs inventions et toutes leurs personnalités.

Merci à ma famille et à mes parents pour leur soutien et leur force extraordinaire. A mes quatre frères pour leur intelligence, leur humour et leur bonté.

Résumé

Cette thèse a pour but de certifier des bornes inférieures de fonctions multivariées à valeurs réelles, définies par des expressions semi-algébriques ou transcendantes et de prouver leur validité en vérifiant les certificats dans l'assistant de preuves COQ.

De nombreuses inégalités de cette nature apparaissent par exemple dans la preuve par Thomas Hales de la conjecture de Kepler.

Dans le cadre de cette étude, on s'intéresse à des fonctions non-linéaires, faisant intervenir des opérations semi-algébriques ainsi que des fonctions transcendantes univariées (cos, arctan, exp, etc).

L'utilisation de différentes méthodes d'approximation permet de relâcher le problème initial en un problème d'optimisation semi-algébrique. On se ramène ainsi à des problèmes d'optimisation polynomiale, qu'on résout par des techniques de sommes de carrés creuses.

Dans un premier temps, nous présentons une technique classique d'optimisation globale. Les fonctions transcendantes univariées sont approchées par les meilleurs estimateurs polynomiaux uniformes de degré d.

Par la suite, nous présentons une méthode alternative, qui consiste a borner certains des constituants de la fonction non-linéaire par des suprema de formes quadratiques (approximation maxplus, introduite à l'origine en contrôle optimal) de courbures judicieusement choisies.

Enfin, cet algorithme d'approximation est amélioré, en combinant l'idée des estimateurs maxplus et de la méthode des gabarits développée par Manna et al. (en analyse statique). Les gabarits non-linéaires permettent un compromis sur la precision des approximations maxplus afin de contrôler la complexité des estimateurs semi-algébriques. Ainsi, on obtient une nouvelle technique d'optimisation globale, basée sur les gabarits, qui exploite à la fois la precision des sommes de carrés et la capacité de passage à l'échelle des méthodes d'abstraction.

L'implémentation de ces méthodes d'approximation a abouti à un outil logiciel : `NLCertify`. Cet outil génère des certificats à partir d'approximations semi-algébriques et de sommes de carrés. Son interface avec COQ permet de bénéficier de l'arithmétique certifiée disponible dans l'assistant de preuves, et ainsi d'obtenir des estimateurs et des bornes valides pour chaque approximation.

Nous démontrons les performances de cet outil de certification sur divers problèmes d'optimisation globale ainsi que sur des inégalités serrées qui interviennent dans la preuve de Hales.

Abstract

The aim of this work is to certify lower bounds for real-valued multivariate functions, defined by semialgebraic or transcendental expressions and to prove their correctness by checking the certificates in the Coq proof system.

The application range for such a tool is widespread; for instance Hales' proof of Kepler's conjecture involves thousands of nonlinear inequalities.

The functions we are dealing with are nonlinear and involve semialgebraic operations as well as some transcendental functions like cos, arctan, exp, etc. Our general framework is to use different approximation methods to relax the original problem into a semialgebraic optimization problem. It leads to polynomial optimization problems which we solve by sparse sums of squares relaxations.

First, we implement a classical scheme in global optimization. Namely, we approximate univariate transcendental functions with best uniform degree-d polynomial estimators.

Then, we present an alternative method, which consists in bounding some of the constituents of the nonlinear function by suprema or infima of quadratic polynomials (max-plus approximation method, initially introduced in optimal control) with a carefully chosen curvature.

Finally, we improve this approximation algorithm, by combining the ideas of the maxplus estimators and of the linear template method developed by Manna et al. (in static analysis). The nonlinear templates control the complexity of the semialgebraic estimators at the price of coarsening the maxplus approximations. In that way, we arrive at a new - template based - global optimization method, which exploits both the precision of sums of squares/SDP relaxations and the scalability of abstraction methods.

We successfully implemented these approximation methods in a software package named `NLCertify`. This tool interleaves semialgebraic approximations with sums of squares witnesses to form certificates. It is interfaced with Coq and thus benefits from the trusted arithmetic available inside the proof assistant. This feature is used to produce, from the certificates, both valid underestimators and lower bounds for each approximated constituent.

We illustrate the efficiency of `NLCertify` with various examples from the global optimization literature, as well as tight inequalities issued from the Flyspeck project.

Chapter 1

Introduction: from Computer Assisted Proofs to Formal Global Optimization

1.1 Problems Involving Computer Assisted Proofs

Numerous problems coming from different fields of mathematics (like combinatorics, geometry or group theory) have led to computer assisted proofs. Two classical examples are the proof of the Four Colours Theorem by Appel and Haken [AH77] and the Kepler conjecture.

The latter can be stated as follows:

Conjecture 1.1 (Kepler (1611)). *The maximal density of sphere packings in three dimensional space is $\pi / \sqrt{18}$.*

This conjecture was proved by Thomas Hales.

Theorem 1.2 (Hales [Hal94, Hal05]). *Kepler's conjecture is true.*

One of the chapters of [Hal05] is coauthored by Ferguson. The publication of the proof, one of the "most complicated [...] that has been ever produced", to quote his author[1], took several years and its verification required "unprecedented" efforts by a team of referees. The verification of proofs with this degree of complexity has motivated the use of formal proof techniques.

The computer-checked proofs of both problems contain computational and mathematical parts. The formal proof of the Four Colours Theorem has been done by Gonthier [Gon08] with the COQ [Coq] proof assistant. The formal proof of Conjecture 1.1 is an ambitious goal addressed by the Flyspeck project, launched by Hales himself [Hal06].

Some other problems can be solved by proof assistants (or "interactive theorem provers") but do not rely on mechanical computation. As an example, one can mention the formal proof of the Feit-Thompson Odd Order Theorem [GAA+].

[1] `https://code.google.com/p/flyspeck/wiki/FlyspeckFactSheet`

1.1.1 Nonlinear Inequalities arising in the Flyspeck Project

Recent efforts have been made to complete the formal verification of Kepler's conjecture. Flyspeck [Hal06] is a large-scale effort which needs to tackle various mathematical tools in a formal setting.

In Flyspeck, extensive computation are mandatory to handle the formal generation of some special planar graphs. The formal proofs of the bounds of linear and nonlinear programs also require major computational time. Details about the two former issues are available in Solovyev's doctoral dissertation [Sol]. Here we focus on the latter issue, namely checking the correctness of hundreds of nonlinear inequalities with an interactive theorem prover. We will often refer to the following inequality taken from Hales' proof:

Example 1.3 (Lemma$_{9922699028}$ Flyspeck). Let K, $\Delta\mathbf{x}$, l, t and f be defined as follows:

$$\begin{aligned}
K &:= [4, 6.3504]^3 \times [6.3504, 8] \times [4, 6.3504]^2 \ , \\
\Delta\mathbf{x} &:= x_1x_4(-x_1 + x_2 + x_3 - x_4 + x_5 + x_6) \\
&\quad + x_2x_5(x_1 - x_2 + x_3 + x_4 - x_5 + x_6) \\
&\quad + x_3x_6(x_1 + x_2 - x_3 + x_4 + x_5 - x_6) \\
&\quad - x_2x_3x_4 - x_1x_3x_5 - x_1x_2x_6 - x_4x_5x_6 \ , \\
l(\mathbf{x}) &:= -\pi/2 + 1.6294 - 0.2213(\sqrt{x_2} + \sqrt{x_3} + \sqrt{x_5} + \sqrt{x_6} - 8.0) \\
&\quad + 0.913(\sqrt{x_4} - 2.52) + 0.728(\sqrt{x_1} - 2.0) \ , \\
t(\mathbf{x}) &:= \arctan \frac{\partial_4 \Delta\mathbf{x}}{\sqrt{4x_1\Delta\mathbf{x}}} \ , \\
f(\mathbf{x}) &:= l(\mathbf{x}) + t(\mathbf{x}) \ .
\end{aligned}$$

Then, $\forall \mathbf{x} \in K, f(\mathbf{x}) \geqslant 0$.

Other examples of inequalities can be found in Appendix A. Note that the inequality of Example 1.3 would be much simpler to check if l was a constant (rather than a function of $\mathbf{x}$). Indeed, semialgebraic optimization methods would provide precise lower and upper bounds for the argument of arctan. Then we could conclude by monotonicity of arctan using interval arithmetic. Here, both l and t depend on $\mathbf{x}$. Hence, by using interval arithmetic addition (without any domain subdivision) on the sum $l + t$, which ignores the correlation between the argument of arctan and the function l, we only obtain a coarse lower bound (equal to -0.87, see Example 2.12 for details); too coarse to assert the inequality . A standard way to improve this bound consists in subdividing the initial box K and performing interval arithmetic on smaller boxes. However, this approach suffers from the so called *curse of dimensionality*. Therefore, it is desirable to develop alternative certified global optimization methods, applicable to a wide class of problems involving semialgebraic and transcendental functions. This is the goal of this dissertation.

Moreover, the nonlinear inequalities of Flyspeck are challenging for numerical solvers for two reasons. First, they involve a medium-scale number of variables (6$\sim$10). Then, they are essentially *tight*. For instance, the function f involved in Example 1.3 has a nonnegative infimum which is less than 10^{-3}. The tightness of the inequalities to be certified is actually a frequent feature in mathematical proofs. Hence, we will pay a special attention in the present work to *scalability* and *numerical precision* issues.

1.1.2 Formal Global Optimization Problems

We now describe the global optimization problems that we shall consider. Let $\langle \mathcal{D} \rangle^{\text{sa}}$ be the set of functions obtained by composing (multivariate) semialgebraic functions with special functions taken from a *dictionary* $\mathcal{D}$. We will typically include in $\mathcal{D}$ the usual functions tan, arctan, cos, arccos, sin, arcsin, exp, log, $(\cdot)^r$ with $r \in \mathbb{R} \setminus \{0\}$. As we allow the composition with semi-algebraic functions in our setting, elementary functions like $+, -, \times, /, |\cdot|, \sup(\cdot,\cdot), \inf(\cdot,\cdot)$ are of course covered. Actually, we shall see that some of the present results remain valid if the dictionary includes semiconcave or semiconvex functions with effective lower and upper bounds on the Hessian, see Chapter 5.

Given $f, f_1, \ldots, f_p \in \langle \mathcal{D} \rangle^{\text{sa}}$, we will address the following global optimization problem:

$$\begin{aligned} \inf_{\mathbf{x} \in \mathbb{R}^n} \quad & f(\mathbf{x}) \ , \\ \text{s.t.} \quad & f_1(\mathbf{x}) \geqslant 0, \ldots, f_p(\mathbf{x}) \geqslant 0 \ . \end{aligned} \tag{1.1.1}$$

The inequalities issued from Flyspeck actually deal with special cases of computation of a certified lower bound for a real-valued multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ over a compact semialgebraic set $K \subset \mathbb{R}^n$. Checking these inequalities boils down to automatically provide lower bounds for the following instance of Problem (1.1.1):

$$f^* := \inf_{\mathbf{x} \in K} f(\mathbf{x}) \ , \tag{1.1.2}$$

We shall also search for *certificates* to assess that:

$$\forall \mathbf{x} \in K, f(\mathbf{x}) \geqslant 0 \ . \tag{1.1.3}$$

A well studied case is when $\mathcal{D}$ is reduced to the identity map $\{Id\}$. Then, $f = f_{\text{sa}}$ belongs to the algebra $\mathcal{A}$ of semialgebraic functions and Problem (1.1.1) specializes to the semialgebraic optimization problem:

$$f_{\text{sa}}^* := \inf_{\mathbf{x} \in K} f_{\text{sa}}(\mathbf{x}) \ . \tag{1.1.4}$$

Another important sub-case is Polynomial Optimization Problems (POP), when $f = f_{\text{pop}}$ is a multivariate polynomial and $K = K_{\text{pop}}$ is given by finitely many polynomial inequalities. Thus, Problem (1.1.4) becomes:

$$f_{\text{pop}}^* := \inf_{\mathbf{x} \in K_{\text{pop}}} f_{\text{pop}}(\mathbf{x}) \ . \tag{1.1.5}$$

We shall see that the presented methods also provide certified lower bounds (possibly coarse), for optimization problems which are hard to solve by traditional POP techniques. Such problems have a relatively large number of variables (10∼100) or are polynomial inequalities of a moderate degree. For illustration purposes, we consider the following running example coming from the global optimization literature

Example 1.4 (Modified Schwefel Problem 43 from Appendix B in [AKZ05]).

$$\min_{\mathbf{x} \in [1,500]^n} f(\mathbf{x}) = -\sum_{i=1}^{n-1} (x_i + \epsilon x_{i+1}) \sin(\sqrt{x_i}) \ ,$$

where ϵ is a fixed parameter in $\{0, 1\}$. In the original problem, $\epsilon = 0$, *i.e.* the objective function f is the sum of independent functions involving a single variable. This property may be exploited by a global optimization solver by reducing it to the problem $\min_{x \in [1,500]} x \sin(\sqrt{x})$. Hence, we also consider a modified version of this problem with $\epsilon = 1$.

The following test examples are taken from Appendix B in [AKZ05]. Some of these examples involve functions that depend on numerical constants, the values of which can be found there. Numerical experiments are performed with the different algorithms described in this dissertation to return certified lower bounds of these functions (see Sections 4.3, 5.4 and 6.5).

- *Hartman 3 (H3)*: $\min\limits_{\mathbf{x} \in [0,1]^3} f(\mathbf{x}) = -\sum\limits_{i=1}^{4} c_i \exp\left[-\sum\limits_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$.

- *Hartman 6 (H6)*: $\min\limits_{\mathbf{x} \in [0,1]^6} f(\mathbf{x}) = -\sum\limits_{i=1}^{4} c_i \exp\left[-\sum\limits_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$.

- *Mc Cormick (MC)*, with $K = [-1.5, 4] \times [-3, 3]$:
  $\min\limits_{\mathbf{x} \in K} f(\mathbf{x}) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$.

- *Modified Langerman (ML)*:
  $\min\limits_{\mathbf{x} \in [0,10]^n} f(\mathbf{x}) = \sum\limits_{j=1}^{5} c_j \cos(d_j/\pi) \exp(-\pi d_j)$, with $d_j = \sum\limits_{i=1}^{n} (x_i - a_{ji})^2$.

- *Paviani Problem (PP)*, with $K = [2.01, 9.99]^{10}$:
  $\min\limits_{\mathbf{x} \in K} f(\mathbf{x}) = \sum\limits_{i=1}^{10} \left[(\log(x_i - 2))^2 - \log(10 - x_i))^2\right] - \left(\prod\limits_{i=1}^{10} x_i\right)^{0.2}$.

- *Shubert (SBT)*: $\min\limits_{\mathbf{x} \in [-10,10]^n} f(\mathbf{x}) = \prod\limits_{i=1}^{n} \Big(\sum\limits_{j=1}^{5} j \cos((j+1)x_i + j)\Big)$.

- *Schwefel Problem (SWF)*: $\min\limits_{\mathbf{x} \in [1,500]^n} f(\mathbf{x}) = -\sum_{i=1}^{n} x_i \sin(\sqrt{x_i})$.

1.1.3 Non Commutative Optimization

Further motivations of the present work arise from global optimization of real polynomials of non-commutative variables.

One recent open problem is the generalized Lax conjecture [Lax58], which states that it is always possible to realize hyperbolicity cones as spectrahedra. A new approach [NT12] to this problem involves non commutative sums of squares (in the Clifford algebras) and has formal computational applications.

The problem of proving non commutative polynomial inequalities has also occurred in a conjecture formulated by Bessis, Moussa and Villani in 1975. This conjecture can be restated as follows:

Conjecture 1.5 (Lieb and Seiringer [LS04]). *For all positive semidefinite matrices A and B and all $m \in \mathbb{N}$, the single variable polynomial $p(t) := \mathbf{Tr}((A + tB)^m) \in \mathbb{R}[t]$ has only nonnegative coefficients.*

Using semidefinite programming and sums of hermitian squares, Schweighofer and Klep [KS08] established a proof of the conjecture for $m \leqslant 13$. Non commutative sums of squares certificates are also amenable to the present techniques. However, we mention that the BMV conjecture has been recently established by Herbert R Stahl [Sta11], following a different line of analysis.

1.2 Certified Global Optimization in the Literature

A common idea to handle Problem (1.1.2) is to first estimate f by multivariate polynomials and then obtain a lower bound of the resulting approximation by polynomial optimization techniques.

Computing lower bounds in constrained POP (see Problem(1.1.5)) is already a difficult problem, which has received much attention. Sums of squares (SOS) relaxation based methods, leading to the resolution of semidefinite programs (SDP) have been developed in [Las01, PS03]. They can be applied to the more general class of semialgebraic problems [Put93]. Moreover, Kojima has developed a sparse refinement of the hierarchy of SDP relaxations (see [WKKM06]). This has been implemented in the SPARSEPOP solver. Checking the validity of the lower bound of POP implies being able to control and certify the numerical error, as SDP solvers are typically implemented using floating point arithmetic. Such techniques rely on hybrid symbolic-numeric certification methods, see Peyrl and Parrilo [PP08] and Kaltofen et al. [KLYZ12]. They allow one to produce positivity certificates for such POP which can be checked in proof assistants such as COQ [MC11, Bes07], HOL-LIGHT [Har07] or MetiTarski [AP10]. Alternative approaches to SOS/SDP are based on Bernstein polynomials [Zum08].

The task is obviously more difficult in presence of transcendental functions. Other methods of choice, not restricted to polynomial systems, include global optimization by interval methods (see e.g. [Han06]), branch and bound methods with Taylor models [CGT11, BM09]. Other methods involve rigorous Chebyshev estimators. An implementation of such approximations is available in the `Sollya` tool [CJL10]. We also mention procedures that solve SMT problems over the real numbers, using interval constraint propagation [GAC12].

Recent efforts have been made to perform a formal verification of several Flyspeck inequalities with Taylor interval approximations in the HOL-LIGHT proof assistant [SH13]. The Flocq library formalizes floating-point arithmetic inside COQ [Mel12]. The tactic `interval`, built on top of Flocq, can simplify inequalities on expressions of real numbers.

1.3 Contribution

1.3.1 A General Certification Scheme

In this thesis, we develop a general certification framework, combining methods from semialgebraic programming (SOS certificates, SDP relaxations) and from approximation theory. This includes classical methods like best uniform polynomials and less classical like maxplus approximation (inspired by optimal control and static analysis by abstract interpretation).

The present approach exploits both the accuracy of SOS relaxations and the scalability of the approximation and abstraction procedure. This leads to a new method in global optimization, the nonlinear template method, that produces certificates, which are ultimately proved in COQ.

The general principle of our framework is explained in Chapter 3. We alternate steps of semialgebraic approximation for some constituents of the objective function f and semialgebraic optimization. The resulting constrained polynomial optimization problems are solved with sums of squares relaxation from Lasserre hierarchy, by calling a semidefinite solver. In this way, each iteration of the algorithms refines the following inequalities:

$$f^* \geqslant f_{sa}^* \geqslant f_{pop}^* \ , \tag{1.3.1}$$

where f^* is the optimal value of the original problem, f_{sa}^* the optimal value of its current semialgebraic approximation and f_{pop}^* the optimal value of the SDP relaxation which we solve. Under certain moderate assumptions, the lower estimate f_{pop}^* does converge to f^* (see Corollary 3.9).

Different semialgebraic approximation schemes for transcendental functions are presented, namely minimax estimators (Chapter 4), maxplus approximations (Chapter 5) and templates abstractions (Chapter 6).

Minimax Polynomial Approximations

A natural workaround to deal with non-polynomial optimization problems is to estimate the objective function f with its best uniform (also called "minimax") degree-d polynomial approximation.

Thus, we obtain a hierarchy of minimax semialgebraic approximations, which leads to the algorithm `minimax_optim`. In practice, an interface with the software `Sollya` [CJL10] provides the univariate minimax polynomials.

Maxplus Estimators

The second method uses maxplus approximation of semiconvex transcendental functions by quadratic functions. The idea of maxplus approximation comes from optimal control: it was originally introduced by Fleming and McEneaney [FM00] and developed by several authors [AGL08a, MDG08, McE07, SGJM10, GMQ11], to represent the value function by a "maxplus linear combination", which is a supremum of certain basis functions, like quadratic polynomials. When applied to the present context, this idea leads to approximate from above and from below every transcendental function appearing in the description of the problem by infima and suprema of finitely many quadratic polynomials.

In that way, we are reduced to a converging sequence of semialgebraic problems. A geometrical way to interpret the method is to think of it in terms of "quadratic cuts" quadratic inequalities are successively added to approximate the graph of a transcendental function.

This work was published in the Proceedings of the 12th European Control Conference [AGMW13b].

Templates Abstractions

The nonlinear template method is an improved version of the maxplus approximation. By comparison, the new component is the introduction of the template technique (approximating projections of the feasible sets), leading to an increase in scalability.

This technique is an abstraction method, which is inspired by the linear template of Sankaranarayanan, Sipma and Manna in static analysis [SSM05], their nonlinear extensions by Adjé et al. [AGG12], and the maxplus basis method [SGJM10].

In the present application, templates are used both to approximate transcendental functions, and to produce coarser but still tractable relaxations when the standard SDP relaxation of the semialgebraic problem is too complex to be handled. As a matter of fact, SDP relaxations are a powerful tool to get tight certified lower bound for semialgebraic optimization problems, but applying them is currently limited to small or medium size problems: their execution time grows very rapidly with the relaxation order, which itself grows with the degree of the polynomials involved in the semialgebraic relaxations. The template method allows to reduce these degrees, by approximating certain projections of the feasible set by a moderate number of nonlinear inequalities.

They are also useful as a replacement of standard Taylor approximations of transcendental functions: instead of increasing the degree of the approximation, one increases the number of functions in the template.

In this dissertation, we present two families of templates:

- **Non-convex quadratic templates for POP** The objective polynomial of a non tractable POP is replaced by a suprema of quadratic polynomials (see Section 6.2).

- **Polynomial underestimators templates for semialgebraic functions** Given a degree d and a semialgebraic function f_{sa} that involves a large number of lifting variables, we build a hierarchy of polynomial approximations, that converge to the best (for the L_1-norm) degree-d polynomial underestimator of f_{sa} (see Section 6.3).

A part of this work was published in the Proceedings of the Conferences on Intelligent Computer Mathematics, in the Lecture Notes in Artificial Intelligence (LNAI 7961) [AGMW13a].

1.3.2 Software Implementation in OCAML and COQ

The final achievement of this work is the software implementation of our general optimization algorithm, leading to the package `NLCertify`.

Given a nonlinear inequality and an approximation method as input, our package `NLCertify` builds a hierarchy of approximations and generates the corresponding semidefinite relaxations, using OCAML libraries and external programs (`Sollya` and SDPA). The correctness of the bounds for semialgebraic optimization problems can be verified using the interface of `NLCertify` with COQ. Thus, the certificate search and the proof checking inside COQ are separated, a so called *sceptical* approach.

When solving a POP, the sums of squares certificate does not match with the system of polynomial inequalities, due to a rounding error. A certified upper bound of this error is obtained inside COQ (Section 7.3.2). This verification procedure is carefully implemented, using an equality test in the ring of polynomials whose coefficients are arbitrary-size rationals (Section 7.2.2). It ensures efficient computation inside the proof assistant.

Moreover, these verifications for POP relaxations are combined to deduce the correctness of semialgebraic optimization procedures, which requires in particular to assert that the semialgebraic functions are well-defined. It allows to handle more complex certificates for non-polynomial problems. Finally, the datatype structure of these certificates allows to reconstruct the steps of the optimization algorithm.

The software implementation package `NLCertify`, is described in Appendix B. At the time the author writes this PhD dissertation, the `NLCertify` tool contains more than 2600 lines of code for the COQ proof scripts and more than 15000 lines of code for the OCAML informal certification program.

1.4 Outline

This dissertation is organized as follows:

Part I introduces the required background about semialgebraic optimization and presents a general global optimization framework, which relies on an approximation algorithm `approx` and the optimization procedure `optim`.

- In Chapter 2, we recall the basic principles of semidefinite programming, the definition and properties of Lasserre relaxations of polynomial problems, together with reformulations by Lasserre and Putinar of semialgebraic problems classes. The sparse variant of these relaxations is also presented.

- In Chapter 3, we describe our general approximation scheme for global optimization. Then, we give a proof of convergence of this algorithm.

Part II instantiates the algorithm `approx` with minimax, maxplus and templates approximations.

- In Chapter 4, we recall some fundamental statements about best uniform polynomial approximations. Then we present the performance results of our algorithm `minimax_optim`, which combines minimax estimators and SOS.

- In Chapter 5, the maxplus approximation and the `samp_optim` algorithm are presented. We show how this algorithm can be combined with standard domain subdivision methods, to reduce the relaxation gap.

- In Chapter 6, we present the nonlinear extension of the template method. We explain how to control the complexity of semialgebraic optimization problems with our algorithm `template_optim`.

Part III focuses on formal proofs for global optimization via templates and sums of squares.

- In Chapter 7, we recall the required background on the arithmetic available inside COQ. Then, we explain how to check in COQ nonlinear inequalities involving multivariate transcendental functions.

- In Chapter 8, we conclude and discuss the perspectives of this work.

Part I

A General Framework for Certified Global Optimization

Chapter 2

Sums of Squares based Optimization

In this chapter, we describe the foundations on which several parts of our work lie. Semidefinite programming (SDP) is introduced in (Section 2.1). We use a hierarchy of SDP relaxations by Lasserre to solve polynomial optimization problems (Section 2.2) as well as semialgebraic optimization problems (Section 2.3). Then, we recall the sparse variant of these relaxations by Kojima (Section 2.4). Finally, we explain how to extract an SOS certificate from the solution of an SDP (Section 2.5).

2.1 Semidefinite Programming and Interior-Points methods

Semidefinite programming is relevant to a wide range of applications. The interested reader can find more details on the connection between SDP and combinatorial optimization in [GLV09], control theory in [BEFB94], positive semidefinite matrix completion in [Lau09]. A survey on semidefinite programming is available in the paper of Vandenberghe and Boyd [VB94]. Here, we give the basic definitions of primal and dual semidefinite programs, then explain how to solve these programs with the interior-points methods and recall some complexity results regarding the cost of these techniques. Even though semidefinite programming is not our main topic of interest, several encountered problems can be cast as semidefinite programs. We emphasize the fact that semidefinite programs can be solved efficiently (up to a few thousand optimization variables) by freely available software (e.g. SeDuMi [Stu98], CSDP [Bor97], SDPA [YFN$^+$10]).

First, we introduce some useful notations. We consider the vector space $\mathcal{S}_n$ of real symmetric $n \times n$ matrices. Given $X \in \mathcal{S}_n$, let $\lambda_{\max}(X)$ (resp. $\lambda_{\min}(X)$) be the maximum (resp. minimum) eigenvalue of X. It is equipped with the usual inner product $\langle X, Y \rangle = \mathbf{Tr}(XY)$ for $X, Y \in \mathcal{S}_n$. Let I_n be the $n \times n$ identity matrix. The Frobenius norm of a matrix $X \in \mathcal{S}_n$ is defined by $\|X\|_F := \sqrt{\mathbf{Tr}(X^2)}$. A matrix $M \in \mathcal{S}_n$ is called positive semidefinite if $\mathbf{x}^T M \mathbf{x} \geqslant 0$, $\forall \mathbf{x} \in \mathbb{R}^n$. In this case, we write $M \succcurlyeq 0$ and define a partial order by writing $X \succcurlyeq Y$ (resp. $X \succ Y$) if and only if $X - Y$ is positive semidefinite (resp. positive definite).

In semidefinite programming, one minimizes a linear objective function subject to a linear matrix inequality (LMI). The variable of the problem is the vector $\mathbf{x} \in \mathbb{R}^m$ and the problem input data are the vector $c \in \mathbb{R}^m$ and symmetric matrices $F_0, \ldots, F_m \in \mathcal{S}_n$. The primal semidefinite program is defined as follows:

$$\begin{aligned} p^*_{\mathrm{sdp}} := \min_{\mathbf{x} \in \mathbb{R}^m} \quad & c^T \mathbf{x} \\ s.t. \quad & F(\mathbf{x}) \succcurlyeq 0 \ , \end{aligned} \tag{2.1.1}$$

where

$$F(\mathbf{x}) := F_0 + \sum_{i=1}^m x_i F_i \ .$$

The primal problem 2.1.1 is convex since the linear objective function and the linear matrix inequality constraint are both convex. We say that $\mathbf{x}$ is primal feasible (resp. strictly feasible) if $F(\mathbf{x}) \succcurlyeq 0$ (resp. $F(\mathbf{x}) \succ 0$). Furthermore, we associate the following dual problem with the primal problem 2.1.1:

$$\begin{aligned} d^*_{\mathrm{sdp}} := \max_{Z \in \mathcal{S}_n} \quad & -\mathbf{Tr}(F_0 Z) \\ s.t. \quad & \mathbf{Tr}(F_i Z) = c_i, \quad i = 1, \ldots, m \ , \\ & Z \succcurlyeq 0 \ . \end{aligned} \tag{2.1.2}$$

The variable of the dual program 2.1.2 is the real symmetric matrix $Z \in \mathcal{S}_n$. We say that Z is dual feasible (resp. strictly feasible) if $\mathbf{Tr}(F_i Z) = c_i, \quad i = 1, \ldots, m$ and $Z \succcurlyeq 0$ (resp. $Z \succ 0$).

We will describe briefly the primal-dual interior-point method (used for instance by the SDPA software), that solves the following primal-dual optimization problem:

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^m, Z \in \mathcal{S}_n} \quad & \eta(\mathbf{x}, Z) \\ \text{s.t.} \quad & \mathbf{Tr}(F_i Z) = c_i, \quad i = 1, \ldots, m \ , \\ & F(\mathbf{x}) \succcurlyeq 0, \quad Z \succcurlyeq 0 \ , \end{aligned} \tag{2.1.3}$$

where,

$$\eta(\mathbf{x}, Z) := c^T \mathbf{x} + \mathbf{Tr}(F_0 Z) \ .$$

We notice that the objective function η of the program 2.1.3 is the difference between the objective function of the primal program 2.1.1 and its dual version 2.1.2. We call this function the duality gap. Let suppose that $\mathbf{x}$ is primal feasible and Z is dual feasible, then η is nonnegative. Indeed, we have:

$$\eta(\mathbf{x}, Z) = \sum_{i=1}^m \mathbf{Tr}(F_i Z) x_i + \mathbf{Tr}(F_0 Z) = \mathbf{Tr}(F(\mathbf{x}) Z) \geqslant 0 \ . \tag{2.1.4}$$

The last inequality comes from the fact that the matrices $F(\mathbf{x})$ and Z are both positive semidefinite.

Then, one can easily prove that the nonnegativity of η implies the following inequalities:

$$d^*_{\mathrm{sdp}} \leqslant -\mathbf{Tr}(F_0 Z) \leqslant c^T \mathbf{x} \leqslant p^*_{\mathrm{sdp}} \ . \tag{2.1.5}$$

Our problems that can be cast as semidefinite programs (SDP) satisfy certain assumptions, so that there exists a (strictly feasible) primal-dual optimal solution (*i.e.* a primal strictly feasible $\mathbf{x}$ solving 2.1.1 and a dual strictly feasible Z solving 2.1.2).

Then, all inequalities in 2.1.5 become equalities and there is no duality gap ($\eta(\mathbf{x}, Z) = 0$):

$$d^*_{\text{sdp}} = -\mathbf{Tr}(F_0 Z) = c^T \mathbf{x} = p^*_{\text{sdp}} \ . \tag{2.1.6}$$

Thus, we will assume that such a primal-dual optimal solution exists in the sequel. We also introduce the barrier function

$$\Phi(\mathbf{x}) := \begin{cases} \log\det(F(\mathbf{x})^{-1}) & \text{if } F(\mathbf{x}) \succ 0 \\ +\infty & \text{otherwise} \ . \end{cases} \tag{2.1.7}$$

This barrier-function exhibits several nice properties: Φ is strictly convex, analytic and self-concordant (see [NN94] for more details). The unique minimizer $\mathbf{x}^*$ of Φ is called the analytic center of the LMI $F(\mathbf{x}) \succcurlyeq 0$. This self-concordant barrier function guarantees that the number of iterations of the interior-point method is bounded by a polynomial in the dimension (n and m) and the number of accuracy digits of the solution.

We report the best known complexity bounds results for SDPs, from [BTN01, §4.6.3]. We consider an instance (p) of the primal SDP program 2.1.1. An ϵ_{sdp}-solution of (p) is a solution $\mathbf{x} \in \mathbb{R}^m$ of the following feasibility problem:

$$\begin{aligned} c^T \mathbf{x} - p^*_{\text{sdp}} &\leqslant \epsilon_{\text{sdp}} \\ F(\mathbf{x}) &\succcurlyeq -\epsilon_{\text{sdp}} I_n \ . \end{aligned} \tag{2.1.8}$$

We note $\text{Compl}(p, \epsilon_{\text{sdp}})$ the number of real arithmetic operations needed to obtain an ϵ_{sdp}-solution of (p) and $\text{Digits}(p, \epsilon_{\text{sdp}})$ the number of accuracy digits in an ϵ_{sdp}-solution of (p) (following the definition of [BTN01, §4.1.2]). We rewrite the result of [BTN01, §4.6.3] for the arithmetic complexity of ϵ_{sdp}-solution with $k_1 = n$ (for the sake of simplicity, the matrices $F_j, j = 1, \dots, m$ are symmetric matrices with a single block of size $k_1 \times k_1 = n \times n$):

$$\text{Compl}(p, \epsilon_{\text{sdp}}) := O(1)(1 + \sqrt{n})(m^3 + n^2 m^2 + m n^3)\text{Digits}(p, \epsilon_{\text{sdp}}) \ , \tag{2.1.9}$$

where the number of accuracy digits is given by the following (see [BTN01, §4.6.3] for the definitions of $\text{Size}(p)$ and $\text{Data}(p)$):

$$\text{Digits}(p, \epsilon_{\text{sdp}}) := \log\Big(\frac{\text{Size}(p) + \|\text{Data}(p)\|_1 + \epsilon_{\text{sdp}}}{\epsilon^2_{\text{sdp}}}\Big) \ . \tag{2.1.10}$$

2.2 Application of SDP to Polynomial Optimization

Here, we explain how to cast a polynomial optimization problem into a semidefinite program. We consider the general constrained polynomial optimization problem (POP):

$$f^*_{\text{pop}} := \inf_{\mathbf{x} \in K_{\text{pop}}} f_{\text{pop}}(\mathbf{x}) \ , \tag{2.2.1}$$

where $f_{\text{pop}} : \mathbb{R}^n \to \mathbb{R}$ is a d-degree multivariate polynomial, K_{pop} is a compact set defined by polynomials inequalities $g_1(\mathbf{x}) \geqslant 0, \dots, g_m(\mathbf{x}) \geqslant 0$ with $g_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$

being a real-valued polynomial of degree $\omega_i, i = 1, \ldots, m$. We call the feasible set of Problem (2.2.1) the domain over which the optimum is taken, *i.e.* , here K_{pop}.

Let $\mathcal{B}_d$ be the basis of monomials for the d-degree real-valued polynomials in n variables :

$$1, x_1, x_2, \ldots, x_1^2, x_1 x_2, \ldots, x_1 x_n, x_2 x_3, \ldots, x_n^2, \ldots, x_1^d, \ldots, x_n^d \ . \tag{2.2.2}$$

We define $s(d) = \binom{n+d}{d}$ the cardinal of $\mathcal{B}_d$. Let $\mathbb{R}_d[\mathbf{x}]$ be the vector space of real forms in n variables of degree at most d and $\mathbb{R}[\mathbf{x}]$ the set of multivariate polynomials in n variables. The dual space V^* of a given vector space V on $\mathbb{R}$ consists of all linear functionals from V to $\mathbb{R}$. Define $\mathbb{R}_d[\mathbf{x}]^*$ to be the dual space of $\mathbb{R}_d[\mathbf{x}]$. If $f_{\mathrm{pop}} : \mathbb{R}^n \to \mathbb{R}$ is a d-degree multivariate polynomial, we write

$$f_{\mathrm{pop}}(x) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}_d^n} p_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} \ , \tag{2.2.3}$$

where $\mathbf{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ and $\mathbb{N}_d^n := \{\boldsymbol{\alpha} \in \mathbb{N}^n : \sum_i \alpha_i \leqslant d\}$. For a finite set $C \subset \mathbb{N}^n$ such that $\emptyset \neq C \subset \{1, \ldots, n\}$, we also define the support of fully dense polynomials of degree at most d:

$$\mathbb{N}_d^C := \{\boldsymbol{\alpha} \in \mathbb{N}_d^n : \alpha_i = 0 \text{ if } i \notin C\} \ .$$

Let $\#\mathbb{N}_d^C$ be the cardinal of $\mathbb{N}_d^C$.

Let $\mathrm{supp}(f_{\mathrm{pop}})$ stand for the support of f_{pop}:

$$\mathrm{supp}(f_{\mathrm{pop}}) := \{\boldsymbol{\alpha} \in \mathbb{N}_d^n : p_{\boldsymbol{\alpha}} \neq 0\} \ .$$

We assume without loss of generality that $p_0 = 0$. Let $\mathbb{R}_d[\mathbf{x}, C]$ denote the set of polynomials whose support is included in C:

$$\mathbb{R}_d[\mathbf{x}, C] := \{f_{\mathrm{pop}} \in \mathbb{R}_d[\mathbf{x}] : \mathrm{supp}(f_{\mathrm{pop}}) \subset C\} \ .$$

We also define the cone of SOS of degree $2d$, i.e.

$$\Sigma_d[\mathbf{x}] = \Big\{ \sum_i q_i^2, \text{ with } q_i \in \mathbb{R}_d[\mathbf{x}] \Big\} \ . \tag{2.2.4}$$

The set $\Sigma_d[\mathbf{x}]$ is a closed, fully dimensional convex cone in $\mathbb{R}_{2d}[\mathbf{x}]$. Let $\Sigma[\mathbf{x}]$ be the cone of SOS of polynomials in n variables.

We introduce the quadratic module $QM(K_{\mathrm{pop}}) \subset \mathbb{R}[\mathbf{x}]$ associated with $g_1, \ldots, g_m$:

$$QM(K_{\mathrm{pop}}) = \Big\{ \sum_{j=0}^m \sigma_j(\mathbf{x}) g_j(\mathbf{x}) : \sigma_j \in \Sigma[\mathbf{x}] \Big\} \ .$$

Definition 2.1. We say that a quadratic module $QM(K_{\mathrm{pop}})$ is Archimedean if there exists a positive constant M such that the polynomial $\mathbf{x} \mapsto M - \|\mathbf{x}\|_2^2$ belongs to $QM(K_{\mathrm{pop}})$.

We also need the following assumption that will be always satisfied in our case:

Assumption 2.2. *The feasible set K_{pop} is compact and there exists a polynomial $u \in \mathbb{R}_d[\mathbf{x}]$ such that the level set $\{\mathbf{x} \in \mathbb{R}^n : u(\mathbf{x}) \geqslant 0\}$ is compact and u lies in $QM(K_{pop})$.*

We refer the reader to [Sch05] for a comprehensive discussion about equivalent statements for this assumption. In our case, we can always ensure that this assumption holds. Indeed, the nonlinear inequalities that the Flyspeck project tries to prove involve typically a variable $\mathbf{x}$ lying in a box $K \subset \mathbb{R}^n$,

$$K := [m_1, M_1] \times \cdots \times [m_n, M_n] \ . \tag{2.2.5}$$

After normalization and indexing the box inequalities as follows:

$$g_1(\mathbf{x}) := 1 - x_1^2, \ldots, g_n(\mathbf{x}) := 1 - x_n^2 \ , \tag{2.2.6}$$

the polynomial $u(\mathbf{x}) := n - \sum_{j=1}^n x_j^2$ satisfies the Assumption 2.2 since one has:

$$\sum_{j=1}^n g_j(\mathbf{x}) = n - \sum_{j=1}^n x_j^2 \ ,$$

and the level set $\{\mathbf{x} \in \mathbb{R}^n : n - \sum_{j=1}^n x_j^2 \geqslant 0\}$ is compact. Hence, one way to ensure that Assumption (2.2) holds is to add the redundant constraint $n - \sum_{j=1}^n x_j^2 \geqslant 0$ to the set of constraints.

To convexify the problem, we write the equivalent formulation:

$$f_{\text{pop}}^* = \inf_{\mathbf{x} \in K_{\text{pop}}} f_{\text{pop}}(\mathbf{x}) = \inf_{\mu \in \mathcal{P}(K_{\text{pop}})} \int f_{\text{pop}} d\mu \ , \tag{2.2.7}$$

where $\mathcal{P}(K_{\text{pop}})$ is the set of all probability measures μ supported on the set K_{pop}.

Theorem 2.3 (Putinar [Put93])**.** *Suppose that the set K_{pop} satisfies Assumption 2.2. Given a multivariate linear form $L : \mathbb{R}[\mathbf{x}] \to \mathbb{R}$, the following are equivalent:*

1. $\exists \mu \in \mathcal{P}(K_{pop}),\ \forall p \in \mathbb{R}[\mathbf{x}],\ L(p) = \int p \, d\mu.$
2. $L(1) = 1$ *and* $L(s_0 + \sum_{j=1}^m s_j g_j) \geqslant 0$ *for any sum of squares* $s_0, \ldots, s_m \in \Sigma[\mathbf{x}]$.

Hence, we can restate 2.2.7 as:

$$\begin{aligned} f_{\text{pop}}^* = \min \{ & L(f) : L : \mathbb{R}[\mathbf{x}] \to \mathbb{R} \text{ linear }, L(1) = 1 \\ & \text{and each } \mathcal{L}_{g_j} \text{ is semidefinite positive} \} \ , \end{aligned}$$

with $g_0 = 1, \mathcal{L}_{g_0}, \ldots, \mathcal{L}_{g_m}$ defined by:

$$\begin{aligned} \mathcal{L}_{g_j} : \mathbb{R}[\mathbf{x}] \times \mathbb{R}[\mathbf{x}] & \to \mathbb{R} \\ (p, q) & \mapsto L(p \cdot q \cdot g_j) \ . \end{aligned}$$

For any $\boldsymbol{\alpha} \in \mathbb{N}_d^n$, we can define the moment variable $y_{\boldsymbol{\alpha}} = L(\mathbf{x}^{\boldsymbol{\alpha}})$ and associate the *d-truncated moment matrix* M_d to the finite sequence $(y_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha} \in \mathbb{N}_d^n}$:

$$M_d(y)_{u,v} := L(u \cdot v), \ \ u, v \in \mathcal{B}_d \ . \tag{2.2.8}$$

Similarly for each g_j, define the *d-truncated localizing matrix* $M_d(g_j y)$

$$M_d(g_j y)_{u,v} := \mathcal{L}_{g_j}(u \cdot v) = L(u \cdot v \cdot g_j), u, v \in \mathcal{B}_d \ . \tag{2.2.9}$$

Define $\omega \mapsto \lceil \omega/2 \rceil$ to be the function , that returns the least integer value greater than or equal to $\omega/2$. Let $k \geqslant k_0 := \max(\lceil d/2 \rceil, \max_{1 \leqslant j \leqslant m} \lceil \omega_j/2 \rceil)$. Consider the following hierarchy of semidefinite relaxations

$$(Q_k) : \left\{ \begin{array}{rcl} \inf_y L(f_{\text{pop}}) & & \\ M_{k-\lceil \omega_j/2 \rceil}(g_j y) & \succcurlyeq & 0, \quad 0 \leqslant j \leqslant m \ , \\ y_{0,\dots,0} & = & 1 \ , \end{array} \right.$$

with

$$L(f_{\text{pop}}) = \sum_{\alpha} p_\alpha y_\alpha \ .$$

Let give a simple example to illustrate the construction of moment and localizing matrices.

Example 2.4. Consider the quadratic polynomial $g_1(\mathbf{x}) := 2 - x_1^2 - x_2^2$. One has $\omega_1 = 2$ and the moment matrix $M_2(y)$ can be written:

$$M_2(y) = \begin{pmatrix} 1 & | & y_{1,0} & y_{0,1} & | & y_{2,0} & y_{1,1} & y_{0,2} \\ - & - & - & - & - & - & - & - \\ y_{1,0} & | & y_{2,0} & y_{1,1} & | & y_{3,0} & y_{2,1} & y_{1,2} \\ y_{0,1} & | & y_{1,1} & y_{0,2} & | & y_{2,1} & y_{1,2} & y_{0,3} \\ - & - & - & - & - & - & - & - \\ y_{2,0} & | & y_{3,0} & y_{2,1} & | & y_{4,0} & y_{3,1} & y_{2,2} \\ y_{1,1} & | & y_{2,1} & y_{1,2} & | & y_{3,1} & y_{2,2} & y_{1,3} \\ y_{0,2} & | & y_{1,2} & y_{0,3} & | & y_{2,2} & y_{1,3} & y_{0,4} \end{pmatrix}$$

From the first order moment matrix:

$$M_1(y) = \begin{pmatrix} 1 & | & y_{1,0} & y_{0,1} \\ & - & - & - \\ y_{1,0} & | & y_{2,0} & y_{1,1} \\ y_{0,1} & | & y_{1,1} & y_{0,2} \end{pmatrix},$$

we obtain the following localizing matrix ($2 - \lceil \omega_1/2 \rceil = 1$):

$$M_1(g_1 y) = \begin{pmatrix} 2 - y_{2,0} - y_{0,2} & 2y_{1,0} - y_{3,0} - y_{1,2} & 2y_{0,1} - y_{2,1} - y_{0,3} \\ 2y_{1,0} - y_{3,0} - y_{1,2} & 2y_{2,0} - y_{4,0} - y_{2,2} & 2y_{1,1} - y_{3,1} - y_{1,3} \\ 2y_{0,1} - y_{2,1} - y_{0,3} & 2y_{1,1} - y_{3,1} - y_{1,3} & 2y_{0,2} - y_{2,2} - y_{0,4} \end{pmatrix}$$

For instance, the last entry is equal to $M_1(g_1 y)(3,3) = L(g_1(\mathbf{x}) \cdot x_2 \cdot x_2) = L(2x_2^2 - x_1^2 x_2^2 - x_2^4) = 2y_{0,2} - y_{2,2} - y_{0,4}$.

Proposition 2.5 (Lasserre [Las01]). *Let us call* $\inf(Q_k)$ *the optimal value of Problem* Q_k*. The sequence* $(\inf(Q_k))_{k \geqslant k_0}$ *is non-decreasing. Under Assumption (2.2), it converges to* f_{pop}^**.*

Example 2.6 (from Lemma$_{4717061266}$ Flyspeck). Consider the inequality $\forall \mathbf{x} \in K$, $\Delta\mathbf{x} \geqslant 0$, where $K := [4, 6.3504]^6$ and $\Delta\mathbf{x}$ is the polynomial defined in Example 1.3. The optimal value of 128 for the problem $\inf_{\mathbf{x} \in K} \Delta\mathbf{x}$ is obtained at the Q_2 relaxation with $\epsilon_{\text{sdp}} = 10^{-8}$.

Now, we explain how to obtain the dual semidefinite program of Q_k.

Proposition 2.7 (Putinar [Put93]). *When the quadratic module $QM(K_{pop})$ is Archimedean, every polynomial strictly positive on K_{pop} belongs to $QM(K_{pop})$.*

Under Assumption (2.2), the following holds for all positive ϵ:

$$(f_{\text{pop}} - f^*_{\text{pop}} + \epsilon) \in QM(K_{\text{pop}}) \ .$$

The search space for a certificate $\sigma_0, \dots, \sigma_m \in \Sigma[\mathbf{x}]$ that satisfy $f_{\text{pop}} - f^*_{\text{pop}} + \epsilon = \sum_{j=0}^m \sigma_j(\mathbf{x}) g_j(\mathbf{x})$ is infinite, thus we introduce $QM_k(K_{\text{pop}}) \subset QM(K_{\text{pop}})$, which is the k-truncated quadratic module associated with $g_1, \dots, g_m$:

$$QM_k(K_{\text{pop}}) = \Big\{ \sum_{j=0}^m \sigma_j(\mathbf{x}) g_j(\mathbf{x}) : \sigma_j \in \Sigma_{k-\lceil \omega_j/2 \rceil}[\mathbf{x}] \Big\} \ .$$

Furthermore, we consider the following hierarchy of semidefinite relaxations for Problem (2.2.1), consisting of the optimization problems $(Q_k)^*$, $k \geqslant k_0$,

$$(Q_k)^* : \begin{cases} \sup_{\mu, \sigma_j} & \mu \ , \\ \text{s.t.} & f_{\text{pop}}(\mathbf{x}) - \mu = \sum\limits_{j=0}^m \sigma_j(\mathbf{x}) g_j(\mathbf{x}) \ , \\ & \mu \in \mathbb{R}, \qquad \sigma_j \in \Sigma_{k-\lceil \omega_j/2 \rceil}[\mathbf{x}], j = 0, \dots, m \ . \end{cases}$$

Let $\mu_k := \sup((Q_k)^*)$ be the optimal value of $(Q_k)^*$. A feasible point $(\mu_k, \sigma_0, \dots, \sigma_m)$ of Problem $(Q_k)^*$ is said to be an *SOS certificate*, showing the implication $g_1(\mathbf{x}) \geqslant 0, \dots, g_m(\mathbf{x}) \geqslant 0 \implies f_{\text{pop}}(\mathbf{x}) \geqslant \mu_k$. We can refer to the SOS polynomials $\sigma_1, \dots, \sigma_m$ as multipliers associated with the generalized Lagrangian function [KKW04a] $(\mathbf{x}, \boldsymbol{\varphi}) \mapsto f_{\text{pop}}(\mathbf{x}) - \sum\limits_{j=1}^m \sigma_j(\mathbf{x}) g_j(\mathbf{x})$, where $\boldsymbol{\varphi} = (\sigma_1, \dots, \sigma_m)$.

Moreover, let $m_k(\mathbf{x})$ be the vector of monomials $(\mathbf{x}^{\boldsymbol{\alpha}}), \boldsymbol{\alpha} \in \mathbb{N}^n_{k-\lceil \omega_j/2 \rceil}, j = 0, \dots, m$ and consider the expansion:

$$\mathbf{x} \mapsto g_j(\mathbf{x}) m_k^j(\mathbf{x}) m_k^j(\mathbf{x})^T = \sum_{\boldsymbol{\alpha}} C^j_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} \ . \tag{2.2.10}$$

Lemma 2.8. *The following statements are equivalent:*

1. *$(f_{pop}(\mathbf{x}) - \mu) \in M_k(K_{pop})$*
2. *There exist some real symmetric matrices $Z_0, Z_1, \dots, Z_m$ such that:*

$$\begin{cases} -\mu & = & \sum\limits_{j=0}^m \mathbf{Tr}(Z_j C^j_0) \ , & \\ p_{\boldsymbol{\alpha}} & = & \sum\limits_{j=0}^m \mathbf{Tr}(Z_j C^j_{\boldsymbol{\alpha}}), & \boldsymbol{\alpha} \in \mathbb{N}^n_{2k}, \boldsymbol{\alpha} \neq 0 \ , \\ Z_j & \in & \mathcal{S}_{s(k-\lceil \omega_j/2 \rceil)}, & j = 0, \dots, m \ , \\ Z_j & \succcurlyeq & 0, & j = 0, \dots, m \ . \end{cases}$$

Proof. It follows from the fact that a polynomial $u \in \mathbb{R}[\mathbf{x}]_{2k}$ is of the form $\sigma_j g_j$ if and only if there exists some real semidefinite positive symmetric matrix Z_j such that $Z_j \in \mathcal{S}_{s(k-\lceil \omega_j/2 \rceil)}$ and $u_{\boldsymbol{\alpha}} = \mathbf{Tr}(Z_j C^j_{\boldsymbol{\alpha}}), \boldsymbol{\alpha} \in \mathbb{N}^n_{2k}$. □

Therefore, $(Q_k)^*$ is equivalent to:

$$\widetilde{(Q_k)^*}\begin{cases} \sup_{Z_j,\mu} & -\sum\limits_{j=0}^{m} \mathbf{Tr}(Z_j C_0^j) \ , \\ & 0 = \sum\limits_{j=0}^{m} \mathbf{Tr}(Z_j C_0^j) + \mu \ , \\ \text{s.t.} & p_{\boldsymbol{\alpha}} = \sum\limits_{j=0}^{m} \mathbf{Tr}(Z_j C_{\boldsymbol{\alpha}}^j), \qquad \boldsymbol{\alpha} \in \mathbb{N}_{2k}^n, \boldsymbol{\alpha} \neq 0 \ , \\ & Z_j \in \mathcal{S}_{s(k-\lceil \omega_j/2 \rceil)}, \qquad j = 0, \ldots, m \ , \\ & Z_j \succcurlyeq 0, \qquad j = 0, \ldots, m \ . \end{cases}$$

To see that $\widetilde{(Q_k)^*}$ is the dual semidefinite program of Q_k, one can rewrite the localizing matrices:

$$M_{k-\lceil \omega_j/2 \rceil}(g_j y) = \sum_{\boldsymbol{\alpha}} C_{\boldsymbol{\alpha}}^j y_{\boldsymbol{\alpha}}, j = 0, \ldots, m \ .$$

This reformulation can be seen as the linearisation of (2.2.10).

2.3 Application of SDP to Semialgebraic Optimization

In this section, we describe how to extend the previous approach to semialgebraic optimization by introducing lifting variables.

Let $\mathcal{A}$ be the semialgebraic functions algebra obtained by composition of polynomials with $|\cdot|, +, -, \times, /, \sup(\cdot,\cdot), \inf(\cdot,\cdot), (\cdot)^{\frac{1}{q}} (q \in \mathbb{N}_0)$. Given a semialgebraic function $f_{\text{sa}} \in \mathcal{A}$, we consider the problem:

$$f_{\text{sa}}^* = \inf_{\mathbf{x} \in K_{\text{sa}}} f_{\text{sa}}(\mathbf{x}) \ , \tag{2.3.1}$$

where $K_{\text{sa}} := \{\mathbf{x} \in \mathbb{R}^n : g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0\}$ is a basic semialgebraic set.

Definition 2.9 (Basic Semialgebraic Lifting)**.** A semialgebraic function f_{sa} is said to have a basic semialgebraic lifting if there exist $p, s \in \mathbb{N}$, polynomials $h_1, \ldots, h_s \in \mathbb{R}[\mathbf{x}, z_1, \ldots, z_p]$ and a basic semialgebraic set K_{pop} defined by:

$$K_{\text{pop}} := \{(\mathbf{x}, z_1, \ldots, z_p) \in \mathbb{R}^{n+p} : \mathbf{x} \in K_{\text{sa}}, h_1(\mathbf{x}, \mathbf{z}) \geqslant 0, \ldots, h_s(\mathbf{x}, \mathbf{z}) \geqslant 0\} \ ,$$

such that the graph of f_{sa} (denoted $\Psi_{f_{\text{sa}}}$) satisfies:

$$\Psi_{f_{\text{sa}}} := \{(\mathbf{x}, f_{\text{sa}}(\mathbf{x})) : \mathbf{x} \in K_{\text{sa}}\} = \{(\mathbf{x}, z_p) : (\mathbf{x}, \mathbf{z}) \in K_{\text{pop}}\} \ .$$

Lemma 2.10 (Lasserre, Putinar [LP10])**.** *Every well-defined $f_{sa} \in \mathcal{A}$ has a basic semialgebraic lifting.*

To ensure that Assumption 2.2 is preserved, we add bound constraints over the lifting variables. These bounds are computed by solving semialgebraic optimization subproblems. The lower (resp. upper) bounds are obtained by calling the function `min_sa` (resp. `max_sa`). The function `min_sa` is described in Figure 2.1. Given a function f_{sa} assimilated with its abstract syntax tree, a semialgebraic set K_{sa} and an SDP relaxation order k, one wants to obtain a lower bound of $\inf_{\mathbf{x}\in K_{\mathrm{sa}}} f_{\mathrm{sa}}(\mathbf{x})$. The first step of the algorithm is to call the auxiliary function `sa_lift` (see Line 1), which introduces extra lifting variables $\mathbf{z}$, a polynomial objective function f_{pop} and a semialgebraic set K_{pop}. Then, the auxiliary function `min_pop` (at Line 2) returns a lower bound of $\inf_{(\mathbf{x},\mathbf{z})\in K_{\mathrm{pop}}} f_{\mathrm{pop}}(\mathbf{x},\mathbf{z})$ by solving the semidefinite relaxation Q_k.

```
Input: Semialgebraic objective function tree $f_{\mathrm{sa}}$, semialgebraic set $K_{\mathrm{sa}}$, variables $\mathbf{x} := (x_1,\dots,x_n)$, SDP relaxation order $k$
Output: lower bound $m$
1: $f_{\mathrm{pop}}, K_{\mathrm{pop}}, (\mathbf{x},\mathbf{z}) :=$ sa_lift($f_{\mathrm{sa}}, K_{\mathrm{sa}}, \mathbf{x}, k$)
2: return min_pop($f_{\mathrm{pop}}, K'_{\mathrm{pop}}, (\mathbf{x},\mathbf{z}), k$)
```

Figure 2.1: `min_sa`

The auxiliary function `sa_lift` (Figure 2.2) reduces the semialgebraic optimization problems to polynomial optimization problems. To describe this algorithm, we assimilate the objective function f_{sa} with its abstract syntax tree.

We assume that the leaves of f_{sa} are multivariate polynomial functions and other nodes are monadic operations `uop` $\in \{(\cdot)^{\frac{1}{q}}(q \in \mathbb{N}_0), |\cdot|\}$ or dyadic operations `bop` $\in \{\max(\cdot,\cdot), \min(\cdot,\cdot), +, -, \times, /\}$.

The simplest case occurs when f_{sa} is a multivariate polynomial. The function `sa_lift` returns the polynomial objective function $f_{\mathrm{sa}} := f_{\mathrm{pop}}$, the semialgebraic set $K_{\mathrm{pop}} := K_{\mathrm{sa}}$ and the initial set of variables $\mathbf{x}$ (Line 1).

If the root of the tree corresponds to a binary operation `bop` with children f_1 and f_2 (Line 2), the function `sa_lift` is applied recursively to get two semialgebraic sets ($K_{\mathrm{pop},1}$ for f_1 and $K_{\mathrm{pop},2}$ for f_2), two sets of lifting variables and two objective polynomial functions ($f_{\mathrm{pop},1}$ for f_1 and $f_{\mathrm{pop},2}$ for f_2). We merge the two sets of lifting variables (Line 5) and we define a semialgebraic set K'_{pop} obtained by combining the polynomial inequality constraints that define $K_{\mathrm{pop},1}$ and $K_{\mathrm{pop},2}$ (Line 6). Moreover, we distinguish six cases, according to `bop`. When $f_{\mathrm{sa}} = \max(f_1, f_2)$ (Line 7), then we use the following identity:

$$2\max(f_1, f_2) = f_1 + f_2 + \sqrt{(f_1 - f_2)^2},$$

and introduce an additional lifting variable z_{p+1} to represent the square root. Thus, we define the semialgebraic set K_{pop} by adding the equality $z_{p+1}^2 - (f_{\mathrm{pop},1} - f_{\mathrm{pop},2})^2 = 0$ (that can be written with two inequalities) and the inequality $z_{p+1} \geqslant 0$ to the inequality constraints of K'_{pop} (see Line 9). The lifting strategy is analogous when `bop` $:= \min$ (Line 11). Another interesting case is when `bop` is the division operator (Line 18). One needs to check if the denominator f_2 has a constant sign on K_{sa} and raise an exception otherwise. The other cases are analogous.

Remark 2.11. We emphasize the fact that our current implementation contains some optimizations to obtain less lifting variables. In particular, when `bop` is the multiplication operator and when a child (let say f_1 without loss of generality) is a multivariate polynomial, it is not mandatory to use a lifting variable to represent f_1. However, the degree of the resulting polynomial objective function becomes $(\deg(f_1) + 1)$ (instead of 2 with a full lifting strategy). This issue is discussed in more details in Chapter 6. The reader can get an overview of the algorithm behaviour with the Example 2.12.

```
Input: Semialgebraic objective function tree $f_{\mathrm{sa}} \in \mathcal{A}$, semialgebraic set $K_{\mathrm{sa}}$, variables
    $\mathbf{x} := (x_1, \dots, x_n)$, SDP relaxation order $k$
Output: Polynomial objective function tree $f_{\mathrm{pop}}$, semialgebraic set $K_{\mathrm{pop}}$, variables
    $(\mathbf{x}, \mathbf{z}) := (x_1, \dots, x_n, z_1, \dots, z_p)$
if $f_{\mathrm{sa}}$ is a multivariate polynomial then return $f_{\mathrm{sa}}, K_{\mathrm{sa}}, \mathbf{x}$
else if $f_{\mathrm{sa}} = \mathtt{bop}(f_1, f_2)$ then
    $f_{\mathrm{pop},1}, K_{\mathrm{pop},1}, (\mathbf{x}_1, \mathbf{z}_1) := \mathtt{sa\_lift}(f_1, K_{\mathrm{sa}}, \mathbf{x}, k)$
    $f_{\mathrm{pop},2}, K_{\mathrm{pop},2}, (\mathbf{x}_2, \mathbf{z}_2) := \mathtt{sa\_lift}(f_2, K_{\mathrm{sa}}, \mathbf{x}, k)$
    $(\mathbf{x}, z_1, \dots, z_p) := (\mathbf{x}_1, \mathbf{z}_1) \cup (\mathbf{x}_2, \mathbf{z}_2)$
    $K'_{\mathrm{pop}} := \{(\mathbf{x}, \mathbf{z}) : (\mathbf{x}_1, \mathbf{z}_1) \in K_{\mathrm{pop},1}, (\mathbf{x}_2, \mathbf{z}_2) \in K_{\mathrm{pop},2}\}$
    if $f_{\mathrm{sa}} = \max(f_1, f_2)$ then
        $f_{\mathrm{pop}} := (f_{\mathrm{pop},1} + f_{\mathrm{pop},2} + z_{p+1})/2$
        $K_{\mathrm{pop}} := \{(\mathbf{x}, \mathbf{z}, z_{p+1}) : (\mathbf{x}, \mathbf{z}) \in K'_{\mathrm{pop}}, z_{p+1}^2 - (f_{\mathrm{pop},1} - f_{\mathrm{pop},2})^2 = 0, z_{p+1} \geqslant 0\}$
        return $f_{\mathrm{pop}}, K_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z}, z_{p+1})$
    else if $f_{\mathrm{sa}} = \min(f_1, f_2)$ then
        $f_{\mathrm{pop}} := (f_{\mathrm{pop},1} + f_{\mathrm{pop},2} - z_{p+1})/2$
        $K_{\mathrm{pop}} := \{(\mathbf{x}, \mathbf{z}, z_{p+1}) : (\mathbf{x}, \mathbf{z}) \in K'_{\mathrm{pop}}, z_{p+1}^2 - (f_{\mathrm{pop},1} - f_{\mathrm{pop},2})^2 = 0, z_{p+1} \geqslant 0\}$
        return $f_{\mathrm{pop}}, K_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z}, z_{p+1})$
    else if $f_{\mathrm{sa}} = f_1 \times f_2$ then return $f_{\mathrm{pop},1} \times f_{\mathrm{pop},2}, K'_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z})$
    else if $f_{\mathrm{sa}} = f_1 - f_2$ then return $f_{\mathrm{pop},1} - f_{\mathrm{pop},2}, K'_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z})$
    else if $f_{\mathrm{sa}} = f_1 + f_2$ then return $f_{\mathrm{pop},1} + f_{\mathrm{pop},2}, K'_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z})$
    else if $f_{\mathrm{sa}} = f_1 / f_2$ then
        $m = \mathtt{min\_pop}(f_{\mathrm{pop},2}, K'_{\mathrm{pop}}, k)$
        $M = \mathtt{max\_pop}(f_{\mathrm{pop},2}, K'_{\mathrm{pop}}, k)$
        if $0 \in [m, M]$ then return Division Exception
        end
        $K_{\mathrm{pop}} := \{(\mathbf{x}, \mathbf{z}, z_{p+1}) : (\mathbf{x}, \mathbf{z}) \in K'_{\mathrm{pop}}, z_{p+1} f_{\mathrm{pop},2} = f_{\mathrm{pop},1}\}$
        return $z_{p+1}, K_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z}, z_{p+1})$
    end
else if $f_{\mathrm{sa}} = (f)^{\frac{1}{q}}$ then
    $f_{\mathrm{pop}}, K'_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z}) := \mathtt{sa\_lift}(f, K_{\mathrm{sa}}, \mathbf{x}, k)$
    $m = \mathtt{min\_pop}(f_{\mathrm{pop}}, K'_{\mathrm{pop}}, k)$
    if $m < 0$ then return Inverse Power Exception
    end
    $K_{\mathrm{pop}} := \{(\mathbf{x}, \mathbf{z}, z_{p+1}) : (\mathbf{x}, \mathbf{z}) \in K'_{\mathrm{pop}}, z_{p+1}^q = f_{\mathrm{pop}}, z_{p+1} \geqslant 0\}$
    return $z_{p+1}, K_{\mathrm{pop}}, (\mathbf{x}, \mathbf{z}, z_{p+1})$
else if $f_{\mathrm{sa}} = |f|$ then return $\mathtt{sa\_lift}(\sqrt{f^2}, K_{\mathrm{sa}}, \mathbf{x}, k)$
end
```

Figure 2.2: sa_lift

The max_sa function is obtained using the following:

$$\sup_{\mathbf{x} \in K_{\mathrm{sa}}} f_{\mathrm{sa}}(\mathbf{x}) = - \inf_{\mathbf{x} \in K_{\mathrm{sa}}} f_{\mathrm{sa}}(\mathbf{x}) \ .$$

Example 2.12 (from Lemma$_{9922699028}$ Flyspeck). Continuing Example 1.3, we consider the function $f_{\mathrm{sa}} := \frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}}$ and the set $K_{\mathrm{sa}} := [4, 6.3504]^3 \times [6.3504, 8] \times [4, 6.3504]^2$.

The latter can be equivalently rewritten as

$$K_{\mathrm{sa}} := \{\mathbf{x} \in \mathbb{R}^6 : g_1(\mathbf{x}) \geqslant 0, \dots, g_{12}(\mathbf{x}) \geqslant 0\} \ ,$$

where $g_1(\mathbf{x}) := x_1 - 4, g_2(\mathbf{x}) := 6.3504 - x_1, \dots, g_{11}(\mathbf{x}) := x_6 - 4, g_{12}(\mathbf{x}) := 6.3504 - x_6$.

We introduce two lifting variables z_1 and z_2, respectively representing the terms $\sqrt{4x_1 \Delta\mathbf{x}}$ and $\frac{\partial_4 \Delta\mathbf{x}}{\sqrt{4x_1\Delta\mathbf{x}}}$.

We also use a lower bound m_1 of $\inf_{\mathbf{x}\in K_{\mathrm{sa}}} \sqrt{4x_1\Delta\mathbf{x}}$ and an upper bound M_1 of $\sup_{\mathbf{x}\in K_{\mathrm{sa}}} \sqrt{4x_1\Delta\mathbf{x}}$ which can be both computed by solving auxiliary subproblems.

Now the basic semialgebraic set K_{pop} and the graph $\Psi_{f_{\mathrm{sa}}}$ of f_{sa} can be defined as follows:

$$\begin{aligned} K_{\mathrm{pop}} &:= \{(\mathbf{x}, z_1, z_2) \in \mathbb{R}^{6+2} : \mathbf{x} \in K_{\mathrm{sa}}, h_j(\mathbf{x}, z_1, z_2) \geqslant 0, j = 1, \dots, 6\} \ , \\ \Psi_{f_{\mathrm{sa}}} &:= \{(\mathbf{x}, f_{\mathrm{sa}}(\mathbf{x})) : \mathbf{x} \in K_{\mathrm{sa}}\} = \{(\mathbf{x}, z_2) : (\mathbf{x}, z_1, z_2) \in K_{\mathrm{pop}}\} \ , \end{aligned}$$

where the multivariate polynomials h_j are defined by:

$$\begin{aligned} h_1(\mathbf{x},\mathbf{z}) &:= z_1 - m_1 \ , & h_4(\mathbf{x},\mathbf{z}) &:= -z_1^2 + 4x_1\Delta\mathbf{x} \ , \\ h_2(\mathbf{x},\mathbf{z}) &:= M_1 - z_1 \ , & h_5(\mathbf{x},\mathbf{z}) &:= z_2 z_1 - \partial_4\Delta\mathbf{x} \ , \\ h_3(\mathbf{x},\mathbf{z}) &:= z_1^2 - 4x_1\Delta\mathbf{x} \ , & h_6(\mathbf{x},\mathbf{z}) &:= -z_2 z_1 + \partial_4\Delta\mathbf{x} \ . \end{aligned}$$

Let $\omega_j := \deg h_j$, $1 \leqslant j \leqslant 7$. The moment variable associated with z_2 is $y_{0,\dots,0,1}$. Then, consider the following semidefinite relaxations:

$$Q_k^{sa} : \left\{ \begin{array}{rcl} & \inf_y y_{0,\dots,0,1} & \\ M_{k-1}(g_i\, y) & \succcurlyeq & 0,\ 1 \leqslant i \leqslant 12 \ , \\ M_{k-\lceil \omega_j/2 \rceil}(h_j\, y) & \succcurlyeq & 0,\ 1 \leqslant j \leqslant 6 \ , \\ y_{0,\dots,0} & = & 1 \ . \end{array} \right.$$

If $k \geqslant k_0 := \max_{1\leqslant j \leqslant 7}\{\lceil \omega_j/2 \rceil\} = 2$, then as a special case of Proposition 2.5, the sequence $(\inf(Q_k^{sa}))_{k\geqslant 2}$ is monotonically non-decreasing and converges to f_{sa}^*. The lower bound $m_2 = -0.618$ computed at the Q_2^{sa} relaxation is too coarse. A tighter lower bound $m_3 = -0.445$ is obtained at the third relaxation, but it consumes more CPU time.

Next, we recall how to reduce the size of the SDP relaxations (Q_k), by using a sparse variant of Lasserre's hierarchy.

2.4 Exploiting the System Properties

Let $f_{\mathrm{sa}} \in \mathcal{A}$ be a semialgebraic function and consider Problem (2.3.1). Let n be the number of variables involved in the polynomials that define the semialgebraic set K_{sa} and p the number of lifting variables that is needed to define a basic semialgebraic lifting for f_{sa}. Let note m the number of inequalities that define K_{pop}.

The size of the truncated moment SDP matrices $M_k(y)$ (as well as the size of the localizing matrices) grows polynomially with the SDP-relaxation order k. Indeed, at fixed n_{pop}, the relaxation Q_k involves $O((2k)^{n_{\mathrm{pop}}})$ SDP moment variables and $(m + 1)$

linear matrix inequalities (LMIs) of size $O(k^{n_{\text{pop}}})$. When k increases, then more accurate lower bounds of f^*_{pop} (as well as f^*_{sa}) can be obtained, at an increasing computational cost, even though the theoretical convergence of the sequence $(\inf(Q_k))_k$ holds. At fixed k, the relaxation Q_k involves $O(n_{\text{pop}}^{2k})$ SDP moment variables and $(m+1)$ linear matrix inequalities (LMIs) of size $O(n_{\text{pop}}^k)$.

There are several ways to decrease the size of these matrices. First, symmetries in SDP relaxations for polynomial optimization problems can be exploited to replace one SDP problem Q_k by several smaller SDPs [RTAL11]. Notice that it is possible only if the multivariate polynomials of the initial problem are invariant under the action of a finite subgroup G of the group $GL_{n_{\text{pop}}}(\mathbb{R})$.

Here we describe how to exploit the structured sparsity of the problem to replace one SDP problem Q_k by an SDP problem of size $O(\kappa^{2k})$ where κ is the average size of the maximal cliques correlation pattern of the polynomial variables (see [WKK+08]). These techniques have been successfully implemented in our `NLCertify` tool (see Appendix B).

For the sake of simplicity, we set $x_{n+i} := z_i, i = 1, \ldots, p$, $K_{\text{pop}} := \{\mathbf{x} \in \mathbb{R}^{n_{\text{pop}}} : g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0\}$ and consider the polynomial optimization problem:

$$\inf_{\mathbf{x} \in K_{\text{pop}}} f_{\text{pop}}(\mathbf{x}) \ , \tag{2.4.1}$$

Let F_k be the index set of variables which are involved in the polynomial g_k.

The correlative sparsity is represented by the $n_{\text{pop}} \times n_{\text{pop}}$ correlative sparsity matrix (csp matrix) $\mathbf{R}$ defined by:

$$\mathbf{R}(i,j) = \begin{cases} 1 & \text{if } i = j \ , \\ 1 & \text{if } \exists \boldsymbol{\alpha} \in \operatorname{supp}(f_{\text{pop}}) \text{ such that } \alpha_i \geqslant 1 \text{ and } \alpha_j \geqslant 1 \ , \\ 1 & \text{if } \exists k \in \{1, \ldots, m\} \text{ such that } i \in F_k \text{ and } j \in F_k \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{2.4.2}$$

We define the undirected csp graph $G(N_{\text{pop}}, E_{\text{pop}})$ with:

$$N_{\text{pop}} = \{1, \ldots, n_{\text{pop}}\} \text{ and } E_{\text{pop}} = \{\{i,j\} : i, j \in N_{\text{pop}}, i < j, \mathbf{R}(i,j) = 1\} \ .$$

Then, let $C_1, \ldots, C_l \subset N_{\text{pop}}$ denote the maximal cliques of $G(N_{\text{pop}}, E_{\text{pop}})$ and define the sets of supports:

$$\mathbb{N}_d^{C_q} := \{\boldsymbol{\alpha} \in \mathbb{N}_d^{n_{\text{pop}}} : \alpha_i = 0 \text{ if } i \notin C_q\}, \ (q = 1, \ldots, l) \ .$$

Define $n_q := \# C_q$ $(q = 1, \ldots, l)$.

We assume that the module $QM(K_{\text{pop}})$ is archimedean and that there is some $M > 0$ such that $M - \sum_{i=1}^{n_{\text{pop}}} x_i^2 \geqslant 0$. Hence, we can add the q redundant additional constraints:

$$g_{m+q} := n_q M^2 - \sum_{i \in C_q} x_i^2 \geqslant 0, \ q = 1, \ldots, l \ , \tag{2.4.3}$$

set $m' = m + q$, define the compact semialgebraic set:

$$K'_{\text{pop}} := \{\mathbf{x} \in \mathbb{R}^{n_{\text{pop}}} : g_1(\mathbf{x}) \geqslant 0, \ldots, g'_m(\mathbf{x}) \geqslant 0\} \ ,$$

and modify Problem (2.4.1) into the following optimization problem:

$$f^*_{\text{pop}} := \inf_{\mathbf{x} \in K'_{\text{pop}}} f_{\text{pop}}(\mathbf{x}) \ , \tag{2.4.4}$$

For each clique C_q, we also define the *sparse d-truncated moment matrix $M_d(y, C_q)$*

$$M_d(y, C_q)_{u,v} := L(u \cdot v), \quad u, v \in \mathcal{B}_d \cap \mathbb{R}_d[\mathbf{x}, \mathbb{N}_d^{C_q}] \ . \tag{2.4.5}$$

Similarly for each g_j (and the associated index set F_j), define the *sparse d-truncated localizing matrix $M_d(g_j y, F_j)$*:

$$M_d(g_j y, F_j)_{u,v} := \mathcal{L}_{g_j}(u \cdot v) = L(u \cdot v \cdot g_j), \quad u, v \in \mathcal{B}_d \cap \mathbb{R}_d[\mathbf{x}, \mathbb{N}_d^{F_j}] \ . \tag{2.4.6}$$

Hence, we can define the sparse variant of the primal semidefinite relaxations Q_k:

$$Q_k^{\text{sparse}} : \left\{ \begin{array}{rcl} \inf_y L(f_{\text{pop}}) & & \\ M_k(g_j y, C_q) & \succcurlyeq & 0, \quad 1 \leqslant q \leqslant l \ , \\ M_{k-\lceil \omega_j/2 \rceil}(g_j y, F_j) & \succcurlyeq & 0, \quad 1 \leqslant j \leqslant m' \ , \\ y_{0,\dots,0} & = & 1 \ . \end{array} \right.$$

For each $q = 1, \dots, l$, We also define the cone of sums of squares of polynomials in $\mathbb{R}_d[\mathbf{x}, \mathbb{N}_d^{C_q}]$, $(q = 1, \dots, l)$:

$$\Sigma[\mathbf{x}, \mathbb{N}_d^{C_q}] := \Big\{ \sum_i q_i^2, \text{ with } q_i \in \mathbb{R}_d[\mathbf{x}, \mathbb{N}_d^{C_q}] \Big\} \ . \tag{2.4.7}$$

Notice that the sums of squares of polynomials that belong to $\Sigma[\mathbf{x}, \mathbb{N}_d^{C_q}]$ only involve variables $x_i (i \in C_q)$.

The dual of Q_k^{sparse} is the sparse variant of the dual semidefinite relaxations $(Q_k)^*$:

$$(Q_k^{\text{sparse}})^* : \left\{ \begin{array}{ll} \sup_{\mu, \sigma_j} & \mu \ , \\ \text{s.t.} & f_{\text{pop}}(\mathbf{x}) - \mu = \sum_{j=0}^{m'} \sigma_j(\mathbf{x}) g_j(\mathbf{x}) \ , \\ & \mu \in \mathbb{R}, \qquad \sigma_j \in \Sigma[\mathbf{x}, \mathbb{N}_{k-\lceil \omega_j/2 \rceil}^{F_j}], j = 1, \dots, m' \ , \\ & \sigma_0 \in \sum_{q=1}^{l} \Sigma[\mathbf{x}, \mathbb{N}_k^{C_q}] \ . \end{array} \right.$$

The interested reader can find more details about the properties of these semidefinite relaxations in [WKKM06]. Since the cliques $C_1, \dots, C_l$ satisfy the running intersection property (see Definition 2.13 below), the optimal values of Q_k^{sparse} converge to the global minimum f_{pop}^*, as a corollary of [Las, Theorem 3.6].

Definition 2.13 (Running Intersection Property). Let $q \in \mathbb{N}_0, I_1, \dots, I_q \subset \{1, \dots, n\}$. We say that $I_1, \dots, I_q$ satisfy the running intersection property if:

$$\forall i = 2, \dots, r, (\exists k < i, (I_i \cap \bigcup_{j<i} I_j) \subset I_k) \ .$$

We illustrate the benefits of these sparse semidefinite relaxations with the following example:

Example 2.14 (from Lemma$_{9922699028}$ Flyspeck). Consider the polynomial $\partial_4 \Delta \mathbf{x}$ of the inequality of Example 2.12:

$$\partial_4 \Delta \mathbf{x} := x_1(-x_1 + x_2 + x_3 - x_4 + x_5 + x_6) + x_2 x_5 + x_3 x_6 - x_2 x_3 - x_5 x_6 \ .$$

Here, $n_{\text{pop}} = 6, d = 2, N_{\text{pop}} = \{1, \ldots, 6\}$. The 6×6 correlative sparsity matrix $\mathbf{R}$ is:

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

The csp graph G associated to $\mathbf{R}$ is depicted in Figure 2.3. The maximal cliques of G are:

$$C_1 := \{1, 4\}, C_2 := \{1, 2, 3, 5\}, C_3 := \{1, 3, 5, 6\} \ .$$

For instance, the set of supports associated with the maximal clique C_1 for monomials of degree at most 2 is:

$$\begin{aligned} \mathbb{N}_2^{C_1} := \{ & (0,0,0,0,0,0), (1,0,0,0,0,0), (0,0,0,1,0,0) \ , \\ & (2,0,0,0,0,0), (1,0,0,1,0,0), (0,0,0,2,0,0) \} \ . \end{aligned}$$

Now consider the second order semidefinite relaxation. In the case of the dense semidefinite relaxation, the number of SDP variables is $\#\mathbb{N}_4^6 = \binom{6+4}{4} = 210$ since we are considering sums of squares of degree at most 4. Conversely, in the case of the sparse relaxation, this number is $\#(\mathbb{N}_4^{C_1} \cup \mathbb{N}_4^{C_2} \cup \mathbb{N}_4^{C_3}) = 115$. The dense moment matrix is a single block diagonal matrix of size 28, while the sparse moment matrix is a block diagonal matrix with a 6×6 block and two 15×15 blocks.

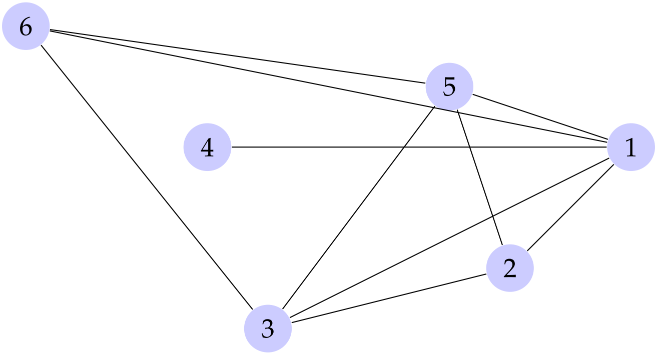


Figure 2.3: Correlative sparsity pattern graph for the variables of $\partial_4 \Delta \mathbf{x}$

One can also reduce the size of the sparse semidefinite relaxations by eliminating the redundant elements that never appear in any SOS representations of the generalized Lagrangian f_L

$$f_L : (\mathbf{x}, \boldsymbol{\varphi}) \mapsto f_{\text{pop}}(\mathbf{x}) - \sum_{j=1}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x}) \ ,$$

where $\boldsymbol{\varphi} = (\sigma_1, \ldots, \sigma_m)$. Let $\mathcal{F}_L$ be the support of the polynomial f_L and let $\mathcal{G}^0 := \bigcup_{q=1}^{l} \mathbb{N}_k^{C_q}$. One also need to define the set of integer vectors of $\mathcal{F}_L$ with even coordinates $\mathcal{F}_L^e := \mathcal{F}_L \cap (2\mathbb{N}^n)$. Then, following the phase 2 of the algorithm described

```
type vertex = Gp of alpha | Fe of alpha
module G = Graph.Persistent.Digraph.ConcreteBidirectional (
    struct
  type t = vertex let compare = compare_vertex
  let hash = Hashtbl.hash let equal = eq_vertex end)

exception Noedges of vertex

let raise_no_outgoing_edges g vertex = match vertex with
  | Gp alpha when G.out_degree g vertex = 0 -> raise (
     Noedges vertex)
  | Fe _ -> ()

let rec elim_sos_phase2 g =
  try G.iter_vertex (raise_no_outgoing_edges g) g; g
  with Noedges vertex ->
    begin
      let g = G.remove_vertex g vertex in
        elim_sos_phase2 g
    end
```

Figure 2.4: A procedure to eliminate the redundant vectors for any SOS representations

in [KKW04b], we build the directed graph $G(V, E)$, where the set of vertices V is defined this way:

$$V := V_{\mathcal{G}^0} \cup V_{\mathcal{F}_L^e} \ ,$$

with

$$V_{\mathcal{G}^0} := \{v_{\boldsymbol{\alpha},\mathcal{G}^0} : \boldsymbol{\alpha} \in \mathcal{G}^0\}, V_{\mathcal{F}_L^e} := \{v_{\boldsymbol{\alpha},\mathcal{F}_L^e} : \boldsymbol{\alpha} \in \mathcal{F}_L^e\} \ .$$

One has two types of edges for $E := E_{\mathcal{G}^0,\mathcal{F}_L^e} \cup E_{\mathcal{G}^0}$:

$$E_{\mathcal{G}^0,\mathcal{F}_L^e} := \{\{v_{\boldsymbol{\alpha},\mathcal{G}^0}, v_{\boldsymbol{\gamma},\mathcal{F}_L^e}\} : v_{\boldsymbol{\alpha},\mathcal{G}^0} \in V_{\mathcal{G}^0}, v_{\boldsymbol{\gamma},\mathcal{F}_L^e} \in V_{\mathcal{F}_L^e} \text{ and } 2\boldsymbol{\alpha} = \boldsymbol{\gamma}\} \ ,$$
$$E_{\mathcal{G}^0} := \{\{v_{\boldsymbol{\alpha},\mathcal{G}^0}, v_{\boldsymbol{\gamma},\mathcal{G}^0}\} : v_{\boldsymbol{\alpha},\mathcal{G}^0}, v_{\boldsymbol{\gamma},\mathcal{G}^0} \in V_{\mathcal{G}^0} \text{ and } (2\boldsymbol{\alpha} - \boldsymbol{\gamma}) \in \mathcal{G}^0 \text{ and } \boldsymbol{\alpha} \neq \boldsymbol{\gamma}\}.$$

If a node $v_{\boldsymbol{\alpha},\mathcal{G}^0} \in V$ has no outgoing edges, then the integer vector $\boldsymbol{\alpha}$ is redundant (see [KKW04b] for more details), thus we can eliminate $\boldsymbol{\alpha}$ from $\mathcal{G}^0$. Then, we actualise $\mathcal{G}^0$ and repeat the procedure until the digraph G has no nodes $v_{\boldsymbol{\alpha},\mathcal{G}^0}$ such that $\boldsymbol{\alpha}$ is redundant. We implemented this procedure in OCAML using the `ocamlgraph` library [CFS07]. The code of the main function is available in Figure 2.4.

2.5 Hybrid Symbolic-Numeric Certification

The previous semidefinite relaxations Q_k and $(Q_k)^*$ (as well as the sparse variants mentioned in Section 2.4) can be solved with SDP solvers (e.g. SDPA [YFN$^+$10]). These solvers are implemented using floating-point arithmetic. In order to build formal proofs, we currently rely on exact rational certificates which are needed to make formal proofs: COQ, being built on a computational formalism, is well equipped for checking the correctness of such certificates.

Such rational certificates can be obtained by the rationalization scheme (rounding and projection algorithm) developed by Peyrl and Parrilo [PP08], with an improvement of Kaltofen et al. [KLYZ12]. Note that when the SDP formulation of the polynomial optimization problem is not strictly feasible, then the rounding and projection algorithm may fail. However, Monniaux and Corbineau proposed a partial workaround for this issue [MC11]. In this way, except in degenerate situations, we arrive at a candidate SOS certificate with rational coefficients, $(\mu, \sigma_0, \dots, \sigma_m)$ from the floating point solution of $(Q_k)^*$.

In practice, the SDP solvers solve the formulation $\widetilde{(Q_k)^*}$ (which is equivalent to $(Q_k)^*$) and return floating-point symmetric matrices $Z_0, \dots, Z_m$ and a lower bound μ_k. We know that for every optimal solution $Z_0, \dots, Z_m$ of $\widetilde{(Q_k)^*}$ (from [Las01, Theorem 4.2 (b)]):

$$f_{\mathrm{pop}}(\mathbf{x}) - \mu_k = \sum_{j=0}^{m} g_j(\mathbf{x}) \sum_{i=1}^{r_j} \lambda_{ij} v_{ij}^2(\mathbf{x}) \enspace , \tag{2.5.1}$$

where the vectors of coefficients of the polynomials v_{ij} are the eigenvectors of Z_j with the associated r_j eigenvalues λ_{ij}. One has the following decomposition:

$$\sigma_j := \sum_{i=1}^{r_j} \lambda_{ij} v_{ij}^2, j = 0, \dots, m \enspace . \tag{2.5.2}$$

Unfortunately from a practical point of view, the solution $(\mu_k, \sigma_0, \dots, \sigma_m)$ satisfies approximately the equality constraint in $(Q_k)^*$:

$$f_{\mathrm{pop}}(\mathbf{x}) - \mu_k \simeq \sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x}) \enspace .$$

Then, let us note $\theta_k := \| f_{\mathrm{pop}}(\mathbf{x}) - \mu_k - \sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x}) \|$ the error for the problem $(Q_k)^*$. The method of Kaltofen et al. [KLYZ12] consists in applying first Gauss-Newton iterations to refine the approximate SOS certificate, until θ_k is less than a given tolerance and then, to apply the algorithm of [PP08]. The number μ_k is approximated by a nearby rational number ${\mu_k}^{\mathbb{Q}} \lessapprox \mu_k$ and the approximate SOS certificate $(\sigma_0, \dots, \sigma_m)$ is converted to a rational SOS (for more details, see [PP08]). Then the refined SOS is projected orthogonally to to the set of rational SOS certificates $({\mu_k}^{\mathbb{Q}}, {\sigma_0}^{\mathbb{Q}}, \dots, {\sigma_m}^{\mathbb{Q}})$, which satisfy (exactly) the equality constraint in $(Q_k)^*$. This can be done by solving a least squares problem, see [PP08] for more information.

In our case, we do not use the rounding and projection algorithm of Peyrl and Parrilo. Instead, we perform a rational SOS extraction, following the procedure depicted in Figure 2.5 (this is an implementation feature of our solver `NLCertify`).

The procedure relies on the LACAML (Linear Algebra with OCAML) library, which implements the BLAS/LAPACK-interface and the ZARITH OCAML library, which implements arithmetic and logical operations over arbitrary-precision integers. Eigenvalues and eigenvectors are computed with the `syev` routine of LACAML (Line 2), then converted into arbitrary precision rationals using the function `Q.of_float` of ZARITH (Line 3). The floating-point SDP optimal value μ_k is also converted into a rational $\tilde{\mu_k}$.

```
Input: Objective polynomial function $f_{\text{pop}}$, polynomial constraints $g_1, \dots, g_m$, SDP relaxation order $k$
Output: Rational bound $\tilde{\mu_k}$, rational SOS certificates $\tilde{\sigma_0}, \dots, \tilde{\sigma_m}$, polynomial remainder $\epsilon_{\text{pop}}$, polynomial remainder lower bound $\epsilon^*_{\text{pop}}$
1: Extract a numerical solution $\mu_k, Z_0, \dots, Z_m$ returned by the SDP solver after solving $Q_k$
2: $\sigma_0, \dots, \sigma_m :=$ from_syev$(Z_0, \dots, Z_m)$
3: $\tilde{\mu_k}, \tilde{\sigma_0}, \dots, \tilde{\sigma_m} :=$ float_to_rat$(\mu_k, \sigma_0, \dots, \sigma_m)$
4: $\epsilon_{\text{pop}}(\mathbf{x}) := f_{\text{pop}}(\mathbf{x}) - \tilde{\mu_k} - \sum_{j=0}^m \tilde{\sigma_j}(\mathbf{x}) g_j(\mathbf{x})$
5: $\epsilon^*_{\text{pop}} := \sum_{\epsilon_{\boldsymbol{\alpha}} \leqslant 0} \epsilon_{\boldsymbol{\alpha}}$
6: return $\tilde{\mu_k}, \tilde{\sigma_0}, \dots, \tilde{\sigma_m}, \epsilon_{\text{pop}}, \epsilon^*_{\text{pop}}$
```

Figure 2.5: `extract_sos`

The next step is simpler than the projection algorithm of Parrilo. We consider the following polynomial (Line 4):

$$\epsilon_{\text{pop}}(\mathbf{x}) := f_{\text{pop}}(\mathbf{x}) - \tilde{\mu_k} - \sum_{j=0}^{m} \tilde{\sigma_j}(\mathbf{x}) g_j(\mathbf{x}) \ .$$

Remark 2.15. The SOS numerical certificate is first extracted from the SDP solvers output data and represented with OCaml floating points polynomials. The coefficients of the SOS certificate polynomials $(\tilde{\sigma_0}, \dots, \tilde{\sigma_m})$ (as well as ϵ_{pop}) lie in $\mathbb{Q}$ after conversion in our implementation. Notice also the relation between the error considered in the rounding algorithm of Parrilo and Peyrl and the polynomial ϵ_{pop}:

$$\theta_k := \|\epsilon_{\text{pop}}\| \ .$$

A lower bound ϵ^*_{pop} of $\inf_{\mathbf{x} \in K_{\text{pop}}} \epsilon_{\text{pop}}(\mathbf{x})$ can be obtained using certified interval arithmetic. Then $\tilde{\mu_k} + \epsilon^*_{\text{pop}}$ is a valid lower bound of f_{pop} on K_{pop}.

One has:

$$\epsilon_{\text{pop}}(\mathbf{x}) := \sum_{\boldsymbol{\alpha} \in \mathbb{N}_{2k}^{n_{\text{pop}}}} \epsilon_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} \ .$$

Then, after normalization and indexing the box inequalities as in 2.2.6, the following holds for each $\mathbf{x} \in [0,1]^n$:

$$\epsilon_{\text{pop}}(\mathbf{x}) \geqslant \sum_{\epsilon_{\boldsymbol{\alpha}} \leqslant 0} \epsilon_{\boldsymbol{\alpha}} \ . \tag{2.5.3}$$

Hence, this procedure allows to compute directly a lower bound ϵ^*_{pop} (Line 5) from the right-hand side of (2.5.3).

Chapter 3

A General Approximation Scheme for Global Optimization

This chapter is devoted to the general framework to solve nonlinear inequalities involving transcendental functions. We first describe the abstract syntax representation of these functions (Section 3.1). Given some approximation tools for univariate or semialgebraic functions, we obtain a hierarchy of semialgebraic estimators, which we bound using SOS relaxations (Section 3.2). We prove the consistency of this approximation scheme under certain assumptions (Section 3.3).

3.1 Abstract Syntax Tree of Multivariate Transcendental Functions

Let $f \in \langle \mathcal{D} \rangle^{\text{sa}}$ be a function and K a box issued from a Flyspeck inequality or a general global optimization problem described in 1.1.2. We assimilate the objective function f with its abstract syntax tree t. We assume that the leaves of t are semialgebraic functions in the set $\mathcal{A}$ and other nodes are univariate transcendental functions (arctan, *etc*) or basic operations $(+, \times, -, /)$. For the sake of the simplicity, we suppose that each univariate transcendental function is monotonic.

Example 3.1. Continuing Example 1.3, we represent the function f with the syntax abstract tree depicted in Figure 3.1.

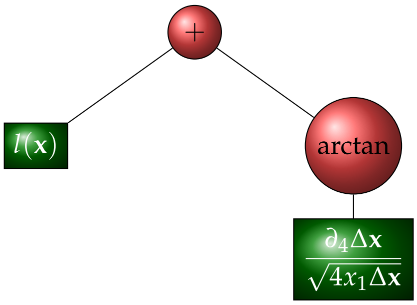

$+$
$l(\mathbf{x})$
arctan
$\frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}}$

Figure 3.1: The syntax abstract tree of the function f defined in Example 1.3

3.2 Combining Semialgebraic Approximations and Sums of Squares

Here we consider an instance of Problem (1.1.3) and explain how to combine semialgebraic optimization techniques with approximation tools for univariate or semialgebraic functions. The auxiliary algorithm `approx` is presented in Figure 3.2.

We identify the objective function f with its abstract syntax tree t. Given an abstract syntax tree t, a semialgebraic set K, an SOS relaxation order k and a precision p which can be either a finite sequence of control points $\mathbf{x}_1, \dots, \mathbf{x}_p \in K$ or a polynomial approximation degree d, the algorithm `approx` computes a lower bound m (resp. upper bound M) of t over K and an underestimator t^- (resp. an overestimator t^+) of t by means of semialgebraic functions.

It relies on an approximation method `unary_approx` for univariate (possibly transcendental) functions and an approximation method `reduce_lift` for semialgebraic functions. The algorithms presented in Part II use particular instances of the procedures `unary_approx` and `reduce_lift`.

The simplest way to define `reduce_lift` is to consider the identity over semialgebraic functions (see Section 4.2 and 5.3). Semialgebraic functions can also be approximated by polynomials (see Section 6.3).

We shall need to consider various approximation schemes, including the approximation of univariate functions by polynomials of increasing degrees, or maxplus approximations in which the precision is determined by certain sets of control points. For some other schemes, the precision may be controlled by the fineness of a mesh. A convenient way to express the refinement of the precision, for general schemes, is to use the vocabulary of nets.

We recall the following definitions, using [Nag74]:

Definition 3.2. A directed set is a set D with a relation $\leqslant$ which is reflexive, transitive and directed, *i.e.* for each $a, b \in D$, there exists some $c \in D$ such that $a \leqslant c$ and $b \leqslant c$.

Definition 3.3. A net in a set X is a map $\lambda : D \to X$. If X is a topological space, we say that the net λ converges to $x \in X$ and write $\lambda \to x$ if and only if for every neighbourhood U of x, there exists some $d \in D$ and some tail $\Lambda_d := \{\lambda(c) : d \leqslant c \in D\}$ such that $\Lambda_d \subseteq U$.

A classical way to approximate an univariate function on a given interval I is to use best uniform polynomials. In this case, the precision parameter is the approximation degree d and `unary_approx` calls the Remez algorithm (Figure 4.1 in Chapter 4.1). The sequence of approximation degrees defines the net.

An alternative approach is to build maxplus estimators from the control points sequence s (Chapter 5). We shall see that for maxplus approximations, the net is the set of finite subsets of I.

More generally, we represent the precision p by an element of a directed set $\mathcal{P}$. For the sequel, we need to make the following assumption on `unary_approx` and `reduce_lift`:

Assumption 3.4. *For every function r of the dictionary $\mathcal{D}$, defined on a closed interval I, the procedure* `unary_approx` *returns two nets of univariate lower semialgebraic estimators $(r_p^-)_{p \in \mathcal{P}}$ and upper semialgebraic estimators $(r_p^+)_{p \in \mathcal{P}}$, that uniformly converge to r on I.*

```
Input: abstract syntax tree $t$, semialgebraic set $K$, SOS relaxation order $k$, precision $p$
Output: lower bound $m$, upper bound $M$, lower semialgebraic estimator $t^-$, upper
    semialgebraic estimator $t^+$
1: if $t \in \mathcal{A}$ then
2:     $t^- := t, t^+ := t$
3: else if $r := \texttt{root}(t)$ with child $c$ then
4:     $m_c, M_c, c^-, c^+ := \texttt{approx}(c, K, k, p)$
5:     $I := [m_c, M_c]$
6:     $r^-, r^+ := \texttt{unary\_approx}(r, I, c, p)$
7:     $t^-, t^+ := \texttt{compose\_approx}(r, r^-, r^+, I, c^-, c^+)$
8: else if $\texttt{bop} := \texttt{root}(t)$ is a binary operation with children $c_1$ and $c_2$ then
9:     $m_{c_i}, M_{c_i}, c_i^-, c_i^+ := \texttt{approx}(c_i, K, k, p)$ for $i \in \{1, 2\}$
10:    $I_2 := [m_{c_2}, M_{c_2}]$
11:    $t^-, t^+ := \texttt{compose\_bop}(c_1^-, c_1^+, c_2^-, c_2^+, \texttt{bop}, I_2)$
12: end
13: $t^- := \texttt{reduce\_lift}(t^-, K, k, p), t^+ := \texttt{reduce\_lift}(t^+, K, k, p)$
14: $\mathbf{x}^- := \texttt{vars}(t^-, K), \mathbf{x}^+ := \texttt{vars}(t^+, K)$
15: return $\texttt{min\_sa}(t^-, K, \mathbf{x}^-, k), \texttt{max\_sa}(t^+, K, \mathbf{x}^+, k), t^-, t^+$
```

Figure 3.2: `approx`: General Semialgebraic Approximation Algorithm

```
Input: univariate function $r$, lower estimator $r^-$, upper estimator $r^+$, interval $I$, lower
    estimator $c^-$, upper estimator $c^+$
Output: lower estimator $t^-$, upper estimator $t^+$
if $r$ is increasing on $I$ then
    $t^- := r^- \circ c^-, t^+ := r^+ \circ c^+$
else if $r$ is decreasing on $I$ then
    $t^- := r^- \circ c^+, t^+ := r^+ \circ c^-$
end
return $t^-, t^+$
```

Figure 3.3: `compose_approx` : Estimators Composition Algorithm

For every semialgebraic function $f_{sa} \in \mathcal{A}$, defined on a compact semialgebraic set K, the procedure `reduce_lift` *returns two nets of lower semialgebraic estimators $(t_p^-)_{p \in \mathcal{P}}$ and upper semialgebraic estimators $(t_p^+)_{p \in \mathcal{P}}$, that uniformly converge to f_{sa} on K.*

The general approximation algorithm `approx` is defined by induction on the abstract syntax tree t.

When t is reduced to a leaf (Line 2), *i.e.* it represents a semialgebraic function of $\mathcal{A}$, we call the functions `min_sa` and `max_sa`, which determine lower and upper bounds using techniques presented in Section 2.4 (see the algorithm presented in Figure 2.1). In this case, the tree t provides an exact semialgebraic estimator.

If the root of t is an univariate function node r taking a single child c as argument, lower and upper bounds m_c and M_c are recursively obtained (Line 4), as well as estimators c^- and c^+. Then we define $I := [m_c, M_c]$ and apply the function `unary_approx` to get estimators r^- and r^+ of r over I. These estimators can be parametrized either by the given control points $\mathbf{x}_1, \ldots, \mathbf{x}_p$ (see e.g. Figure 5.3) or a polynomial approximation degree d and are composed with c^- and c^+ (so-called `compose_approx` function)

```
Input: lower estimator $c_1^-$, upper estimator $c_1^+$, lower estimator $c_2^-$, upper estimator
    $c_2^+$, binary operation bop, interval $I_2$ (optional parameter for the division case)
Output: lower estimator $t^-$, upper estimator $t^+$
 1: if bop $= +$ then
 2:     $t^- := c_1^- + c_2^-$, $t^+ := c_1^+ + c_2^+$
 3: else if bop $= -$ then
 4:     $t^- := c_1^- - c_2^+$, $t^+ := c_1^+ - c_2^-$
 5: else if bop $= \times$ then
 6:     $t^- := \min\{c_1^- c_2^-, c_1^+ c_2^-, c_1^- c_2^+, c_1^+ c_2^+\}$
 7:     $t^+ := \max\{c_1^- c_2^-, c_1^+ c_2^-, c_1^- c_2^+, c_1^+ c_2^+\}$
 8: else if bop $= /$ then
 9:     if $0 \in I_2$ then return Division Exception
10:     end
11:     $t^- := \inf\{c_1^- / c_2^-, c_1^+ / c_2^-, c_1^- / c_2^+, c_1^+ / c_2^+\}$
12:     $t^+ := \sup\{c_1^- / c_2^-, c_1^+ / c_2^-, c_1^- / c_2^+, c_1^+ / c_2^+\}$
13: end
14: return $t^-$, $t^+$
```

Figure 3.4: `compose_bop` : Semialgebraic Arithmetic Algorithm

to obtain an underestimator t^- as well as an overestimator t^+. The `compose_approx` function is depicted in Figure 3.3. The composition depends on the monotonicity properties of r (from Line 1 to Line 4). For instance, if r is increasing, an underestimator of t is obtained by composing the underestimator r^- of r with the underestimator c^- of c.

Remark 3.5. When the root of t is the power function $(\cdot)^{1/p}$ ($p \in \mathbb{N}_0$), taking a single child c as argument, the estimators of t are obtained by composition of this power function approximation with the estimators of c. In practice, we raise an exception if the lower bound of the child argument (m_c) is negative. This exception can be handled by increasing either the SOS relaxation order k or the degree d of the best uniform approximation polynomial.

The last case occurs when the root of t is a binary operation whose arguments are two children c_1 and c_2. We can apply recursively `approx` to each child and get semialgebraic underestimators c_1^-, c_2^- and overestimators c_1^+, c_2^+. The `compose_bop` algorithm, depicted in Figure 3.4, presents the different operations between semialgebraic estimators. We emphasize that the semialgebraic arithmetic rules are analogous with the interval arithmetic operations. For instance, when the binary operation is the multiplication (or the division), one has to consider the four possible products (Line 5) between an estimator of c_1 and an estimator of c_2: $c_1^- c_2^-, c_1^+ c_2^-, c_1^- c_2^+$ and $c_1^+ c_2^+$. We obtain a valid underestimator (resp. overestimator) by taking the minimum (resp. maximum) of the four possible products. When the binary operation is the division, we raise an exception if the sign of each of the estimators of c_2 is not constant (Line 9). This exception can be handled by getting more accurate bounds with either the semialgebraic optimization function `min_sa` (with a higher order k) or the approximation function `reduce_lift`.

Finally, we obtain a semialgebraic underestimator t^- of t^- (resp. an overestimator t^+ of t^+) with the algorithm `reduce_lift`. We get the list of variables of the semialgebraic optimization problems using the auxiliary `vars` algorithm (Line 14). Then,

```
Input: abstract syntax tree $t$, semialgebraic set $K$, $iter_{\max}$ (optional argument), precision $p$
Output: lower bound $m$
 1: $s := [$argmin(randeval($t$))$]$                         ▷ $s \in K$
 2: $m := -\infty$
 3: $iter := 0$
 4: while $iter \leqslant iter_{\max}$ do
 5:     Choose an SDP relaxation order $k \geqslant k_0$
 6:     $m, M, t^-, t^+ :=$ approx($t, K, k, p$)
 7:     $\mathbf{x}_{opt} :=$ guess_argmin($t^-$)                ▷ $t^-(\mathbf{x}_{opt}) \simeq m$
 8:     $s := s \cup \{\mathbf{x}_{opt}\}$
 9:     $p :=$ update_precision($p$)
10:     $iter := iter + 1$
11: done
12: return $m$, $\mathbf{x}_{opt}$
```

Figure 3.5: `optim` : General Semialgebraic Optimization Algorithm

we call the functions `min_sa` and `max_sa` which determine lower and upper bounds using techniques presented in Section 2.4 (see the algorithm presented in Figure 2.1).

Remark 3.6. The implementation of the multiplication can be improved by inspecting the sign of the estimators. Suppose that c_1 has a constant sign. We can assume that c_1 is positive without loss of generality (this can be inferred from the nonnegativity of m_{c_1}). Then, it follows that:

$$\inf\{c_1^- c_2^-, c_1^+ c_2^-\} \leqslant c_1 c_2^- \leqslant c_1 c_2 \leqslant c_1 c_2^+ \leqslant \sup\{c_1^- c_2^+, c_1^+ c_2^+\}.$$

Thus, one has to consider semialgebraic estimators that involve either infimum or supremum of two products (instead of four when no sign information is available). We have observed that in practice, all the inequalities that we consider in the Flyspeck project satisfy this restriction.

Let $c_1, \dots, c_s$ be the components of the tree t, on which one calls approximation algorithms with respective precisions $p_1 \in \mathcal{P}_1, \dots, p_s \in \mathcal{P}_s$. Let $\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_s$ be the set of precisions, ordered with the product order. The global precision parameter $p \in \mathcal{P}$ can be updated with the `update_precision` procedure.

Now we describe our main semialgebraic optimization algorithm `optim` (see Figure 3.5). Given an abstract syntax tree t and a compact semialgebraic set K this algorithm returns a lower bound m of t using semialgebraic minimax estimators computed recursively with `approx`. The relaxation order k (Line 5) is a parameter of the semialgebraic optimization functions `min_sa` (as well as `max_sa`) and `reduce_lift`. Let suppose that K is described by polynomial inequalities $g_1(\mathbf{x}) \geqslant 0, \dots, g_m(\mathbf{x}) \geqslant 0$.

The semidefinite relaxation order must be at least $k_0 := \max_{1 \leqslant j \leqslant m}\{\lceil \deg(g_j)/2 \rceil)\}$. In practice, we solve semialgebraic optimization problems with the second or third SDP Lasserre's relaxation and take $k = k_0$. At the beginning, the set of control points consists of a single point of the box K. This point is chosen so that it minimizes the value of the function associated to the tree t among a set of random points (Line 1). Then, at each iteration of the loop from Lines 4 to 11, the auxiliary function `approx` is called to compute a lower bound m of the function t (Line 6), using the estimators t^- and t^+. At Line 7, a minimizer candidate $\mathbf{x}_{opt}$ of the underestimator tree t^- is computed. It is obtained by projecting a solution $\mathbf{x}_{sdp}$ of the SDP relaxation of Section 2.4 on the coordinates representing the first order moments, following [Las01, Theorem 4.2]. However, the projection may not belong to K when the relaxation order k is not large enough. This is why tools like SPARSEPOP use local optimization solver in a post-processing step to provide a point in K which may not be a global minimizer. In any case, $\mathbf{x}_{opt}$ is then added to the set of control points (Line 8). Alternatively, if we are only interested in determining whether the infimum of t over K is nonnegative (Problem (1.1.3)), the loop can be stopped as soon as $m \geqslant 0$.

3.3 Convergence of the General Semialgebraic Approximation Algorithm

Under certain assumptions and given an accuracy $\epsilon > 0$, we prove that the objective function f can be uniformly ϵ-approximated over the semialgebraic set K with the algorithm `approx`. Let the relaxation order k be fixed and t_p^- (resp. t_p^+) be the underestimator (resp. overestimator) of t on K obtained with the `approx` function at precision p.

The limit of a net indexed by $p \in \mathcal{P}$ is obtained by increasing the precision of each elementary approximation algorithms applied to $c_1, \dots, c_s$.

To prove this proposition, we recall the definition of the modulus of continuity.

Definition 3.7 (Modulus of continuity)**.** Let u be a function defined on an interval I. The modulus of continuity is defined as:

$$\omega(\delta) := \sup_{\substack{x_1, x_2 \in I \\ |x_1 - x_2| < \delta}} \mid u(x_1) - u(x_2) \mid$$

Proposition 3.8 (Convergence of the general semialgebraic approximation algorithm)**.** *Under Assumption (3.4), the nets $(t_p^-)_p$ and $(t_p^+)_p$ uniformly converge to t on K.*

Proof. By induction on the structure of t.

- When t represents a semialgebraic function of $\mathcal{A}$, the underestimator (resp. overestimator) net $(t_p^-)_p$ (resp. $(t_p^+)_p$) converges uniformly to t by the assumption made on `reduce_lift` in 3.4.

- The second case occurs when the root of t is an univariate function $r \in \mathcal{U}$ with the single child c. Suppose that r is increasing without loss of generality. We consider the net of underestimators $(c_p^-)_p$ (resp. overestimators $(c_p^+)_p$) as well as lower and upper bounds m_{c_p} and M_{c_p} which are obtained recursively. Since K is a compact semialgebraic set, one can always find an interval I_0 enclosing the values of r_p^+ (*i.e.* such that $[m_{c_p}, M_{c_p}] \subset I_0$), for all p.

  The induction hypothesis is the uniform convergence of $(c_p^-)_p$ (resp. $(c_p^+)_p$) to c on K. Now, we prove the uniform convergence of $(t_p^+)_p$ to t on K. One has:

  $$\|t - t_p^+\|_\infty \leqslant \|r \circ c - r_p^+ \circ c\|_\infty + \|r_p^+ \circ c - t_p^+\|_\infty. \tag{3.3.1}$$

  Let note ω the modulus of continuity of r_p^+ on I_0. Thus, the following holds:

$$\|r_p^+ \circ c - r_p^+ \circ c_p^+\|_\infty \leqslant \omega(\|c - c_p^+\|_\infty). \tag{3.3.2}$$

Let $\epsilon > 0$ be given. The univariate function r_p^+ is uniformly continuous on I_0, thus there exists $\delta > 0$ such that:

$$\omega(\delta) \leqslant \epsilon/2. \tag{3.3.3}$$

Let choose such a δ. By induction hypothesis, there exists a precision p_0 such that for all $p \geqslant p_0$, $\|c - c_p^+\|_\infty \leqslant \delta$. Hence, using (3.3.2), the following holds:

$$\|r_p^+ \circ c - r_p^+ \circ c_p^+\|_\infty \leqslant \epsilon/2. \tag{3.3.4}$$

Moreover, from the uniform convergence of $(r_p^+)_{p\in\mathbb{N}}$ to r on K, there exists a precision p_1 such that for all $p \geqslant p_1$:

$$\|r \circ c - r_p^+ \circ c\|_\infty \leqslant \epsilon/2. \tag{3.3.5}$$

Using (3.3.1) together with (3.3.4) and (3.3.5) yield the desired result. The proof of the uniform convergence of the underestimators is analogous.

- If the root of t is a binary operation whose arguments are two children c_1 and c_2, then by induction hypothesis, we obtain semialgebraic estimators $c_{1,p}^-$, $c_{2,p}^-$, $c_{1,p}^+$, $c_{2,p}^+$ that verify:

$$\lim_{p\to\infty} \|c_1 - c_{1,p}^-\|_\infty = 0, \qquad \lim_{p\to\infty} \|c_1 - c_{1,p}^+\|_\infty = 0, \tag{3.3.6}$$

$$\lim_{p\to\infty} \|c_2 - c_{2,p}^-\|_\infty = 0, \qquad \lim_{p\to\infty} \|c_2 - c_{2,p}^+\|_\infty = 0. \tag{3.3.7}$$

If $\texttt{bop} = +$, by using the triangle inequality:

$$\|c_1 + c_2 - c_{1,p}^- - c_{2,p}^-\|_\infty \leqslant \|c_1 - c_{1,p}^-\|_\infty + \|c_2 - c_{2,p}^-\|_\infty,$$
$$\|c_1 + c_2 - c_{1,p}^+ - c_{2,p}^+\|_\infty \leqslant \|c_1 - c_{1,p}^+\|_\infty + \|c_2 - c_{2,p}^+\|_\infty.$$

Then, the uniform convergence comes from (3.3.6) and (3.3.7). The proof for the other cases is analogous.

□

For a precision p, define m_p^* to be the optimal value of the underestimator t_p^- on K:

$$m_p^* := \inf_{\mathbf{x}\in K} t_p^- .$$

Notice that if we suppose that the SOS Assumption (2.2) holds, then we can theoretically obtain the optimal value m_p^* of the semialgebraic underestimator t_p^-, using the semialgebraic optimization techniques described in Section 2.4.

Corollary 3.9 (Convergence of the estimators optimal values). *Suppose that Assumption (3.4) holds. Then, the net $(m_p^*)_p$ converges to the optimal value t^* of t.*

Proof. Let $\mathbf{x}_p^*$ be a minimizer of t_p^- on K and note $\mathbf{x}^*$ one minimizer of t on K, then one has:

$$t(\mathbf{x}^*) = t^*, \quad t_p^-(\mathbf{x}_p^*) = m_p^*.$$

By definition, the following inequalities hold:

$$t_p^-(\mathbf{x}_p^*) \leqslant t_p^-(\mathbf{x}^*) \leqslant t(\mathbf{x}^*) \leqslant t(\mathbf{x}_p^*). \tag{3.3.8}$$

From (3.3.8), one has:

$$0 \leqslant t(\mathbf{x}^*) - t_p^-(\mathbf{x}_p^*). \tag{3.3.9}$$

Let $\epsilon > 0$ be given. From Proposition 3.8, there exists a precision d_0 such that for all $d \geqslant d_0$, one has:

$$t(\mathbf{x}^*) - t_p^-(\mathbf{x}^*) < \epsilon/2, \tag{3.3.10}$$

$$t(\mathbf{x}_p^*) - t_p^-(\mathbf{x}_p^*) < \epsilon/2. \tag{3.3.11}$$

From (3.3.8), $t_p^-(\mathbf{x}^*) \leqslant t(\mathbf{x}_p^*)$, thus one has the following:

$$t(\mathbf{x}^*) - t_p^-(\mathbf{x}_p^*) \leqslant [t(\mathbf{x}^*) - t_p^-(\mathbf{x}^*)] + [t(\mathbf{x}_p^*) - t_p^-(\mathbf{x}_p^*)].$$

Then, as a consequence of (3.3.10) and (3.3.11), one has:

$$t(\mathbf{x}^*) - t_p^-(\mathbf{x}_p^*) < \epsilon. \tag{3.3.12}$$

The inequalities from (3.3.9) and (3.3.12) yield the desired result. □

To study the convergence of the minimizers of t_p^-, we first introduce some background on the Γ-convergence (we refer the reader to [Mas93] for more details) and the lower semicontinuous envelope.

The topology of Γ-Convergence is known to be metrizable hence, we shall consider the Γ-Convergence of sequences (rather than nets).

Definition 3.10 (Γ-Convergence)**.** The sequence $(t_p)_{p\in\mathbb{N}}$ Γ-converges to t if the following two conditions hold:

1. (asymptotic common lower bound) for all $\mathbf{x} \in K$ and every $(\mathbf{x}_p)_{p\in\mathbb{N}}$ such that $\lim_{p\to\infty} \mathbf{x}_p = \mathbf{x}$,
$$t(\mathbf{x}) \leqslant \liminf_{p\to\infty} t_p(\mathbf{x}_p).$$

2. (existence of recovery sequences) for all $\mathbf{x} \in K$, there exists some $(\mathbf{x}_p)_{p\in\mathbb{N}}$ such that $\lim_{p\to\infty} \mathbf{x}_p = \mathbf{x}$ and
$$\limsup_{p\to\infty} t_p(\mathbf{x}_p) \geqslant t(\mathbf{x}).$$

Define $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$ to be the extended real number line.

Definition 3.11 (Lower Semicontinuous Envelope)**.** Given $t : K \mapsto \overline{\mathbb{R}}$, the lower semicontinuous envelope of t is defined by:

$$t^{\mathrm{lsc}}(\mathbf{x}) := \sup\{g(\mathbf{x}) \mid g : K \mapsto \overline{\mathbb{R}} \text{ is lower semicontinuous and } g \leqslant f \text{ on } K\}.$$

If t is continuous, then $t^{\mathrm{lsc}} := t$.

Theorem 3.12 (Fundamental Theorem of Γ-Convergence [Mas93]). *Suppose that the sequence $(t_p)_{p \in \mathbb{N}}$ Γ-converges to t and $\mathbf{x}_p$ minimizes t_p. Then every limit point of the sequence $(\mathbf{x}_p)_{p \in \mathbb{N}}$ is a global minimizer of t.*

Theorem 3.13 (Γ and Uniform Convergence [Mas93]). *If $(t_p)_{p \in \mathbb{N}}$ uniformly converges to t, then $(t_p)_{p \in \mathbb{N}}$ Γ-converges to t^{lsc}.*

Theorem 3.13 also holds for nets, since the topology of Γ-Convergence is metrizable.

The following corollary describes the correspondence between the minimizers of the underestimators net $(t_p^-)_p$ and the global minimizers of t.

Corollary 3.14 (Convergence of the Minimizers Net). *Suppose that Assumption (3.4) holds. Then, every limit point of the net of minimizers $(\mathbf{x}_p^*)_{p \in \mathbb{N}}$ is a global minimizer of t over K.*

Proof. From Proposition 3.8, the underestimators net $(t_p^-)_{p \in \mathbb{N}}$ uniformly converge to t on K. Then, by using Theorem 3.13, the net $(t_p^-)_{p \in \mathbb{N}}$ Γ-converges to $t^{\mathrm{lsc}} := t$ (by continuity of t). It follows from the fundamental Theorem of Γ-Convergence 3.12 that every limit point of the net of minimizers $(\mathbf{x}_p^*)_{p \in \mathbb{N}}$ is a global minimizer of t over K. □

Part II

Nonlinear Optimization via Maxplus Templates

Chapter 4

Minimax Semialgebraic Optimization

In this chapter, we present a method which combines minimax estimators and semialgebraic optimization. We first focus on the approximation of univariate functions using best uniform polynomials (Section 4.1). It leads to an approximation algorithm (Section 4.2), parametrized by the degrees of polynomial estimators. Numerical results are presented in Section 4.3. The convergence of the approximation algorithm is a consequence of the uniform convergence property of the minimax polynomials (Section 4.4). We conclude this chapter by presenting a natural extension of this method using Taylor polynomials (Section 4.5).

In the sequel, we consider an instance of Problem (1.1.3) and we assume that K is a box. We suppose that the univariate functions involved in the objective function f are differentiable.

4.1 Minimax Polynomials for Univariate Functions

We first recall the principles of the best uniform polynomial approximation algorithm for a transcendental univariate function u on a given interval $I := [m, M] \subset \mathbb{R}$, with $m < M$. The infinite norm of u (also called Chebyshev norm or sup norm) is $\|u\|_\infty := \sup_{x \in I} |u(x)|$.

For the following definition, we use [MH03, Definition 3.2]:

Definition 4.1 (Best Uniform Polynomial Approximation). An approximation $f_d \in \mathbb{R}_d[x]$ is said to be best, if the following holds for any other approximation $p \in \mathbb{R}_d[x]$:

$$\|u - f_d\|_\infty \leqslant \|u - p\|_\infty \ .$$

This best approximation f_d exists and is unique if u is continuous on I. The best uniform degree-d polynomial approximation (or minimax polynomial) solves the following optimization problem:

$$\min_{p \in \mathbb{R}_d[x]} \|u - p\|_\infty = \min_{p \in \mathbb{R}_d[x]} (\sup_{x \in I} |u(x) - p(x)|) \ .$$

The degree-d minimax polynomial f_d can be obtained through an iterative algorithm designed by Remez. The algorithm computes a sequence of polynomials

$f_d^{(1)}, \dots, f_d^{(k)}$ until the approximation error $\|u - f_d^{(k)}\|_\infty$ becomes closed enough to the optimal value of the previous optimization problem. We refer the reader to [Che09] for more details about the Remez algorithm implementations.

In our approximation algorithm (see Figure 4.1), we use the function `remez` available in the `Sollya` tool [CJL10]. The parameters of `remez` are the univariate function u, the degree-d of the minimax polynomial estimator and the closed interval $I \subset \mathbb{R}$ where u must be approximated.

If the algorithm converges and returns a degree-d polynomial f_d, then a numerical approximation of the infinity norm of the error function $(u - f_d)$ on the interval I can be obtained with the so-called `sollyainfnorm` function. This function can call either the `infnorm` or `dirtyinfnorm` routine from `Sollya` (see remark 4.2). Then, if the numerical result of the function is ϵ_d, let consider the functions u^- and u^+ defined as follows:

$$u^- : x \mapsto f_d(x) - \epsilon_d, \quad u^+ : x \mapsto f_d(x) + \epsilon_d \ . \tag{4.1.1}$$

It is straightforward to prove that u^- (resp. $u+$) is a degree-d underestimator (resp. overestimator) of u on the interval I.

Remark 4.2. The `dirtyinfnorm` procedure is based on an algorithm which detects where the derivative of $(u - f_d)$ changes its sign for consecutive points in the interval I. Even though the returned result bound of `dirtyinfnorm` is generally accurate, more rigorous supremum norms can be used as an alternative for `sollyainfnorm`. For instance, the function `infnorm` is based on interval arithmetic combined with Taylor recursions, L'Hopital recursions as well as bisection techniques [CJL10]. Another algorithm using automatic differentiation is described in [CJL09].

Example 4.3. Continuing Example 1.3, we consider the function f, defined on the box $K := [4, 6.3504]^3 \times [6.3504, 8] \times [4, 6.3504]^2$:

$$f(\mathbf{x}) := l(\mathbf{x}) + \arctan \frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}} \ .$$

In Example 2.12, we already computed lower bounds for the argument of the arctan function. We obtained $m_2 := -0.618$ and $m_3 := -0.445$. Similarly, we can obtain upper bounds $M_2 := 0.891$ and $M_3 := 0.874$. Hence, we apply the `remez` function of the `Sollya` tool on the arctan function either on the interval $[m_2, M_2]$ (SDP relaxation Q_2^{sa}) or $[m_3, M_3]$ (SDP relaxation Q_3^{sa}). On the other hand, the function l involves a linear combination of square roots of the components of $\mathbf{x}$. Thus, we apply `remez` on the square root function either on the interval $[4, 6.3504]$ or $[6.3504, 8]$.

In Table 4.1, we present the corresponding test results for these two univariate functions involved in the Flyspeck Lemma$_{9922699028}$. The integer d represents the degree of the minimax polynomial used to approximate the function u on the interval I. For each approximation, an upper bound of the approximation error ϵ_d is also given.

First, consider the approximation error ϵ_d on the arctan function over the interval $[m_2, M_2]$ obtained at a low SDP relaxation order. When the minimax degree-d increases, the value of ϵ_d decreases, as expected. Now if we fix the degree-d, we notice that the error also decreases when the interval is tighter.

Then, consider the approximations of the square root function. On the interval $[6.3504, 8]$, a quadratic minimax polynomial already provides a good approximation since the upper bound on ϵ_2 is less than 10^{-7}. On the other hand, the polynomial

Table 4.1: Upper Bounds of Minimax Approximation Errors for the Univariate Functions involved in Lemma$_{9922699028}$ Flyspeck

u	I	d	Upper bound of ϵ_d
arctan	$[-0.618, 0.891]$	4	1.47×10^{-3}
		5	3.08×10^{-4}
		6	1.03×10^{-4}
	$[-0.445, 0.874]$	4	6.34×10^{-4}
		5	2.00×10^{-4}
		6	2.64×10^{-5}
$\sqrt{}$	$[4, 6.3504]$	2	4.31×10^{-5}
		3	3.10×10^{-5}
		4	2.50×10^{-5}
	$[6.3504, 8]$	2	9.34×10^{-8}

approximation on the interval $[4, 6.3504]$ is less accurate and increasing the degree does not significantly improve the upper bounds of the error.

4.2 Combining Minimax Approximations and Sums of Squares

The approximation algorithm minimax_approx is obtained from the recursive algorithm approx (see Figure 3.2) by taking the identity function for reduce_lift and the minimax_unary_approx procedure for unary_approx.

The auxiliary algorithm minimax_unary_approx is presented in Figure 4.1. Given an univariate function r (which has a single child c), an approximation degree-d, the child c and a closed interval I, this algorithm returns an underestimator r^- as well as an overestimator r^+ of r. If the function r belongs to $\mathcal{U} \setminus \mathcal{D}$ (*i.e.* r is either the absolute value or a power function) and the child is a linear polynomial (the two conditions in Line 1), then r provides an exact estimator.

Otherwise, we apply the function remez that builds the best uniform polynomial approximation f_d of a given degree-d on the interval I (Line 4), by using the Remez algorithm on r, as explained above. Then, we use the sollyainfnorm function on the error function $(r - f_d)$, defined on I and we obtain an upper bound of the error induced by this approximation (Line 5). Note that minimax_unary_approx does not depend on the sequence of control points $s = (\mathbf{x}_1, \dots, \mathbf{x}_p)$.

Now we present the optimization algorithm minimax_optim, obtained from the general optim algorithm (depicted in Figure 3.5). Given an abstract syntax tree t and a compact semialgebraic set K this algorithm returns a lower bound m of t using semialgebraic minimax estimators computed recursively with approx = minimax_approx. The resulting procedure is depicted in Figure 4.2. Here, we set $iter = d_{\min}$ (resp. $iter_{\max} = d_{\max}$), which is the minimal (resp. maximal) degree of the minimax polynomial approximations. The condition $d_{\min} \leqslant k_0$ allows to obtain valid SOS relaxations, when we call the auxiliary functions min_sa and max_sa. The parameter $d_{\max}$ shall be selected after consideration of the computational power available since one may

```
Input: univariate function $r$, interval $I$, child $c$, polynomial approximation degree-$d$
Output: underestimator $r^-$, overestimator $r^+$
if $r \in \mathcal{U} \setminus \mathcal{D}$ and $c \in \mathbb{R}_1[x]$ then
    $r^- := r, r^+ := r$
else if $r \in \mathcal{U}$ then
    $f_d := \texttt{remez}(r, d, I)$
    $\epsilon_d := \texttt{sollyainfnorm}(r - f_d, I)$
    $r^- := f_d - \epsilon_d, r^+ := f_d + \epsilon_d$
end
return $r^-, r^+$
```

Figure 4.1: `minimax_unary_approx`: Minimax Approximation Algorithm

```
Input: abstract syntax tree $t$, semialgebraic set $K$, $d_{\min}$, $d_{\max}$
Output: lower bound $m$
 1: $s := [\texttt{argmin}(\texttt{randeval}(t))]$
 2: $m := -\infty$
 3: $d := d_{\min}$                                        ▷ $d_{\min} \leqslant k_0$
 4: while $d \leqslant d_{\max}$ do
 5:     Choose an SDP relaxation order $k \geqslant k_0$
 6:     $m, M, t^-, t^+ := \texttt{minimax\_approx}(t, K, k, s)$
 7:     $\mathbf{x}_{opt} := \texttt{guess\_argmin}(t^-)$
 8:     $s := s \cup \{\mathbf{x}_{opt}\}$
 9:     $d := d + 1$
10: done
11: return $m$, $\mathbf{x}_{opt}$
```

Figure 4.2: `minimax_optim` : Minimax Semialgebraic Optimization Algorithm

need to solve semidefinite programs involving at most $O(d_{\max}^n)$ variables with matrices of size $O(\lceil d_{\max}/2 \rceil^n)$. In practice, we often consider quartic ($d_{\max} = 4$) or sextic ($d_{\max} = 6$) minimax polynomial to approximate the univariate functions involved in t.

At step d, t is approximated by a semialgebraic function involving minimax polynomials of degree-d (Line 6).

4.3 Numerical Test Results

We now present some numerical test results by applying the semialgebraic minimax optimization method to examples from the global optimization literature, as well as inequalities from the Flyspeck project. Our tool is implemented in OCAML and interfaced with the `Sollya` tool. Experiments are performed on an Intel Core i5 CPU (2.40 GHz).

For each problem presented in Table 4.2, our aim is to certify a lower bound m of a function f on a box K. We build minimax approximations of degree at most $d_{\max}$ for the univariate transcendental functions involved in f. The semialgebraic optimization problems are solved at the SDP relaxation order k.

If the bound $m_{d_{\max}}$ obtained at the final iteration with degree-$d_{\max}$ minimax approximations is lower than m, then the relaxation gap $(t^* - m_{d_{\max}})$ is too high to certify the requested bound.

Then, we perform a domain subdivision in order to reduce this gap: we divide the maximal width interval of K in two halves to get two sub-boxes K_1 and K_2 such that $K = K_1 \cup K_2$. We repeat this subdivision procedure, by applying `minimax_optim` on a finite set of sub-boxes, until we succeed to certify that m is a lower bound of f. We denote by #boxes the total number of sub-boxes generated by the algorithm. In Table 4.2, the *time* column indicates the total informal verification time, *i.e.* we certify neither the minimax estimators nor the lower bounds with COQ (we refer the reader to Chapter 7 for more details about the formal computation of lower bounds). When the minimax approximations are composed with nonlinear polynomials, we use lifting variables to bound the degree of the resulting polynomial optimization problem. We illustrate this technique with the following Example 4.4.

Example 4.4. Let consider Problem *Hartman 3 (H3)*:

$$\min_{\mathbf{x} \in [0,1]^3} f(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2) \enspace .$$

Our strategy to solve *(H3)* consists in approximating the exp function with quartic minimax polynomials, then solving the resulting semialgebraic optimization problems with a low SDP relaxation order $k = 2$.

First, for all $(i = 1, \dots, 4)$, we compute the intervals I_i enclosing the values of the polynomials $-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2$. Then, for each $(i = 1, \dots, 4)$, we apply the Remez algorithm to obtain valid overestimators u_i^+ of the exponential function over the intervals I_i. We use $n_{\text{lifting}} = 4$ auxiliary variables $(z_1, \dots, z_4)$ to represent the four quadratic polynomials. Finally, we solve the following polynomial optimization problem:

$$\begin{cases} \min\limits_{\mathbf{x} \in [0,1]^3} & -\sum_{i=1}^{4} u_i^+(z_i) \\ \text{s.t.} & z_i = -\sum\limits_{j=1}^{3} a_{ij}(x_j - p_{ij})^2, i = 1, \dots, 4, \\ & z_1 \in I_1, \dots, z_4 \in I_4 \enspace . \end{cases}$$

All test problems with a small number of variables ($n < 6$) can be solved within a couple of minutes. For a given problem, when the minimax approximation degree is higher, the verification time may also increase, even though the number of branch and bound iterations becomes smaller. It comes from the fact that the number of SDP variables grows rapidly with the relaxation order.

Then, we discuss about the numerical performance of the method for medium-scale problems. For instance, consider *SWF* for $n = 10$. The quartic minimax approximation of the function $x \mapsto \sin\sqrt{x}$ over the interval $I = [1, 500]$ has poor accuracy. Thus, obtaining accurate approximations requires 170 times more number of branch and bound iterations #boxes (compared to the $n = 5$ case), hence the computation time blows up.

The last two rows show the results obtained for Flyspeck inequalities involving a single transcendental function (arctan) and six square roots. Each square root is approximated by a quartic minimax polynomial (see 4.1 for the error upper bounds).

Table 4.2: Numerical results for `minimax_optim`

Problem	n	m	n_{lifting}	k	$d_{\max}$	#boxes	time
H3	3	-3.863	4	1	2	53	132 s
				2	4	19	57 s
				3	6	12	101 s
H6	6	-3.33	4	2	4	53	51 s
MC	2	-1.92	0	1	2	8	6.3 s
				2	4	4	3.2 s
				3	6	2	3.0 s
				4	8	0	1.9 s
ML	10	-0.966	5	1	2	1	6.4 s
				2	4	1	8.1 s
				3	6	2	20 s
SWF ($\epsilon = 0$)	5	-430	0	2	4	3	21 s
	10					512	2280 s
9922699028 Flyspeck	6	0	2	2	4	14	244 s
3318775219 Flyspeck						266	4423 s

4.4 Convergence of the Semialgebraic Minimax Approximation Algorithm

Let the relaxation order k be fixed and let denote by t_d^- (resp. t_d^+) the semialgebraic underestimator (resp. overestimator) of t on K obtained with the `minimax_approx` function with a degree-d parameter and the relaxation order k. Given an accuracy $\epsilon > 0$, we prove that the objective function f can be uniformly approximated with absolute accuracy ϵ over the semialgebraic set K with the algorithm `minimax_approx`.

Let $I := [m, M], u \in \mathcal{C}(I)$ and denote by f_d the degree-d minimax polynomial of u on I.

Theorem 4.5 (Jackson's Theorem). *The sequence of best uniform polynomial approximations $(f_d)_{d \in \mathbb{N}}$ to a function u, continuous on $[-1, 1]$, satisfies:*

$$\|u - f_d\|_\infty \leqslant C\omega(1/d),$$

C being a constant.

Proof. For the proof, see [GST07, Chap. 3]. □

Corollary 4.6. *It follows from Theorem 4.5 that the sequence $(f_d)_{d \in \mathbb{N}}$ uniformly converge to u on I.*

Assumption 4.7. *The Haar condition is fulfilled, so that the Remez algorithm always converges towards a polynomial which is optimal.*

The interested reader can find more details about the Haar condition in [Che82, page 74] and the convergence of the first and second algorithms of Remez in [Che82, Chapter 3, Section 8].

Lemma 4.8. *Suppose that Assumption 4.7 holds. Let r be a continuous univariate function defined on the closed interval I and let r_d^- (resp. r_d^+) be the underestimator (resp. overestimator) of r over I obtained at step d of the* `minimax_optim` *iteration loop. Then the sequences $(r_d^-)_{d\in\mathbb{N}}$ and $(r_d^+)_{d\in\mathbb{N}}$ uniformly converge to r on I.*

Proof. When $r \in \mathcal{U} \setminus \mathcal{D}$, then it comes from the fact that $r_d^- = r$ and $r_d^+ = r$. Otherwise, by Assumption 4.7, the Remez procedure yields the sequence of degree-d minimax polynomials $(f_d)_{d\in\mathbb{N}}$. This sequence uniformly converges to r on I, as a consequence of Corollary 4.6. □

The `reduce_lift` function is the identity. Thus, Assumption 3.4 holds as a consequence of Lemma 4.8. Finally, the convergence of the algorithm `minimax_approx` follows from Proposition 3.8.

Now, suppose that the SDP relaxation order k is chosen to be large enough so that $\mathbf{x}_d$ is a global minimizer of t_d^-.

Lemma 4.9. *Under Assumption 4.7, every accumulation point of the sequence $(\mathbf{x}_d)_{d\in\mathbb{N}}$ is a global minimizer of t over K.*

Proof. It follows from Corollary 3.14. □

4.5 Taylor Expansions

It is also possible to approximate the univariate functions with Taylor expansions. In practice, this leads to call an appropriate `taylor` procedure (also available in `Sollya`) instead of the `remez` algorithm (Line `4` of the `minimax_approx` function). An upper bound of the approximation error can be similarly obtained with the `sollyainfnorm` function. Thus, we can derive an optimization algorithm combining Taylor polynomials and SOS.

Definition 4.10 (Real analytic functions)**.** A real analytic function u is an infinitely differentiable function such that the Taylor series at any point c in its domain $T(x) = \sum_{d=0}^{\infty} \dfrac{u(c)^d}{d!}(x-c)^d$ converges to $u(x)$ for x in a neighbourhood of c point-wise and uniformly.

As we consider smooth univariate functions (which are real analytic), then this Taylor polynomials based optimization algorithm shares the same theoretical convergence properties than `minimax_optim` (Lemma 4.9).

Chapter 5

Maxplus Semialgebraic Estimators and Sum of Squares

In Chapter 4 we obtained an optimization method, where the degree was the precision parameter. We now present an algorithm involving low degree maxplus estimators, in which the precision depends on a finite set of control points. We first recall some required background about maxplus approximation (Section 5.1). We next examine the special case of maxplus approximation for semiconvex functions (Section 5.2). The main algorithms based on this maxplus approach are presented in Section 5.3. Numerical experiments on various problems (including Flyspeck inequalities and random inequalities) are depicted in Section 5.4. In this chapter, we assume that the univariate functions are of class $\mathcal{C}^2$.

5.1 The Basis of Maxplus Functions

Let $\mathcal{B}$ be a set of functions $\mathbb{R}^n \to \mathbb{R}$, whose elements will be called *maxplus basis functions*. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, we look for a representation of f as a linear combination of basis functions in the maxplus sense, i.e.,

$$f = \sup_{w \in \mathcal{B}} (a(w) + w) \ , \tag{5.1.1}$$

where $(a(w))_{w \in \mathcal{B}}$ is a family of elements of $\mathbb{R} \cup \{-\infty\}$ (the "coefficients"). The correspondence between the function $x \mapsto f(x)$ and the coefficient function $w \mapsto a(w)$ is a well studied problem, which has appeared in various guises (Moreau conjugacies, generalized Fenchel transforms, Galois correspondences, see [AGK05] for more background).

The idea of maxplus approximation [FM00, McE06, AGL08a] is to choose a space of functions f and a corresponding set $\mathcal{B}$ of basis functions w and to approximate from below a given f in this space by a finite maxplus linear combination, $f \simeq \sup_{w \in \mathcal{F}} (a(w) + w)$, where $\mathcal{F} \subset \mathcal{B}$ is a finite subset. Note that $\sup_{w \in \mathcal{F}} (a(w) + w)$ is not only an approximation but a valid lower bound of f. This is reminiscent of classical linear approximation methods and in particular of the finite element methods, in which a function in an finite dimensional space is approximated by a linear combination of prescribed elementary functions. Note that the term "basis" is abusive in the maxplus setting, as the family of functions $w \in \mathcal{F}$ is generally not free in the tropical sense.

A convenient choice of maxplus basis functions is the following [FM00, AGL08a]. For each constant $\gamma \in \mathbb{R}$, we shall consider the family of quadratic functions $\mathcal{B} = \{w_{\mathbf{y}} \mid \mathbf{y} \in \mathbb{R}^n\}$ where

$$w_{\mathbf{y}}(\mathbf{x}) := -\frac{\gamma}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 \ . \tag{5.1.2}$$

Whereas in classical approximation problems, the ambient function spaces of interest are Sobolev spaces H^k, or spaces $\mathcal{C}^k$ of k times differentiable functions, in the tropical settings, the appropriate spaces, consistent with the choice of quadratic maxplus basis functions, turn out to consist of *semiconvex functions*, which we next examine.

5.2 Maxplus Approximation for Semiconvex Functions

The following definition is standard in variational analysis.

Definition 5.1 (Semiconvex function)**.** Let γ denote a nonnegative constant. A function $\phi : \mathbb{R}^n \to \mathbb{R}$ is said to be *γ-semiconvex* if the function $\mathbf{x} \mapsto \phi(\mathbf{x}) + \frac{\gamma}{2}\|\mathbf{x}\|_2^2$ is convex.

Proposition 5.2. *Let $\mathcal{B}$ denote the set of quadratic functions $w_{\mathbf{y}}$ of the form* (5.1.2) *with $\mathbf{y} \in \mathbb{R}^n$. Then, the set of functions f which can be written as a maxplus linear combination* (5.1.1) *for some function $a : \mathcal{B} \to \mathbb{R} \cup \{-\infty\}$ is precisely the set of lower semicontinuous γ-semiconvex functions.*

Proof. Let us note $h^\star : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ the Legendre-Fenchel transform of a function $h : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$, so that

$$h^\star(p) := \sup_{x \in \mathbb{R}^n} \langle p, x\rangle - h(x) \ .$$

A known fact is that a convex lower semicontinuous function $g : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ is the supremum of the affine functions that it dominates [Roc70, Th. 12.1]. Actually, it is shown there that

$$g(x) = g^{\star\star}(x) = \sup_{p \in \mathbb{R}^n} \langle p, x\rangle - g^\star(p) \ .$$

By applying this result to the function $g(x) = f(x) + \frac{\gamma}{2}\|x\|_2^2$, we deduce that

$$\begin{aligned} f(x) &= \sup_{p \in \mathbb{R}^n} \langle p, x\rangle - \frac{\gamma}{2}\|x\|_2^2 - g^\star(p) \\ &= \sup_{p \in \mathbb{R}^n} -\frac{\gamma}{2}\|x - \frac{1}{\gamma}p\|_2^2 - g^\star(p) + \frac{1}{2\gamma}\|p\|_2^2 \ , \end{aligned}$$

which is of the form (5.1.2).

Conversely, since an arbitrary supremum of γ-semiconvex and lower semicontinuous is also γ-semiconvex and lower semicontinuous, the supremum in (5.1.2) defines a γ-semiconvex and lower semicontinuous function. □

The transcendental functions which we consider here are twice continuously differentiable. Hence, their restriction to any bounded convex set is γ-semiconvex for a sufficiently large γ, so that they can be approximated by finite suprema of the form $\sup_{w \in \mathcal{F}}(a(w) + w)$ with $\mathcal{F} \subset \mathcal{B}$.

The following Theorem 5.3 (see [GMQ11, Theorem 3.2] for the original statement) shows that if $N = |\mathcal{F}|$ basis functions are used, then the best approximation error is $O(1/N^{2/n})$ (the error is the sup-norm, over any compact set), provided that the function to be approximated is of class $\mathcal{C}^2$. We call $\mathcal{D}^2(\phi)(\mathbf{x})$ the Hessian matrix of ϕ at $\mathbf{x}$ and suppose that we approximate the function ϕ by the finite supremum of N γ-semiconvex functions parametrized by $p_i (i = 1, \ldots, N)$ and $a_i (i = 1, \ldots, N)$:

$$\phi \simeq \tilde{\phi}_N := \max_{1 \leqslant i \leqslant N} \{ \frac{\gamma}{2} \|\mathbf{x}\|_2^2 + p_i^T \mathbf{x} + a(p_i) \} \ .$$

Theorem 5.3 (sup approximation error). *Let $\gamma \in \mathbb{R}$, $\epsilon > 0$ and let $K \subset \mathbb{R}^n$ denote any full dimensional compact convex subset. If $\phi : \mathbb{R}^n \mapsto \mathbb{R}$ is $(\gamma - \epsilon)$-semiconvex of class $\mathcal{C}^2$, then there exists a positive constant α depending only on n such that:*

$$\|\phi - \tilde{\phi}_N\|_\infty \sim \frac{\alpha}{N^{2/n}} \Big(\int_K [\det(\mathcal{D}^2(\phi)(\mathbf{x}) + \gamma I_n)]^{\frac{1}{2}} d\mathbf{x} \Big)^{\frac{2}{n}} \ as\ N \to \infty \ .$$

Thus, the best approximation satisfies

$$\|\phi - \tilde{\phi}_N\|_\infty \simeq \frac{C(\phi)}{N^{2/n}} \ , \tag{5.2.1}$$

where the constant $C(\phi)$ is explicit (it depends of $\det(\mathcal{D}^2(\phi) + \gamma I_n)$ and is bounded away from 0 when ϵ is fixed). This estimate indicates that some curse of dimensionality is unavoidable: to get a uniform error of order ϵ, one needs a number of basis functions of order $1/\epsilon^{n/2}$. Equivalently, the approximation error is of order $O(h^{\frac{2}{n}})$ where h is a space discretization step. However, in what follows, we shall always apply the approximation to small dimensional constituents of the optimization problems. For the applications considered in this chapter, $n = 1$.

Remark 5.4. One may notice that the error of maxplus approximation is of the same order as the one obtained by conventional P_1 finite elements under the same regularity assumption.

Remark 5.5. The assumption that $\tilde{\phi}_N$ is of class $\mathcal{C}^2$ in Theorem 5.3 is needed to obtain the asymptotics of the approximation error. However, the best maxplus approximation $\tilde{\phi}_N$ does converge uniformly to ϕ under a milder assumption: it suffices that ϕ be γ-semiconvex and Lipschitz continuous, as shown in [AGL08b]. This is due to the asymmetrical character of the maxplus approximation (a "one-sided" regularity, captured by the semiconvexity condition, is involved).

In this way, starting from a transcendental univariate elementary function $f \in \mathcal{D}$, such as arctan, exp, *etc* , defined on a real bounded interval I, we arrive at a semialgebraic lower bound of f, which is nothing but a supremum of a finite number of quadratic functions.

Example 5.6. Consider the function $f = \arctan$ on an interval $I := [m, M]$. For every point $a \in I$, we can find a constant γ such that

$$\arctan(x) \geqslant \operatorname{par}_a^-(x) := -\frac{\gamma}{2}(x-a)^2 + f'(a)(x-a) + f(a) \ .$$

Choosing $\gamma = \sup_{x \in I} -f''(x)$ always work. However, it will be convenient to allow γ to depend on the choice of a to get tighter lower bounds. Choosing a finite subset $A \subset I$, we arrive at an approximation

$$\forall x \in I, \arctan(x) \geqslant \max_{a \in A} \operatorname{par}_a^-(x) \ . \tag{5.2.2}$$

Semialgebraic overestimators $x \mapsto \min_{a \in A} \mathrm{par}_a^+(x)$ can be defined in a similar way. Examples of such underestimators and overestimators are depicted in Figure 5.1.

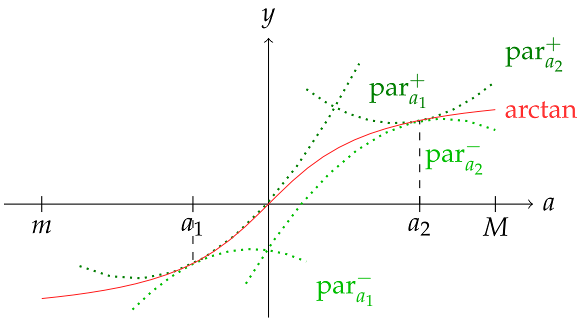


Figure 5.1: Semialgebraic Underestimators and Overestimators for arctan

Example 5.7. Consider the bivariate function $g : (x_1, x_2) \mapsto \sin(x_1 + x_2)$, defined on $K := [-1.5, 4] \times [-3, 3]$, which is a component of the objective function from Problem MC. As in the previous example, we can build underestimators for the sin function. Choosing $\gamma = 1$, for every $(x_1, x_2) \in K$ and every $a \in [-4.5, 7]$, one has:

$$\sin(x_1 + x_2) \geqslant -\frac{1}{2}(x_1 + x_2 - a)^2 + \cos(a)(x_1 + x_2 - a) + \sin(a) \ .$$

Figure5.2 displays the function g (the red surface) and two underestimators of g on K (the green surfaces) obtained with $a := -4.5$ and $a := -2/3$.

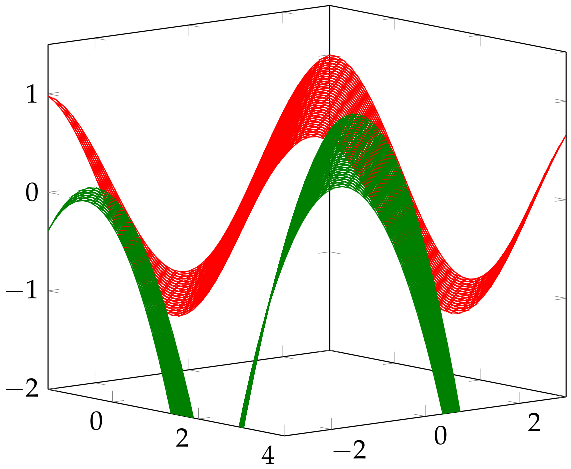


Figure 5.2: Semialgebraic Underestimators for $(x_1, x_2) \mapsto \sin(x_1 + x_2)$

5.3 Combining Maxplus Approximations and Semialgebraic Optimization

5.3.1 An Adaptive Semialgebraic Approximation Algorithm

We consider an instance of Problem (1.1.3). We assume as in Section 4.1 that K is a box. We assimilate the objective function f with its abstract syntax tree t. We assume that the leaves of t are semialgebraic functions in the set $\mathcal{A}$. Other nodes are univariate transcendental functions (arctan, *etc*) or basic operations ($+$, $\times$, $-$, $/$). Moreover, we suppose that the univariate transcendental functions are lower semicontinuous and semiconvex.

We first introduce the auxiliary algorithm `samp_approx`, which relies on the univariate approximation function `samp_unary_approx` described in Figure 5.3. As for the minimax approximation algorithm `minimax_approx`, we take the identity function for `reduce_lift`.

Given an abstract syntax tree t and a box K, this algorithm computes lower and upper bounds of t over K and maxplus approximations of t by means of semialgebraic functions. It is also parametrized by a finite sequence of control points $\mathbf{x}_1, \dots, \mathbf{x}_p \in K$ used to approximate transcendental functions by means of parabola.

```
Input: univariate function $r$, $I$, child $c$, control points sequence $s = \mathbf{x}_1, \dots, \mathbf{x}_p \in K$
Output: underestimator $r^-$, overestimator $r^+$
1: if $r \in \mathcal{D}$ then
2:     $a_j := c(\mathbf{x}_j)$ for $j \in \{1, \dots, p\}$
3:     $\mathrm{par}^-_{a_j}, \mathrm{par}^+_{a_j} :=$ build_par$(r, I, a_j)$ for $j \in \{1, \dots, d\}$
4:     $r^- := \max_{1 \leqslant j \leqslant p} \mathrm{par}^-_{a_j}$
5:     $r^+ := \min_{1 \leqslant j \leqslant p} \mathrm{par}^+_{a_j}$
6: else if $r \in \mathcal{U} \setminus \mathcal{D}$ then
7:     $r^- := r, r^+ := r$
8: end
9: return $r^-, r^+$
```

Figure 5.3: `samp_unary_approx`: Maxplus Approximation Algorithm

The algorithm `samp_approx` is defined by induction on the abstract syntax tree t.

If t is a semialgebraic function, we obtain estimators and bounds, following the general procedure `approx` (Figure 3.2, Line 2), using the semialgebraic optimization functions `min_sa` and `max_sa`.

If the root of t corresponds to a univariate function node $r \in \mathcal{U}$ taking a single child c as argument, we obtain recursively an interval I, enclosing the values of c over K. We compute a finite sequence of points $a_1, \dots, a_p \in I$ from the control points sequence $s = \mathbf{x}_1, \dots, \mathbf{x}_p \in K$ (Figure 5.3, Line 2). Then we apply the function `build_par` (Line 3) that builds the parabola at $a_1, \dots, a_p$, by using the convexity/semiconvexity properties of r on I, as explained in Section 5.2. An underestimator t^- as well as an overestimator t^+ are determined by composition (recall the `compose_approx` function, described in Figure 3.3) of the parabola with c^- and c^+. These approximations t^- and t^+ are semialgebraic functions of $\mathcal{A}$, whence we can also compute their lower and upper bounds using `min_sa` and `max_sa`.

If t is a binary operation whose arguments are two children c_1 and c_2, we use the semialgebraic arithmetic algorithm `compose_bop` (see Figure 3.4) to determine valid estimators.

5.3.2 An Optimization Algorithm based on Maxplus Estimators

Our main optimization algorithm `samp_optim`, relies on `samp_approx` and chooses the sequence of control points s dynamically. It is obtained by taking `approx` $:=$ `samp_approx` (see Figure 3.5) and `unary_approx` $:=$ `samp_unary_approx`. Here, we choose $iter := 0$ and we call $iter_{\max}$ times `samp_approx` inside the loop from Lines 4 to 11 in `optim`.

Remark 5.8. It is not mandatory to always compute recursively the underestimators and overestimators as well as bounds of all the nodes and the leaves of the abstract syntax tree. Instead, we "decorate" the tree with interval and semialgebraic values containing these information, based on previous iterations. When we call the procedure `samp_unary_approx` at step p (the sequence of control points is $s = \mathbf{x}_1, \ldots, \mathbf{x}_p$), we only need to get the equations of the parabola $\text{par}^-_{a_p}$ and $\text{par}^+_{a_p}$.

Example 5.9 (Lemma$_{9922699028}$ Flyspeck). We continue Example 2.12. Since we computed lower and upper bounds (m and M) for $f_{\text{sa}} := \frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}}$, we know that the f_{sa} argument of arctan lies in $I := [m, M]$. We describe three iterations of the algorithm `samp_approx`. Figure 5.4 illustrates the related semialgebraic underestimators hierarchy.

0. Multiple evaluations of f return a set of values and we obtain a first minimizer guess x_1 := `argmin(randeval(`f`))` corresponding to the minimal value of the set. One has $x_1 := (4.8684, 4.0987, 4.0987, 7.8859, 4.0987, 4.0987)$.

1. We compute $a_1 := f_{\text{sa}}(x_1) = 0.3962$, get the equation of $\text{par}^-_{a_1}$ with `build_par` and finally compute $m_1 \leqslant \min_{x \in K}\{l(\mathbf{x}) + \text{par}^-_{a_1}(f_{\text{sa}}(\mathbf{x}))\}$. For $k = 2$, we obtain $m_1 = -0.2816 < 0$ and $x_2 := (4, 6.3504, 6.3504, 6.3504, 6.3504, 6.3504)$.

2. From the second control point, we get $a_2 := f_{\text{sa}}(x_2) = -0.4449$ and $m_2 \leqslant \min_{x \in K}\{l(\mathbf{x}) + \max_{1 \leqslant i \leqslant 2}\{\text{par}^-_{a_i}(f_{\text{sa}}(\mathbf{x}))\}\}$. For $k = 2$, we get $m_2 = -0.0442 < 0$ and $x_3 := (4.0121, 4.0650, 4.0650, 6.7455, 4.0650, 4.0650)$.

3. From the third control point, we get $a_3 := f_{\text{sa}}(x_3) = 0.1020$, $\text{par}^-_{a_3}$ and $m_3 \leqslant \min_{x \in K}\{l(\mathbf{x}) + \max_{1 \leqslant i \leqslant 3}\{\text{par}^-_{a_i}(f_{\text{sa}}(\mathbf{x}))\}\}$. For $k = 2$, we obtain $m_3 = -0.0337 < 0$ and get a new minimizer x_4.

Example 5.9 illustrates two common difficulties we encountered so far.

First, many iterations of `samp_optim` are required to get a good underestimator of f. Hence, the number of lifting variables and equalities constraints may become large enough to make interior-point methods fail (when no strictly feasible solutions exist) and SDP solvers return bad numerical results.

Moreover, if we cannot increase the SDP relaxation order k sufficiently, then we cannot ensure convergence of `samp_optim` (see Corollary 3.14). For a given relaxation order and a control points sequence $(\mathbf{x}_1, \ldots, \mathbf{x}_p)$, we always compute the images sequence $a_1 := f_{\text{sa}}(\mathbf{x}_1), \ldots, a_p := f_{\text{sa}}(\mathbf{x}_p)$, then a lower bound of the minimum of the

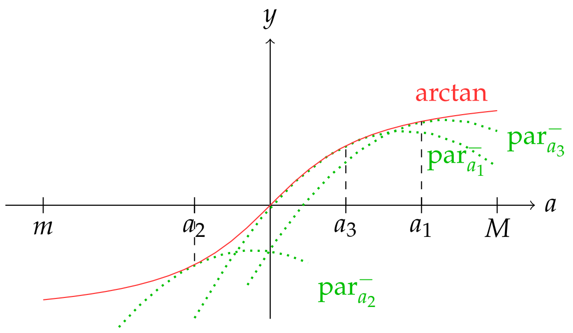


Figure 5.4: A hierarchy of Semialgebraic Underestimators for arctan

semialgebraic function $\mathbf{x} \mapsto \max_{1 \leqslant i \leqslant d} \{\mathrm{par}_{a_i}^-(f_{\mathrm{sa}}(\mathbf{x}))\}$. If the number of parabola becomes large enough, this lower bound could be negative even though the actual minimum of the semialgebraic underestimator is positive. Section 5.3.4 describes a subdivision algorithm to handle this problem.

5.3.3 Convergence Results

We note t_p^- the underestimator (resp. t_p^+ the overestimator) computed at the p^{th} iteration of the algorithm `samp_optim` and by $\mathbf{x}_p$ the corresponding minimizer candidate.

Lemma 5.10 (Uniform convergence of the semialgebraic maxplus estimators)**.** *The estimators sequences $(t_p^-)_p$ and $(t_p^+)_p$ uniformly converge to t on the box K.*

Proof. First, we prove that if r is a univariate function defined on an interval I, then the function `minimax_unary_approx` provides a sequence of underestimators $(r_p^-)_p$ (resp. overestimators $(r_p^+)_p$), which uniformly converge to r on I. This is trivial when r is not transcendental, since $r^- := r$ and $r^+ := r$. Otherwise, $r \in \mathcal{D}$ and we can apply Theorem 5.3 that implies the uniform convergence of the maxplus estimators, obtained with the `build_par` procedure.

As for Lemma 4.8, Assumption 3.4 holds and the convergence of the algorithm `samp_approx` is a direct consequence of Proposition 3.8. □

Furthermore, by applying Corollary 3.14, each limit point of the sequence of control points yields a global minimizer of t over K.

5.3.4 Refining Bounds by Domain Subdivisions

The time complexity of our algorithm strongly depends on the relaxation order k. Indeed, if p is the number of the control points, then the number of moment variables in the SDP problem Q_k is in $O((2k)^{n+p})$ and the size of linear matrix inequalities involved are in $O(k^{n+p})$. The complexity of `samp_optim` is therefore polynomial in k.

A small relaxation order ensures fast computation of the lower bounds but the relaxation gap may remain too high to ensure the convergence of the algorithm. This is particularly critical when we want to certify that a given transcendental multivariate function is non-negative. In this section, we explain how to reduce the relaxation gap using domain subdivision in order to solve problems of the form (1.1.3).

Suppose that the algorithm `samp_optim` returns a negative lower bound m and a global minimizer candidate $\mathbf{x}_c$. Our approach consists in cutting the initial box K in several boxes $(K_i)_{1 \leqslant i \leqslant N}$. We explain the partitioning of K with the following heuristic.

Let $\mathcal{B}_{\mathbf{x}_c,r}$ be the intersection of the sup-ball of center $\mathbf{x}_c$ and radius r with the set K. Then, let $f_{\mathbf{x}_c,r}$ be the quadratic polynomial defined by:

$$\begin{aligned} f_{\mathbf{x}_c,r} : \mathcal{B}_{\mathbf{x}_c,r} &\longrightarrow \mathbb{R} \\ x &\longmapsto f(\mathbf{x}_c) + \mathcal{D}(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) \\ &\quad + \frac{1}{2}(\mathbf{x} - \mathbf{x}_c)^T \mathcal{D}^2(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) \\ &\quad + \frac{1}{2}\lambda \|\mathbf{x} - \mathbf{x}_c\|_2^2 \ , \end{aligned} \tag{5.3.1}$$

with λ given by:

$$\lambda := \min_{\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}} \{\lambda_{\min}(\mathcal{D}^2(f)(\mathbf{x}) - \mathcal{D}^2(f)(\mathbf{x}_c))\} \ . \tag{5.3.2}$$

Lemma 5.11. $\forall \mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}$, $f(\mathbf{x}) \geqslant f_{\mathbf{x}_c,r}$.

Proof. From the first order Taylor expansion with the integral form for the remainder, the following holds:

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}_c) + \mathcal{D}(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) \\ &\quad + \int_0^1 (1-\tau)(\mathbf{x} - \mathbf{x}_c)^T \mathcal{D}^2(f)(\mathbf{x}_c + \tau(\mathbf{x} - \mathbf{x}_c))(\mathbf{x} - \mathbf{x}_c) d\tau, \end{aligned} \tag{5.3.3}$$

for all $\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}$. Then, notice that, for all $\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}$:

$$\int_0^1 (1-\tau)(\mathbf{x} - \mathbf{x}_c)^T \mathcal{D}^2(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) d\tau = \frac{1}{2}(\mathbf{x} - \mathbf{x}_c)^T \mathcal{D}^2(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) \ .$$

Define $\delta_{\tau,\mathbf{x}_c}(\mathbf{x}) := \mathbf{x}_c + \tau(\mathbf{x} - \mathbf{x}_c)$ and the quadratic polynomial $q_{\mathbf{x}_c,r}$:

$$q_{\mathbf{x}_c,r}(\mathbf{x}) := f(\mathbf{x}_c) + \mathcal{D}(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_c)^T \mathcal{D}^2(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) \ .$$

Then, for all $\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}$,

$$\begin{aligned} f(\mathbf{x}) &= q_{\mathbf{x}_c,r}(\mathbf{x}) \\ &\quad + \int_0^1 (1-\tau)(\mathbf{x} - \mathbf{x}_c)^T \Big[\mathcal{D}^2(f)(\delta_{\tau,\mathbf{x}_c}(\mathbf{x})) - \mathcal{D}^2(f)(\mathbf{x}_c)\Big](\mathbf{x} - \mathbf{x}_c) d\tau. \end{aligned} \tag{5.3.4}$$

Let $H_{\tau,\mathbf{x}_c}(\mathbf{x}) := \mathcal{D}^2(f)(\delta_{\tau,\mathbf{x}_c}(\mathbf{x})) - \mathcal{D}^2(f)(\mathbf{x}_c)$. By definition of the minimal eigenvalue, for all $\tau \in [0,1]$, for all $\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}$,

$$(\mathbf{x} - \mathbf{x}_c)^T H_{\tau,\mathbf{x}_c}(\mathbf{x})(\mathbf{x} - \mathbf{x}_c) \geqslant \lambda_{\min}(H_{\tau,\mathbf{x}_c}(\mathbf{x}))\|\mathbf{x} - \mathbf{x}_c\|_2^2 \ .$$

Furthermore, for all $\tau \in [0,1]$, for all $\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}$:

$$\lambda_{\min}(H_{\tau,\mathbf{x}_c}(\mathbf{x})) \geqslant \min_{\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}, \tau \in [0,1]} \{\lambda_{\min}(H_{\tau,\mathbf{x}_c}(\mathbf{x}))\} \ .$$

Moreover, for all $\tau \in [0,1]$, for all $\mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}$,

$$\|\delta_{\tau,\mathbf{x}_c}(\mathbf{x})\|_\infty = \|(1-\tau)\mathbf{x}_c + \tau\mathbf{x}\|_\infty \leqslant (1-\tau)r + \tau r = r \ ,$$

then, one can write the following:

$$\mathcal{B}_{\mathbf{x}_c,r} := \{\delta_{\tau,\mathbf{x}_c}(\mathbf{x}) \mid \tau \in [0,1], \mathbf{x} \in \mathcal{B}_{\mathbf{x}_c,r}\} \ .$$

Hence, one has:

$$\min_{\mathbf{x}\in\mathcal{B}_{\mathbf{x}_c,r},\tau\in[0,1]} \lambda_{\min}(H_{\tau,\mathbf{x}_c}(\mathbf{x})) = \min_{\mathbf{x}\in\mathcal{B}_{\mathbf{x}_c,r}} \{\lambda_{\min}(\mathcal{D}^2(f)(\mathbf{x}) - \mathcal{D}^2(f)(\mathbf{x}_c))\} = \lambda \ .$$

Therefore, we obtain a lower bound of $\frac{1}{2}\lambda\|\mathbf{x}-\mathbf{x}_c\|_2^2$ for the integral of the right hand side of (5.3.4), that completes the proof.

□

To underestimate the value of λ, we determine the following interval matrix:

$$\widetilde{\mathcal{D}^2(f)} := ([\underline{d_{ij}}, \overline{d_{ij}}])_{1\leqslant i,j\leqslant n} \ ,$$

containing coarse bounds of the difference $(\mathcal{D}^2(f)(\mathbf{x}) - \mathcal{D}^2(f)(\mathbf{x}_c))$ on $\mathcal{B}_{\mathbf{x}_c,r}$ using interval arithmetic or `samp_approx` with a small number of control points and a low SDP relaxation order. We then apply on $\widetilde{\mathcal{D}^2(f)}$ a robust SDP method on interval matrix described by Calafiore and Dabbene in [CD08] and obtain a lower bound λ' of λ.

Now, we detail this procedure in our particular case.

Let $\widetilde{H} := ([\underline{d_{ij}}, \overline{d_{ij}}])_{1\leqslant i,j\leqslant n}$ be such an interval matrix. The problem is to find the minimal eigenvalue of $\widetilde{H}$:

$$\lambda' := \lambda_{\min}(\widetilde{H}) = \min_{H\in\mathcal{S}_n}\{\lambda_{\min}(H) : \underline{d_{ij}} \leqslant H_{ij} \leqslant \overline{d_{ij}} \ \ (1 \leqslant i,j \leqslant n)\} \ . \tag{5.3.5}$$

For each interval $[\underline{d_{ij}}, \overline{d_{ij}}]$, define the symmetric matrix B:

$$B_{ij} := \max\{\mid \underline{d_{ij}} \mid, \mid \overline{d_{ij}} \mid\}, \ \ 1 \leqslant i,j \leqslant n \ .$$

Let $\mathcal{S}^n$ be the set of diagonal matrices of sign:

$$\mathcal{S}^n := \{\text{diag}\ (s_1,\ldots,s_n),\ s_1 = \pm 1, \ldots\ s_n = \pm 1\} \ .$$

The following lemma specializes the result of the robust optimization procedure with reduced vertex set [CD08, Theorem 2.1].

Lemma 5.12 (Interval matrix eigenvalue optimization with reduced vertex set)**.** *The robust interval SDP Problem* (5.3.5) *is equivalent to the following SDP in the single variable* $t \in \mathbb{R}$*:*

$$\begin{cases} \min_t & -t \\ s.t. & -tI - SBS \succcurlyeq 0 \ , \\ & S = diag\ (1, S'), \quad \forall S' \in \mathcal{S}^{n-1} \ . \end{cases}$$

In practice, we solve the Problem $\lambda' := \lambda_{\min}(\widetilde{\mathcal{D}^2(f)})$ and obtain a valid underestimator $f^-_{\mathbf{x}_c,r}$ of $f_{\mathbf{x}_c,r}$ on $\mathcal{B}_{\mathbf{x}_c,r}$:

$$f^-_{\mathbf{x}_c,r} := q_{\mathbf{x}_c,r}(\mathbf{x}) + \lambda' \|\mathbf{x} - \mathbf{x}_c\|_2^2 \enspace .$$

Notice that $f^-_{\mathbf{x}_c,r}$ underestimates f on $\mathcal{B}_{\mathbf{x}_c,r}$, as a consequence of Lemma 5.11.

Our branch and bound algorithm `samp_bb` (see Figure 5.5) relies on the semialgebraic optimization procedure `samp_optim`. The `dicho_ball` function (Line 3) computes by dichotomy the sup-ball $\mathcal{B}_{\mathbf{x}_c,r}$ of maximal radius r such that the underestimator $f_{\mathbf{x}_c,r}$ is nonnegative on $\mathcal{B}_{\mathbf{x}_c,r}$. The nonnegativity of $f^-_{\mathbf{x}_c,r}$ can be certified with `min_sa`.

Remark 5.13. For the sake of clarity, we mention that λ and λ' depend both on the choice of the sup-ball $\mathcal{B}_{\mathbf{x}_c,r}$. At each step of the dichotomy performed in the auxiliary function `dicho_ball`, we solve an instance of Problem (5.3.5).

```
Input: tree $t$, box $K$, $iter_{\max}$
Output: lower bound $m$
 1: $m, \mathbf{x}_c :=$ samp_optim($t, K, iter_{\max}$)
 2: if $m < 0$ then
 3:     $\mathcal{B}_{\mathbf{x}_c,r} :=$ dicho_ball($t, K, \mathbf{x}_c$)
 4:     Obtain a partition of $K \setminus \mathcal{B}_{\mathbf{x}_c,r} := (K_i)_{1 \leqslant i \leqslant N}$
 5:     $K_0 := \mathcal{B}_{\mathbf{x}_c,r}$
 6:     $m := \min_{0 \leqslant i \leqslant N} \{$samp_bb($t, K_i, iter_{\max}$)$\}$
 7:     return $m$
 8: else
 9:     return $m$
10: end
```

Figure 5.5: Description of `samp_bb`

An illustration of our subdivision algorithm is given in Figure 5.6 in the two dimensional case.

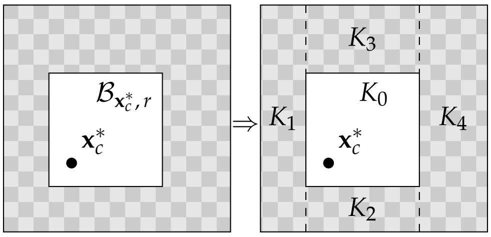


Figure 5.6: A two dimensional example for our box subdivision

5.4 Numerical Results

5.4.1 Flyspeck Inequalities

We next present the numerical results obtained with our method for both small and medium-sized inequalities taken from the Flyspeck project.

In Tables 5.1 and 5.2, the inequalities are indexed by the first four digits of the hash code. We also indicate in subscript the number of variables involved in each inequality. The integer $n_{\mathcal{D}}$ represents the number of transcendental univariate nodes in the corresponding abstract syntax trees. The parameter $k_{\max}$ is the highest SDP relaxation order used to solve the polynomial optimization problems with SPARSEPOP. We note #POP the total number of polynomial optimization problems that have to be solved to prove the inequality and by #boxes the number of domain cuts that are performed during the subdivision algorithm. Finally, m is the lower bound of the function f on K that we obtain with our method, i.e. the minimum of all the computed lower bounds of f among the #boxes sub-boxes of K.

The inequalities reported in Table 5.1 are similar to the one presented in Example 1.3. They involve the addition of the function $\mathbf{x} \mapsto \arctan \frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}}$ with an affine function over $\sqrt{x_i}$ $(1 \leqslant i \leqslant 6)$.

Table 5.1: Results for small-sized Flyspeck inequalities

Inequality id	$n_{\mathcal{D}}$	$k_{\max}$	#POP	#boxes	m	time
9922_6	1	2	222	27	3.07×10^{-5}	1200 s
3526_6	1	2	156	17	4.89×10^{-6}	780 s
6836_6	1	2	173	22	4.68×10^{-5}	840 s
6619_6	1	2	163	21	4.57×10^{-5}	783 s
3872_6	1	2	250	30	7.72×10^{-5}	1224 s
3139_6	1	2	162	17	1.03×10^{-5}	775 s
4841_6	1	2	624	73	2.34×10^{-6}	3014 s
3020_5	1	3	80	9	2.96×10^{-5}	1847 s
3318_6	1	3	26	2	3.12×10^{-5}	4324 s

Table 5.2 provides the numerical results obtained on medium-sized Flyspeck inequalities. Inequalities 7394_i $(3 \leqslant i \leqslant 5)$ are obtained from a same inequality 7394_6 involving six variables, by instantiating some of the variables by a constant value. Inequalities 7726_6 and 7394_6 are both of the form $l(\mathbf{x}) + \sum_{i=1}^{3} \arctan(q_i(\mathbf{x}))$ where l is an affine function over $\sqrt{x_i}$, $q_1(\mathbf{x}) := \frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}}$, $q_2(\mathbf{x}) := q_1(x_2, x_1, x_3, x_5, x_4, x_6)$ and $q_3(\mathbf{x}) := q_1(x_3, x_1, x_2, x_6, x_4, x_5)$.

Table 5.2: Results for medium-size Flyspeck inequalities

Inequality id	$n_{\mathcal{D}}$	$k_{\max}$	#POP	#boxes	m	time
7726_6	3	2	450	70	1.22×10^{-6}	12240 s
7394_3	3	3	1	0	3.44×10^{-5}	11 s
7394_4	3	3	47	10	3.55×10^{-5}	1560 s
7394_5	3	3	290	55	3.55×10^{-5}	43200 s

5.4.2 Random Inequalities

In Table 5.3, we compared `samp_optim` with the MATLAB `intsolver` toolbox [Mon09] (based on the Newton interval method [HG83]) for random inequalities involving two transcendental functions. Let n be the number of variables and m the lower bound that we obtain.

The functions that we consider are of the form $x \mapsto \arctan(p(\mathbf{x})) + \arctan(q(\mathbf{x}))$, where p is a four-degree polynomial and q is a quadratic polynomial. All variables lie in $[0, 1]$. Both p and q have random coefficients (taken in $[0, 1]$) and are sparse. The speed-up factor results indicate that for such medium-scale examples, our method may outperform interval arithmetic.

Table 5.3: Comparison results for random examples

n	m	time t_1 (`samp_optim` with $k = 3$)	time t_2 (`intsolver`)	Speed-up Factor (t_2/t_1)
3	0.4581	3.8 s	15.5 s	4.1
4	0.4157	12.9 s	172.1 s	13.3
5	0.4746	58 s	612 s	10.6
6	0.4476	276 s	12240 s	44.3

5.4.3 Certification of MetiTarski Bounds

Here we explain how to certify the semialgebraic univariate estimators of MetiTarski with our tool.

MetiTarski [AP10] is a theorem prover that can handle nonlinear inequalities involving special functions such as ln, cos, *etc* . These univariate transcendental functions (as well as the square root) are approximated by a hierarchy of estimators which are rational functions derived from Taylor expansions or continued fractions expansions (for more details, see Cuyt et al. [CBBH08]).

The framework available in MetiTarski to check inequalities is similar to the optimization algorithms `minimax_optim` or `samp_optim`. The rational function estimators are used instead of the best uniform approximation polynomials. This may provide more accurate bounds.

For instance, consider the logarithm function on the interval $[1.1, 9]$. The second (resp. third) overestimator $\ln_2^+$ (resp. $\ln_3^+$) are defined as follows, for all $x \in I$:

$$\begin{aligned} \ln_2^+(x) &:= \frac{(x+5)(x-1)}{2(2x+1)} \ , \\ \ln_3^+(x) &:= \frac{(x^2+19x-10)(x-1)}{3(3x^2+6x+1)} \ . \end{aligned}$$

Checking the validity of these estimators on real closed intervals leads to give formal proofs of nonlinear univariate inequalities. For the two overestimators of ln, the inequalities are:

$$\forall x \in [1.1, 10], \ln(x) \leqslant \ln_2^+(x) \ , \tag{$\ln_2^+$}$$

$$\forall x \in [2, 10], \ln(x) \leqslant \ln_3^+(x) \ . \tag{$\ln_3^+$}$$

Table 5.4: Comparison results for MetiTarski estimators certification

Inequality Id.	`samp_optim`			`minimax_optim`		
	#s	#boxes	time	d	#boxes	time
$\ln_2^+$	1	56	18.3 s	2	9	4.3 s
	2	33	13.5 s	4	6	3.4 s
	3	28	14.2 s	6	6	3.7 s
$\ln_3^+$	1	56	14.4 s	2	39	26 s
	2	56	19.1 s	5	3	2 s
	3	30	16.4 s	6	3	2.4 s
$\arctan_2^+$	1	81	16.5 s	2	76	20 s
	2	19	7.6 s	4	6	4.1 s
	3	12	7.3 s	6	2	1.4 s
$\arctan_5^+$	1	171	60 s	2	10	7.4 s
	2	86	57 s	4	5	3.9 s
	3	47	39 s	6	4	3.4 s

Similarly, the inequalities that imply the validity of MetiTarski overestimators for the arctan function are:

$$\forall x \in [-1, -0.01], \arctan(x) \leqslant \frac{3x}{x^2+3} \;, \qquad (\arctan_2^+)$$

$$\forall x \in [0.5, 5], \arctan(x) \leqslant \frac{(64x^4 + 735x^2 + 945)x}{15(15x^4 + 70x^2 + 63)} \;. \qquad (\arctan_5^+)$$

In table 5.4, we present some comparison results obtained with the procedure `minimax_optim`, using a sequence of control points of length #s and `samp_optim`, using a degree-d minimax polynomial.

The results indicate that univariate inequalities are easier to solve by minimax polynomial approximation since the number of subdivisions decreases, as well as the computation time.

Chapter 6

The Templates Method

Here, we improve the maxplus approximation method presented in Chapter 5 by reducing the complexity of semialgebraic optimization problems that we solve. We explain how templates are related with maxplus estimators (Section 6.1). Maxplus based template approximation can be used to obtain coarse bounds for non-trivial POP (Section 6.2). Furthermore, semialgebraic functions can be approached by a sequence of polynomial templates which converge to the best polynomial underestimators for the L_1 norm (Section 6.3). The nonlinear template optimization algorithm is presented in Section 6.4. Finally, we analyse the performance of the algorithm (Section 6.5).

6.1 Max-plus Approximations and Nonlinear Templates

The non-linear template method is a refinement of polyhedral based methods in static analysis [SSM05]. It can also be closely related to the non-linear extension [AGG12] of the template method and to the class of affine relaxation methods [Mes99].

Templates allow one to determine invariants of programs by considering parametric families of subsets of $\mathbb{R}^n$ of the form $S(\alpha) = \{\mathbf{x} \mid w_i(\mathbf{x}) \leqslant \alpha_i,\ 1 \leqslant i \leqslant p\}$, where the vector $\alpha \in \mathbb{R}^p$ is the parameter, and $w_1, \dots, w_p$ (the template) are fixed possibly non-linear functions, tailored to the program characteristics.

Notice that by taking a trivial template (bound constraints, *i.e.* , functions of the form $\pm x_i$), the template method specializes to a version of interval calculus, in which bounds are derived by SDP techniques. By comparison, templates allow one to get tighter bounds, taking into account the correlations between the different variables. In most basic examples, the functions w_i of the template are linear or quadratic functions.

The max-plus basis method introduced in Section 5.1 is equivalent to the approximation of the epigraph of a function by a set $S(\alpha)$. This method involves the approximation from below of a function f in n variables by a supremum

$$f \gtrapprox g := \sup_{1 \leqslant i \leqslant p} \lambda_i + w_i \ . \tag{6.1.1}$$

The functions w_i are fixed in advance, or dynamically adapted by exploiting the problem structure. The parameters λ_i are degrees of freedom.

The template method consists in propagating approximations of the set of reachable values of the variables of a program by sets of the form $S(\alpha)$. The non-linear template and max-plus approximation methods are somehow related. Indeed, the 0-level set of g, $\{\mathbf{x} \mid g(\mathbf{x}) \leqslant 0\}$, is nothing but $S(-\lambda)$, so templates can be recovered from max-plus approximations and vice versa.

The functions w_i are usually required to be quadratic polynomials,

$$w_i(\mathbf{x}) = p_i^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T A_i \mathbf{x} \ ,$$

where $p_i \in \mathbb{R}^n$ and A_i is a symmetric matrix. A basic choice is $A_i = -\gamma I_n$, where γ is a fixed constant. Then, the parameters p remain the only degrees of freedom.

A basic question here is to estimate the number of template basis functions needed to attain a prescribed accuracy. The typical result stated in Theorem 5.3 is a corollary of techniques of Grüber concerning the approximation of convex bodies by circumscribed polytopes. For optimization purposes, a uniform approximation is not needed (one only needs an approximation tight enough near the optimum, for which fewer basis functions are enough).

We shall also apply the approximation by templates to certain relevant small dimensional projections of the set of lifted variables, leading to a smaller effective n.

6.2 Reducing the Size of SOS Relaxations for Polynomial Optimization Problems

Let f be a degree-d multivariate polynomial. When d is too high, then the first SDP Lasserre relaxation of order $k_{\min} := \lceil d/2 \rceil$ may be intractable. The aim of this section is to present an approximation scheme which allows to provide coarse lower bounds for such intractable cases.

We first recall some basic definitions.

Definition 6.1. Given a symmetric real-valued matrix $M \in \mathcal{S}_n$, the spectral radius of M is given by:

$$\rho(M) := \max(\lambda_{\max}(M), -\lambda_{\min}(M)) \ .$$

Definition 6.2. The L_1 matrix norm, subordinate to the L_1 vector norm, is given by:

$$||M||_1 := \max_{x \neq 0} \frac{||Mx||_1}{||x||_1} \ .$$

One can easily prove that $||M||_1$ is the maximum column sum of the absolute values of the entries of M. In the sequel, we use the following inequality:

Proposition 6.3.

$$\rho(M) \leqslant ||M||_1 \ .$$

Let $K \subset \mathbb{R}^n$ be a box and consider a multivariate nonlinear function $f : K \to \mathbb{R}$. In Section 5.3.4, we derived coarse underestimators of f on sup-balls, using robust Semidefinite programming. Here, we define similar quadratic polynomials on the box K.

Definition 6.4. Given $\mathbf{x}_c \in K$, we define the quadratic polynomial $f_{\mathbf{x}_c}$ as follows:

$$\begin{aligned} f_{\mathbf{x}_c} : K &\longrightarrow \mathbb{R} \\ x &\longmapsto f(\mathbf{x}_c) + \mathcal{D}(f)(\mathbf{x}_c)\,(\mathbf{x} - \mathbf{x}_c) \\ &\quad + \frac{1}{2}(\mathbf{x} - \mathbf{x}_c)^T \mathcal{D}^2(f)(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) \\ &\quad + \frac{1}{2}\lambda' \|\mathbf{x} - \mathbf{x}_c\|_2^2 \ , \end{aligned} \tag{6.2.1}$$

with,

$$\lambda' \leqslant \lambda := \min_{\mathbf{x} \in K} \{\lambda_{\min}(\mathcal{D}^2(f)(\mathbf{x}) - \mathcal{D}^2(f)(\mathbf{x}_c))\} \ . \tag{6.2.2}$$

The following statement is analogous to Lemma 5.11:

Lemma 6.5. $\forall \mathbf{x} \in K,\ f(\mathbf{x}) \geqslant f_{\mathbf{x}_c}$.

In this particular case, notice that the entries of the matrix $(\mathcal{D}^2(f)(\mathbf{x}) - \mathcal{D}^2(f)(\mathbf{x}_c))$ are degree-$(d-2)$ polynomials. To underestimate the value of λ, we determine an interval matrix $\widetilde{\mathcal{D}^2(f)} := ([\underline{d_{ij}}, \overline{d_{ij}}])_{1 \leqslant i,j \leqslant n}$, using SOS techniques on the entries of the Hessian difference on K. It leads to SDP relaxations of minimal order $(k_{\min} - 1)$.

6.2.1 Lower Bounds of Interval Matrix Minimal Eigenvalues

Different approximations of λ can be considered.

Tight lower bound of λ

Let λ'_1 be the solution of the single variable semidefinite program described in Lemma 5.12.

Proposition 6.6. $\lambda'_1 \leqslant \lambda$.

Proof. Let us consider the minimal eigenvalue λ' of the interval matrix $\widetilde{\mathcal{D}^2(f)}$, where each entry is obtained by solving SOS relaxations by Lasserre. This hierarchy provides lower (resp. upper) bounds for polynomial minimization (resp. maximization) problems, thus λ' is a lower bound of λ. Moreover λ'_1 is a lower bound of λ' thus one has $\lambda'_1 \leqslant \lambda$, the desired result. □

However this method introduces a subset of sign matrices of cardinal 2^{n-1}, thus reduces the problem to a manageable size only if n is small.

Coarse lower bound of λ

Here, one writes $\widetilde{\mathcal{D}^2(f)} := X + Y$, where X and Y are defined as follows:

$$X_{ij} := \Big[\frac{\underline{d_{ij}} + \overline{d_{ij}}}{2}, \frac{\underline{d_{ij}} + \overline{d_{ij}}}{2}\Big], \quad Y_{ij} := \Big[-\frac{\overline{d_{ij}} - \underline{d_{ij}}}{2}, \frac{\overline{d_{ij}} - \underline{d_{ij}}}{2}\Big] \ .$$

Define $\lambda'_2 := \lambda_{\min}(X) - \max_{1 \leqslant i \leqslant n} \Big\{ \sum_{j=1}^{n} \frac{\overline{d_{ij}} - \underline{d_{ij}}}{2} \Big\}$.

Proposition 6.7. $\lambda_2' \leqslant \lambda$.

Proof. By concavity and homogeneity of the $\lambda_{\min}$ function, one has:

$$\lambda_{\min}(X+Y) \geqslant \lambda_{\min}(X) + \lambda_{\min}(Y) = \lambda_{\min}(X) - \lambda_{\max}(-Y) \ . \tag{6.2.3}$$

Using Proposition 6.3, the following inequality holds:

$$\lambda_{\max}(-Y) \leqslant \max_{1 \leqslant i \leqslant n} \Big\{ \sum_{j=1}^{n} \frac{\overline{d_{ij}} - \underline{d_{ij}}}{2} \Big\} \ . \tag{6.2.4}$$

□

The matrix X is real valued and symmetric matrix, thus one can compute its minimal eigenvalue with the classical semidefinite program:

$$\left\{ \begin{array}{ll} \min & -t \\ s.t. & X - tI \succcurlyeq 0 \ . \end{array} \right.$$

Finally, we can compute a coarse lower bound λ_2' of λ with a procedure which is polynomial in n.

6.2.2 A Template Algorithm for POP

We present a first template optimization algorithm `pop_template_optim` (depicted in Figure 6.1), derived from the general procedure `optim` (Figure 3.5). Here, instead of taking the identity function for `reduce_lift`, we provide non-trivial estimators of the degree-d multivariate polynomial f, using the techniques presented in Section 6.2.1.

A sub-routine `build_quadratic_form` returns the polynomial defined in (6.2.1). In particular, one has $f_{\mathbf{x}_c,1} :=$ `build_quadratic_form`$(f, \mathbf{x}_c, \lambda_1')$, which is built with the tight eigenvalue approximation λ_1'. Similarly, $f_{\mathbf{x}_c,2}$ is defined with the coarse eigenvalue approximation λ_2'. In the sequel, the input index i ($i = 1$ or 2) refers to one of the previous eigenvalue approximation methods (Section 6.2.1).

First, we select the control point $\mathbf{x}_1$ randomly to build either the estimator $f_{\mathbf{x}_1,1}$ or $f_{\mathbf{x}_1,2}$.

At each iteration of the loop (from Line 3 to Line 12), we compute an underestimator of f, using the sequence of control points $s = \mathbf{x}_1, \dots, \mathbf{x}_p$. We consider the supremum f_p of the p quadratic polynomials $f_{\mathbf{x}_1,i}, \dots, f_{\mathbf{x}_p,i}$ (Line 6) and get a lower bound of f_p using the semialgebraic optimization procedure `min_sa` at SDP relaxation order k. We compute a minimizer candidate $\mathbf{x}_{opt}$ and update the control points sequence, which allows to refine the approximation of f.

6.2.3 Numerical Results

Let $K = [0,1]^n$ and $r_1, \dots, r_n$ be positive random numbers. We consider the following instance of Problem (1.1.2):

$$\min_{\mathbf{x} \in [0,1]^n} f(\mathbf{x}) := \Big(\frac{1}{n} \sum_{i=1}^{n} \frac{4}{r_i^2} x_i (r_i - x_i) \Big)^{\lceil d/2 \rceil} \ . \tag{6.2.5}$$

```
Input: polynomial $f$, box $K$, $iter$, approximation method index $i$
Output: lower bound $m$
 1: $p := 0$
 2: $s := [\texttt{argmin}(\texttt{randeval}(f))]$          ▷ control point sequence initialization
 3: while $p \leqslant iter$ do
 4:                    ▷ At step $p$, the control point sequence is $s = \mathbf{x}_1, \ldots, \mathbf{x}_p$
 5:     For $c \in \{1, \ldots, p\}$: $f_{\mathbf{x}_c,i} := \texttt{build\_quadratic\_form}(t, \mathbf{x}_j, \lambda_i')$
 6:     $f_p := \max_{1 \leqslant c \leqslant p}\{f_{\mathbf{x}_c,i}\}$
 7:     Choose and SDP relaxation order $k$
 8:     $m := \texttt{min\_sa}(f_p, k)$
 9:     $\mathbf{x}_{opt} := \texttt{guess\_argmin}(f_p)$
10:     $p := p + 1$
11:     $s := s \cup \{\mathbf{x}_{opt}\}$
12: done
```

Figure 6.1: `pop_template_optim` : Quadratic Template Optimization Algorithm for POP

Notice that the range of the degree-d polynomial f is $[0, 1]$. We also emphasize the fact that f has no sparsity pattern.

Now we describe several experiments, performed on an Intel Core i5 CPU (2.40 GHz). Figure 6.2 displays the results of successive lower bounds computation using tight or coarse approximations of λ. For a given value of n, hundred instances of Problem (6.2.5) have been generated with $d = 4$. Here, we solve quadratically constrained nonconvex quadratic problems (Line8) at the first SDP relaxation order (Shor relaxation). The cardinal of the finite control points sequence corresponds to a given number of quadratic cuts (x-coordinate). The curves are obtained by averaging the resulting lower bounds for a given number of quadratic cuts.

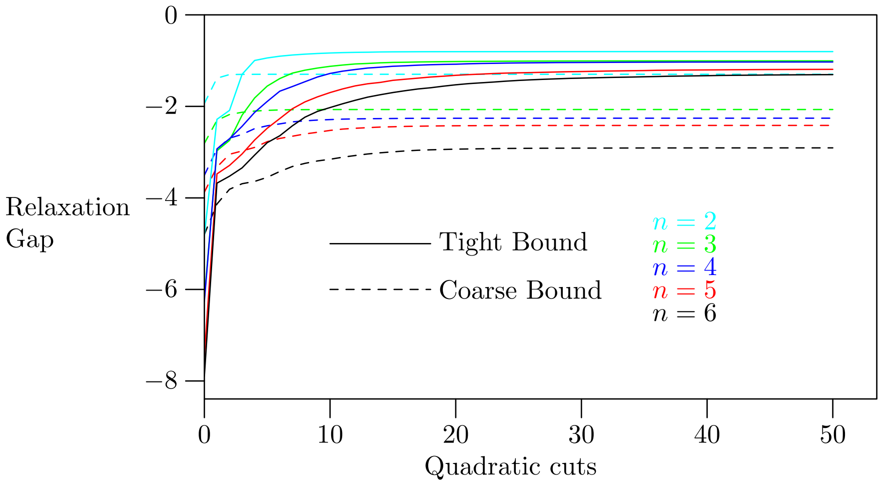
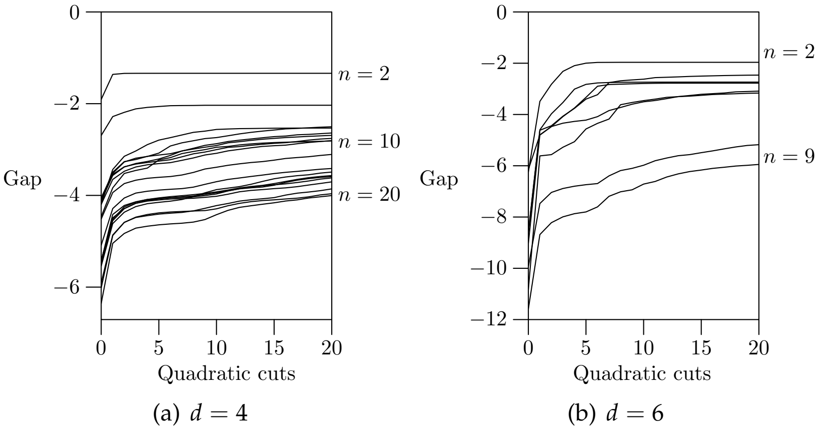


Figure 6.2: Comparison of lower bounds sequences for POP, using tight and coarse approximations of λ

The sequence of lower bounds converges towards a negative value. The relaxation gap (y-coordinate) between this value and the actual infimum of f (which is 0) is a consequence of keeping low the relaxation order. Notice also that the speed of convergence of this sequences decreases when n is larger. The tight approximation of minimal eigenvalues (SDP robust approach) yields more precise lower bounds for the POP. Besides, the computation cost of the coarse approximation is much cheaper. Indeed, for a given instance of Problem 6.2.5, the lower bound λ_2' computation algorithm is polynomial in n.

We next compare Problem (6.2.5) with quartic ($d = 4$) and sextic ($d = 6$) random polynomials. The lower bounds computation are displayed on Figure 6.3. They rely on coarse approximations of the Hessian matrix minimal eigenvalues. We explain this specific choice below.

For the quartic case, it takes 2 min to compute λ_1' when $n = 15$ and only 2 ms to obtain λ_2'. When $n = 30$, it takes about 10 ms to compute λ_2' with the second approach whereas it is impossible to get λ_1' (the semidefinite solver SDPA returns an out of memory exception). Moreover, one needs to compute $\widetilde{\mathcal{D}^2(f)}$ only once when using the second approach by choosing appropriate X and Y.



(a) $d = 4$ (b) $d = 6$

Figure 6.3: Lower bounds computation for medium-scale POP, using coarse approximations of λ

Table 6.1: Comparisons between the lower bound CPU time t and the coarse eigenvalue approximation CPU time t_2 for medium-scale POP after 20 quadratic cuts

n		2	3	4	5	6	7	8	9
$d = 4$	t (s)	2.02	2.12	2.75	3.03	3.72	5.75	5.87	7.01
	t_2/t (%)	19.4	21.9	30.3	42.2	48.7	56.7	61.2	68.5
$d = 6$	t (s)	1.88	2.48	4.11	6.52	12.43	21.80	47.34	108.19
	t_2/t (%)	18.7	31.5	42.4	72.0	83.3	87.8	93.0	96.3

In Table 6.1, we report some computation time results to get the lower bounds of Figures 6.3 after 20 quadratic cuts. We consider the total time t spent to get the lower bounds m and the time t_2 spent to obtain the first coarse approximation λ_2'.

For medium-scale POP ($n \lesssim 9$), the ratio t_2/t is closed to 1. The bottleneck of the algorithm `pop_template_optim` becomes the computation of the Hessian matrix $\widetilde{\mathcal{D}^2(f)}$ entries, which requires to solve $n(n+1)$ semidefinite relaxations of order at least $(k_{\min} - 1)$. By comparison, the size of the semidefinite relaxation of the nonconvex quadratic optimization problem (Line 8) is polynomial in the number of variables and quadratic cuts.

6.3 Polynomial Underestimators for Semialgebraic Functions

Given a box $K \subset \mathbb{R}^n$ and a semialgebraic leaf $f_{\text{sa}} : K \to \mathbb{R}$ of the abstract syntax tree of f, we consider an instance of Problem (1.1.2), where f_{sa} is involved. A common way to represent f_{sa} is to use its semialgebraic lifting, which leads to solve semialgebraic optimization problems with a possibly large number of lifting variables n_{lifting}. One way to reduce this number is to underestimate f_{sa} with a degree-d polynomial h_d, which should involve less variables than n_{lifting}. This section describes how to obtain such an h_d, which has the property to minimize the L_1-norm of the difference $(f_{\text{sa}} - h)$, over all degree-d polynomial underestimators h of f_{sa}.

We exploit a technique of Lasserre and Thanh [LT13], who showed how to obtain convex underestimators of polynomials. The method of [LT13] can be summarized as follows. Given a polynomial f_{pop}, a box K and a positive integer d, one can build a sequence of convex polynomials, which converge to the best convex degree-d polynomial underestimator of f_{pop}. This sequence is obtained with the optimal solutions of semidefinite programs.

Here, we derive a similar hierarchy of SDP relaxations, whose optimal solutions are the best (for the L_1-norm) degree-d (but possibly non convex) polynomial underestimators of t on K. We assume without loss of generality that K is hypercube $[0,1]^n$. By comparison with [LT13], the main difference is that the input is a semialgebraic function, rather than a polynomial.

6.3.1 Best Polynomial Underestimators of Semialgebraic Functions for the L_1 norm

Let $f_{\text{sa}} : K \to \mathbb{R}$ be a semialgebraic leaf of the abstract syntax tree of f and λ_n be the standard Lebesgue measure on $\mathbb{R}^n$, which is normalized so that $\lambda_n([0,1]^n) = 1$. We also introduce some auxiliary material. Define $g_1 := x_1(1-x_1), \ldots, g_n := x_n(1-x_n)$. The function f_{sa} has a basic semialgebraic lifting, thus there exist $p, s \in \mathbb{N}$, polynomials $g_{n+1}, \ldots, g_{n+s} \in \mathbb{R}[\mathbf{x}, z_1, \ldots, z_p]$ and a basic semialgebraic set K_{pop} defined by:

$$K_{\text{pop}} := \{(\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{n+p} : g_1(\mathbf{x}, \mathbf{z}) \geqslant 0, \ldots, g_m(\mathbf{x}, \mathbf{z}) \geqslant 0, g_{m+1}(\mathbf{x}, \mathbf{z}) \geqslant 0\} \enspace ,$$

such that the graph $\Psi_{f_{\text{sa}}}$ satisfies:

$$\Psi_{f_{\text{sa}}} := \{(\mathbf{x}, f_{\text{sa}}(\mathbf{x})) : \mathbf{x} \in K_{\text{sa}}\} = \{(\mathbf{x}, z_p) : (\mathbf{x}, \mathbf{z}) \in K_{\text{pop}}\} \enspace ,$$

with $m := n + s$ and $g_{m+1} := M - \|\mathbf{z}\|_2^2$, for some positive constant M obtained by adding bound constraints over the lifting variables $\mathbf{z}$ (to preserve Assumption 2.2). Define the polynomial $f_{\text{pop}}(\mathbf{x}, \mathbf{z}) := z_p$ and the total number of variables $n_{\text{pop}} := n + p$.

Consider the following optimization problem with optimal value m_d:

$$(P^{\mathrm{sa}}) \begin{cases} \min\limits_{h \in \mathbb{R}_d[\mathbf{x}]} & \displaystyle\int_K (f_{\mathrm{sa}} - h) d\lambda_n \\ \text{s.t.} & f_{\mathrm{sa}} - h \geqslant 0 \text{ on } K \ . \end{cases}$$

Lemma 6.8. *Problem (P^{sa}) has a degree-d polynomial minimizer h_d.*

Proof. Let us equip the vector space $\mathbb{R}_d[\mathbf{x}]$ of polynomials h of degree at most d with the norm $\|h\|_\infty := \sup_{|\boldsymbol{\alpha}| \leqslant d}\{|h_{\boldsymbol{\alpha}}|\}$.

Let H be the admissible set of Problem (P^{sa}). Observe that H is closed in the topology of the latter norm. Moreover, the objective function of Problem (P^{sa}) can be written as $\phi : h \in H \mapsto \|f_{\mathrm{sa}} - h\|_{L_1(K)}$, where $\| \cdot \|_{L_1(K)}$ is the norm of the space $L^1(K, \lambda_n)$. The function ϕ is continuous in the topology of $\| \cdot \|_\infty$ (for polynomials of bounded degree, the convergence of the coefficients implies the uniform convergence on every bounded set for the associated polynomial functions, and a fortiori the convergence of these polynomial functions in $L^1(K, \lambda_n)$). We claim that for every $t \in \mathbb{R}$, the sub-level set $S_t := \{h \in H \mid \phi(h) \leqslant t\}$ is bounded. Indeed, when $\phi(h) \leqslant t$, we have:

$$\|h\|_{L_1(K)} \leqslant \|f_{\mathrm{sa}} - h\|_{L_1(K)} + \|f_{\mathrm{sa}}\|_{L_1(K)} \leqslant t + \|f_{\mathrm{sa}}\|_{L_1(K)} \ .$$

Since on a finite dimensional vector space, all the norms are equivalent, there exists a constant $C > 0$ such that $\|h\|_\infty \leqslant C\|h\|_{L_1(K)}$ for all $h \in H$, so we deduce that $\|h\|_\infty \leqslant C(t + \|f_{\mathrm{sa}}\|_{L_1(K)})$ for all $h \in S_t$, which shows the claim. Since ϕ is continuous, it follows that every sublevel set of ϕ, which is a closed bounded subset of a finite dimensional vector space, is compact. Hence, the minimum of Problem (P^{sa}) is attained.

□

Let $QM(K_{\mathrm{pop}})$ be the quadratic module associated with $g_1, \ldots, g_{m+1}$:

$$QM(K_{\mathrm{pop}}) = \left\{ \sum_{j=0}^{m+1} \sigma_j(\mathbf{x}, \mathbf{z}) g_j(\mathbf{x}, \mathbf{z}) : \sigma_j \in \Sigma[\mathbf{x}, \mathbf{z}] \right\} \ .$$

By Proposition 2.7, the optimal solution h_d of (P^{sa}) is a maximizer of the following problem:

$$(P_d) \begin{cases} \max\limits_{h \in \mathbb{R}_d[\mathbf{x}]} & \displaystyle\int_{[0,1]^n} h \, d\lambda_n \\ \text{s.t.} & (f_{\mathrm{pop}} - h) \in QM(K_{\mathrm{pop}}) \ . \end{cases}$$

Let μ_d be the optimal value of (P_d). Then, one has $m_d = \int_K f_{\mathrm{sa}} \, d\lambda - \mu_d$. Note also that $\int_{[0,1]^n} h \, d\lambda_n = \int_{[0,1]^n} h(\mathbf{x}) \, d\lambda_n(\mathbf{x}) = \int_{[0,1]^{n+p}} h(\mathbf{x}, \mathbf{z}) \, d\lambda_{n+p}(\mathbf{x}, \mathbf{z}) =: \int_{[0,1]^{n+p}} h \, d\lambda_{n+p}$.

6.3.2 A Convergent Hierarchy of Semidefinite Relaxations

Let $\tilde{\omega}_0 := \lceil (\deg g_0)/2 \rceil, \ldots, \tilde{\omega}_{m+1} := \lceil (\deg g_{m+1})/2 \rceil$ and let k_0 be defined as follows:

$$k_0 := \max\{\lceil d/2 \rceil, \lceil (\deg f_{\mathrm{pop}})/2 \rceil, \tilde{\omega}_0, \ldots, \tilde{\omega}_{m+1}\} \ .$$

Now, we define the following sums of squares relaxation (P_{dk}) of (P_d), with optimal value μ_{dk}:

$$(P_{dk}) \begin{cases} \max_{h \in \mathbb{R}_d[\mathbf{x}], \sigma_j} & \int_{[0,1]^{n+p}} h \, d\lambda_{n+p} \\ \text{s.t.} & f_{\text{pop}}(\mathbf{x}, \mathbf{z}) = h(\mathbf{x}) + \sum_{j=0}^{m+1} \sigma_j(\mathbf{x}, \mathbf{z}) g_j(\mathbf{x}, \mathbf{z}), \ \forall (\mathbf{x}, \mathbf{z}) \ , \\ & \sigma_j \in \Sigma_{k - \tilde{\omega}_j}[\mathbf{x}, \mathbf{z}], \ 0 \leqslant j \leqslant m+1 \ , \end{cases}$$

with $k \geqslant k_0$.

This problem is an SDP program with variables $(h_d, \sigma_0, \ldots, \sigma_{m+1})$. Notice that $h = \sum_{\boldsymbol{\alpha} \in \mathbb{N}_d^n} h_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$. Then, the objective function of the optimization problem (P_{dk}) can be written $\sum_{\boldsymbol{\alpha} \in \mathbb{N}_d^n} h_{\boldsymbol{\alpha}} \gamma_{\boldsymbol{\alpha}}$, with $\gamma_{\boldsymbol{\alpha}} := \int_{[0,1]^n} \mathbf{x}^{\boldsymbol{\alpha}} d\mathbf{x}$ for all $\boldsymbol{\alpha} \in \mathbb{N}_d^n$.

Define the moment sequence $y \in \mathbb{R}_{2k}[\mathbf{x}, \mathbf{z}]^*$ associated with the Lebesgue measure λ_{n+p} on $[0,1]^{n+p}$, and in particular, let $y_{\text{pop}} := \int_{[0,1]^{n+p}} z_p \, d\lambda_{n+p}$ denote the entry of y corresponding to the moment arising from the variable z_p. Hence, the dual semidefinite program of (P_{dk}) can be defined as follows:

$$(P_{dk}^*) \begin{cases} \min_{y \in \mathbb{R}_{2k}[\mathbf{x}, \mathbf{z}]^*} & y_{\text{pop}} \\ \text{s.t.} & M_{k - \tilde{\omega}_j}(g_j y) \succcurlyeq 0, \quad 0 \leqslant j \leqslant m+1 \ , \\ & y_{\boldsymbol{\alpha}} = \gamma_{\boldsymbol{\alpha}}, \qquad\qquad \forall \boldsymbol{\alpha} \in \mathbb{N}_d^n \ . \end{cases}$$

Lemma 6.9. *For sufficiently large $k \geqslant k_0$, the semidefinite program (P_{dk}) has a maximizer h_{dk}.*

Proof. This is a special case of the proof of [LT13, Lemma 3.2], with $\mathbf{T}^*$ being the null operator, so that the dual program (P_{dk}^*) has a strictly feasible solution $y = \gamma$. Hence, there is no duality gap between (P_{dk}) and (P_{dk}^*). Moreover, $\int_{[0,1]^{n+p}} h \, d\lambda_{n+p}$ is bounded above by $\int_{[0,1]^{n+p}} f_{\text{pop}} \, d\lambda_{n+p}$, so the optimal value μ_{dk} is finite and the problem (P_{dk}) has an optimal solution. □

Let m_d be the optimal value of Problem (P^{sa}). As in [LT13], the optimal value of the hierarchy of semidefinite relaxations (P_{dk}) can become as close as desired to $m_d - f_{\text{sa}}^*$.

Theorem 6.10. *Let us call μ_{dk} the optimal value of the semidefinite program $(P_{dk}), k \in \mathbb{N}$. The sequence $(\int_K f_{sa} d\lambda - \mu_{dk})_{k \geqslant k_0}$ is non-increasing and converges to m_d. Moreover, if h_{dk} is a maximizer of (P_{dk}), then the sequence $(\|f_{sa} - h_{dk}\|_1)_{k \geqslant k_0}$ is non-increasing and converges to m_d. Furthermore, any accumulation point of the sequence $(h_{dk})_{k \geqslant k_0}$ is an optimal solution of Problem (P^{sa}).*

Proof. The proof is analogous with [LT13, Theorem 3.3]. □

Remark 6.11. Given $\mathbf{x}_c \in K$, the constraints of Problem P^{sa} can be reinforced so that f_{sa} and the underestimator h share the same value at $\mathbf{x}_c$. This can be formulated as follows:

$$(P_c^{\text{sa}}) \begin{cases} \min_{h \in \mathbb{R}_d[\mathbf{x}]} & \int_K (f_{\text{sa}} - h) d\lambda \\ \text{s.t.} & f_{\text{sa}} - h \geqslant 0 \text{ on } K \ , \\ & h(\mathbf{x}_c) = f_{\text{sa}}(\mathbf{x}_c) \ . \end{cases}$$

In this case, we solve the following variant of the SDP relaxation (P_{dk}^*):

$$(P^*_{dkc}) \begin{cases} \min\limits_{y \in \mathbb{R}_{2k}[\mathbf{x},\mathbf{z}]^*, y_c} & y_{\mathrm{pop}} + f_{\mathrm{sa}}(\mathbf{x}_c) y_c \\ \text{s.t.} & M_{k-\tilde{\omega}_j}(g_j y) \succcurlyeq 0, \quad 0 \leqslant j \leqslant m+1 \ , \\ & y_{\boldsymbol{\alpha}} = \gamma_{\boldsymbol{\alpha}} - \mathbf{x}_c^{\boldsymbol{\alpha}} y_c, \quad \forall \boldsymbol{\alpha} \in \mathbb{N}^n_d \ . \end{cases}$$

However, the relaxation (P^*_{dk}) involves fewer SDP variables than (P^*_{dkc}). Indeed, the equality constraints $y_{\boldsymbol{\alpha}} = \gamma_{\boldsymbol{\alpha}} (\boldsymbol{\alpha} \in \mathbb{N}^n_d)$ fix the values of $\binom{n+d}{d}$ variables for (P^*_{dk}).

6.3.3 Exploiting Sparsity and the Running Intersection Property

Let $I_1, \ldots, I_l$ be the cliques obtained from the chordal extension of the csp graph of the variables $\mathbf{x}$ (for more details, see Section 2.4). The collection $\{I_1, \ldots, I_l\}$ satisfies the running intersection property. We also add the l redundant additional constraints:

$$g_{m+q} := n_q M^2 - \sum_{i \in C_q} x_i^2 \geqslant 0, \ q = 1, \ldots, l \ , \tag{6.3.1}$$

set $m' = m + 1 + l$, define the compact semialgebraic set:

$$K'_{\mathrm{pop}} := \{\mathbf{x} \in \mathbb{R}^{n_{\mathrm{pop}}} : g_1(\mathbf{x}) \geqslant 0, \ldots, g'_m(\mathbf{x}) \geqslant 0\}.$$

Let F_k be the index set of variables which are involved in the polynomial g_k.

Now, we can define the sparse variant of the primal semidefinite relaxations P_{dk}, for $k \geqslant k_0$:

$$(P^{\mathrm{sparse}}_{dk}) \begin{cases} \max\limits_{h \in \mathbb{R}_d[\mathbf{x}], \sigma_j} & \displaystyle\int_{K'_{\mathrm{pop}}} h \, d\lambda_{\mathrm{pop}} \\ \text{s.t.} & f_{\mathrm{pop}}(\mathbf{x},\mathbf{z}) = h(\mathbf{x}) + \displaystyle\sum_{j=0}^{m'} \sigma_j(\mathbf{x},\mathbf{z}) g_j(\mathbf{x},\mathbf{z}), \ \forall (\mathbf{x},\mathbf{z}) \ , \\ & \sigma_j \in \Sigma[\mathbf{x},\mathbf{z},\mathbb{N}^{F_j}_{k-\tilde{\omega}_j}], \quad 1 \leqslant j \leqslant m' \ , \\ & \sigma_0 \in \displaystyle\sum_{1 \leqslant q \leqslant l} \Sigma[\mathbf{x},\mathbf{z},\mathbb{N}^{C_q}_k] \ . \end{cases}$$

The dual of $(P^{\mathrm{sparse}}_{dk})$ is the sparse variant of the dual semidefinite relaxations $(P_{dk})^*$:

$$(P^{\mathrm{sparse}}_{dk})^* \begin{cases} \min\limits_{y \in \mathbb{R}_{2k}[\mathbf{x},\mathbf{z}]^*} & y_{\mathrm{pop}} \\ \text{s.t.} & M_k(y, C_q) \succcurlyeq 0, \qquad 1 \leqslant q \leqslant l \ , \\ & M_{k-\tilde{\omega}_j}(g_j y, F_j) \succcurlyeq 0, \quad 1 \leqslant j \leqslant m' \ , \\ & y_{\boldsymbol{\alpha}} = \gamma_{\boldsymbol{\alpha}}, \qquad \forall \boldsymbol{\alpha} \in \mathbb{N}^{I_q}_d, 1 \leqslant q \leqslant l \ . \end{cases}$$

Remark 6.12 (Reducing the computational complexity). The semidefinite relaxation $(P^{\mathrm{sparse}}_{dk})^*$ involves at most $(\sum_{q=1}^l \binom{n_q+2k}{2k} - \sum_{q=1}^l \binom{n_q+d}{d})$ SDP moment variables. Let assume that the integers n_q are close to each other, in such a way that $n_q \simeq n_{\mathrm{pop}}/l$, then the number of variables is bounded by:

$$n^{\mathrm{sparse}}_{\mathrm{sdp}} := O\big(l(\frac{n_{\mathrm{pop}}}{l})^{2k} - l(\frac{n}{l})^d\big) \ .$$

Now we shall compare $n^{\mathrm{sparse}}_{\mathrm{sdp}}$ with the number of SDP variables involved in Problem $(P_{dk})^*$, which is $n_{\mathrm{sdp}} := O(n^{2k}_{\mathrm{pop}} - n^d)$. When the degrees of the polynomials involved in the semialgebraic lifting of f_{sa} are larger than d, then k_0 (resp. k) is larger than d and the computational cost saving is more significant.

Theorem 6.13. *Let μ_{dk}^{sparse} be the optimal value of the semidefinite program (P_{dk}^{sparse}), $k \in \mathbb{N}$. The sequence $(\int_K f_{sa} d\lambda - \mu_{dk}^{sparse})_{k \geqslant k_0}$ is non-increasing and converges to m_d.*

Proof. To prove the result, we use the running intersection property of the cliques and the redundant conditions defined in (6.3.1). For an analogous proof, we refer the reader to [Las, §4.1]. □

6.3.4 Numerical Experiments

We present the numerical results obtained when computing the best polynomial underestimators of semialgebraic functions for the L_1 norm, using the techniques presented in Section 6.3.3. Given a semialgebraic function f_{sa} defined on a compact semialgebraic set K_{sa} and an approximation degree d, we underestimate f_{sa} by a degree d polynomial obtained at the relaxation order k of (P_{dk}^{sparse}) (denoted by h_{dk}). This polynomial h_{dk} only depends on the variables $\mathbf{x}$. The "tightness" score $\|f_{\text{sa}} - h_{dk}\|_1$ evaluates the quality of the estimator h_{dk}, together with its lower bound μ_{dk}. The semidefinite relaxations of Problem (P^{sa}) have been implemented with OCAML (using SDPA), on an Intel Core i5 CPU (2.40 GHz).

Example 6.14. In Example 2.12, we considered the semialgebraic function $f_{\text{sa}} := \frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4x_1 \Delta \mathbf{x}}}$ and the set $K_{\text{sa}} := [4, 6.3504]^3 \times [6.3504, 8] \times [4, 6.3504]^2$. One can easily obtain the lower bound $m_2 = -0.618$ of f_{sa} at the second relaxation (resp. $m_3 = -0.445$ at the third relaxation), using two lifting variables that represent $\sqrt{4x_1 \Delta \mathbf{x}}$ and $\frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4x_1 \Delta \mathbf{x}}}$, as well as additional inequality constraints. However, when solving inequalities involving f_{sa}, one would like to solve POP that do not necessarily include these two lifting variables and the associated constraints. The Table 6.2 displays the tightness scores and the lower bounds of the estimators obtained for various values of the approximation degree d and the relaxation order k. Notice that μ_{dk} only bounds from below the actual infimum h_{dk}^* of the underestimator h_{dk}. It requires a few seconds to compute estimators at $k = 2$ against 10 minutes at $k = 3$, but one shall consider to take advantage of replacing f_{sa} by its estimator h_{63} to solve more complex POP.

Table 6.2: Comparing the tightness score $\|f_{\text{sa}} - h_{dk}\|_1$ and μ_{dk} for various values of d and k

d	k	Upper bound of $\|f_{\text{sa}} - h_{dk}\|_1$	μ_{dk}
2	2	0.8024	-1.171
	3	0.3709	-0.4479
4	2	1.617	-1.056
	3	0.1766	-0.4493
6	3	0.08826	-0.4471

Example 6.15. To illustrate the method, we consider the six variables function rad2_x issued from Flyspeck and its projection rad2_x$_2$ with respect to the first two coordinates (x_1, x_2) on the box $[4, 8]^2$ (we instantiated the remaining variables by the constant value 8):

$$\text{rad2\_x}_2 : (x_1, x_2) \mapsto \frac{-64x_1^2 + 128x_1x_2 + 1024x_1 - 64x_2^2 + 1024x_2 - 4096}{-8x_1^2 + 8x_1x_2 + 128x_1 - 8x_2^2 + 128x_2 - 512} .$$

In Figure 6.4 is displayed rad2_x$_2$ after scaling on $[0, 1]^2$ (the red surface), as well as the linear underestimator h_{13} (blue surface) and quadratic underestimator h_{23} (green surface). Both underestimators are obtained at the third relaxation order.

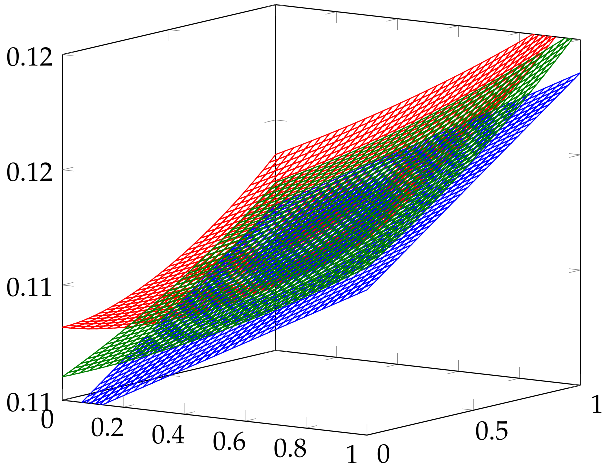


Figure 6.4: Linear and Quadratic Polynomial Underestimators for the rad2_x$_2$ function

6.4 The Template Optimization Algorithm

We now consider an instance of Problem (1.1.2). We assume that K is a box and we identify the objective function f with its abstract syntax tree t. The approximation algorithm `template_approx` is very close to the one defined in Chapter 5, but it can now limit the complexity of a given semialgebraic underestimator t^- (or overestimator t^+). Indeed, instead of taking the identity function for `reduce_lift`, we select a semialgebraic approximation procedure `build_template`.

Bounding the objective function by semialgebraic estimators.

Semialgebraic lower and upper estimators t^- and t^+ are computed by induction, following exactly the procedure `samp_approx` described in Section 5.3.1. For the sake of clarity, we briefly recall this inductive procedure:

When the tree is reduced to a leaf, *i.e.* $t \in \mathcal{A}$, it suffices to set $t^- = t^+ := t$.

When the root of the tree corresponds to a binary operation, then the semialgebraic estimators of the two children are composed using the function `compose_bop`.

Finally, if t corresponds to the composition of a transcendental (unary) function ϕ with a child c, we first bound c with semialgebraic functions c^+ and c^-. We bound ϕ from above and below by computing parabola at given control points (function `build_par`), thanks to the semiconvexity properties of ϕ. These parabola are composed with c^+ and c^- (function `samp_unary_approx` depicted in Figure 5.3).

These steps correspond to the part of the algorithm `template_approx` from Lines 1 to 12.

Reducing the complexity of semialgebraic estimators using templates

The semialgebraic estimators previously computed are used to determine lower and upper bounds of the function associated with the tree t, at each step of the induction. The bounds are obtained by calling the functions `min_sa` and `max_sa` respectively, which reduce the semialgebraic optimization problems to polynomial optimization problems by introducing extra lifting variables (see Section 2.4).

However, the complexity of solving the POPs can grow significantly because of the number n_{lifting} of lifting variables. If k denotes the relaxation order, the corresponding SDP problem Q_k indeed involve linear matrix inequalities of size $O((n + n_{\text{lifting}})^k)$ over $O((n + n_{\text{lifting}})^{2k})$ variables.

Consequently, this is crucial to control the number of lifting variables, or equivalently, the complexity of the semialgebraic estimators. For this purpose, we introduce the function `build_template`. This function can call two different procedures, as explained below.

1. The first method `build_quadratic_template` is an extension of the previous `pop_template_optim` algorithm (see Section 6.2, Figure 6.1). It allows to approximate of the tree t by means of suprema/infima of quadratic functions, when the number of lifting variables exceeds a user-defined threshold value $n_{\text{lifting}}^{\max}$. The algorithm is depicted in Figure 6.6. Using a heuristics, it first builds candidate quadratic polynomials q_j^- approximating t at each control point $\mathbf{x}_j$ (function `build_quadratic_form`, previously described). Since each q_j^- does not necessarily underestimate the function t, we then determine the lower bound m_j^- of the semialgebraic function $t^- - q_j^-$, which ensures that $q_j^- + m_j^-$ is a quadratic lower-approximation of t.

2. An alternative approach is the semidefinite relaxation P_{dk} described in Section 6.3.2: the semialgebraic function t^- is replaced with its degree-d polynomial underestimator h_{dk}. We denote this method by `build_l1_template`.

The returned semialgebraic expression (either $\max_{1 \leqslant j \leqslant r}\{q_j^- + m_j^-\}$ or h_{dk}) now generates only one lifting variable (representing max). Similarly, we can obtain a coarser upper-approximation by calling the procedure `build_template` on the opposite of the semialgebraic overestimator t^+.

Remark 6.16 (Dynamic choice of the control points). As in Section 5.3, the sequence s of control points is computed iteratively. We initialize the set s to a single point of K, chosen so as to be a minimizer candidate for t (e.g. with a local optimization solver). Calling the algorithm `template_approx` on the main objective function t yields an underestimator t^-. Then, we compute a minimizer candidate $\mathbf{x}_{opt}$ of the underestimator tree t^-. We add $\mathbf{x}_{opt}$ to the set of control points s. Consequently, we can refine dynamically our templates based max-plus approximations by iterating the previous procedure to get tighter lower bounds. This procedure can be stopped as soon as the requested lower bound is attained.

Example 6.17 (Modified Schwefel Problem). We illustrate our method with the function f from Example 1.4 and the finite set $\{135, 251, 500\}$ of control points. For each $i = 1, \dots, n$, consider the sub-tree $\sin(\sqrt{x_i})$. First, we represent each sub-tree $\sqrt{x_i}$ by a lifting variable y_i and compute $b_1 := \sqrt{135}$, $b_2 := \sqrt{251}$, $b_3 := \sqrt{500}$. Then, we

```
Input: tree $t$, box $K$, semidefinite relaxation order $k$, control points sequence $s = \{\mathbf{x}_1, \dots, \mathbf{x}_p\} \subset K$
Output: lower bound $m$, upper bound $M$, lower semialgebraic estimator $t_2^-$, upper semialgebraic estimator $t_2^+$
if $t \in \mathcal{A}$ then
    $t^- := t, t^+ := t$
else if $\texttt{bop} := \texttt{root}\ (t)$ is a binary operation with children $c_1$ and $c_2$ then
    $m_{c_i}, M_{c_i}, c_i^-, c_i^+ := \texttt{template\_approx}(c_i, K, k, s)$ for $i \in \{1, 2\}$
    $I_2 := [m_{c_2}, M_{c_2}]$
    $t^-, t^+ := \texttt{compose\_bop}(c_1^-, c_1^+, c_2^-, c_2^+, \texttt{bop}, I_2)$
else if $r := \texttt{root}(t) \in \mathcal{D}$ with child $c$ then
    $m_c, M_c, c^-, c^+ := \texttt{template\_approx}(c, K, k, s)$
    $I := [m_c, M_c]$
    $\text{par}^-, \text{par}^+ := \texttt{samp\_unary\_approx}(r, I, c, s)$
    $t^-, t^+ := \texttt{compose\_approx}(r, \text{par}^-, \text{par}^+, I, c^-, c^+)$
end
$t_2^- := \texttt{build\_template}(t, K, k, s, t^-), t^+ := -\texttt{build\_template}(t, K, k, s, -t^+)$
$\mathbf{x}^- := \texttt{vars}(t_2^-, K), \mathbf{x}^+ := \texttt{vars}(t_2^+, K)$
return $\texttt{min\_sa}(t_2^-, K, \mathbf{x}^-, k), \texttt{max\_sa}(t_2^+, K, \mathbf{x}^+, k), t^-, t^+$
```

Figure 6.5: `template_approx`

get the equations of $\text{par}_{b_1}^-$, $\text{par}_{b_2}^-$ and $\text{par}_{b_3}^-$ with $\texttt{build}_{\texttt{par}}$, which are three underestimators of the function sin on the real interval $I := [1, \sqrt{500}]$. Similarly we obtain three overestimators $\text{par}_{b_1}^+$, $\text{par}_{b_2}^+$ and $\text{par}_{b_3}^+$. Finally, we obtain the underestimator $t_{1,i}^- := \max_{j \in \{1,2,3\}} \{\text{par}_{b_j}^-(y_i)\}$ and the overestimator $t_{1,i}^+ := \min_{j \in \{1,2,3\}} \{\text{par}_{b_j}^+(y_i)\}$. To solve the modified Schwefel problem, we consider the following POP:

$$\begin{cases} \min\limits_{\mathbf{x} \in [1,500]^n, \mathbf{y} \in [1, \sqrt{500}]^n, \mathbf{z} \in [-1,1]^n} & -\sum_{i=1}^n (x_i + \epsilon x_{i+1}) z_i \\ \text{s.t.} & z_i \leqslant \text{par}_{b_j}^+(y_i), j \in \{1, 2, 3\}, i = 1, \dots, n \ , \\ & y_i^2 = x_i, i = 1, \dots, n \ . \end{cases}$$

Notice that the number of lifting variables is $2n$ and the number of equality constraints is n, thus we can obtain coarser semialgebraic approximations of f by considering the function $b \mapsto \sin(\sqrt{b})$ (see Figure 6.7). We get new estimators $t_{2,i}^-$ and $t_{2,i}^+$ of $\sin(\sqrt{x_i})$ with the functions `min_sa`, `max_sa` and `build_quadratic_form`. The resulting POP involves only n lifting variables. Besides, it does not contain equality constraints anymore, which improves in practice the numerical stability of the POP solver.

Convergence of the nonlinear template method

Let t_p^- and t_p^+ be the estimators obtained at the p^{th} iteration of the `template_optim` algorithm. Here, we suppose that the semialgebraic functions are underestimated with the procedure `build_l1_template` (SDP relaxation P_{dk} in Section 6.3.2).

Lemma 6.18 (Uniform convergence of the semialgebraic templates)**.** *The estimators sequences $(t_p^-)_p$ and $(t_p^+)_p$ uniformly converge to t on the box K.*

```
Input: tree $t$, box $K$, relaxation order $k$, control points sequence $s = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\} \subset K$,
    semialgebraic underestimator $t^-$
Output: lower semialgebraic estimator $t_2^-$
if the number of lifting variables exceeds $n_{\text{lifting}}^{\max}$ then
    for $\mathbf{x}_c \in s$ do
        $f_{\mathbf{x}_c}$ := build_quadratic_form($t, \mathbf{x}_j, \lambda$)
        $m_j^-$ := min_sa($t^- - q_j^-, k$)                    ▷ $q_j^- + m_j^- \leqslant t^- \leqslant t$
    done
    return $\max_{1 \leqslant j \leqslant r}\{q_j^- + m_j^-\}$
else
    return $t^-$
end
```

Figure 6.6: build_quadratic_template

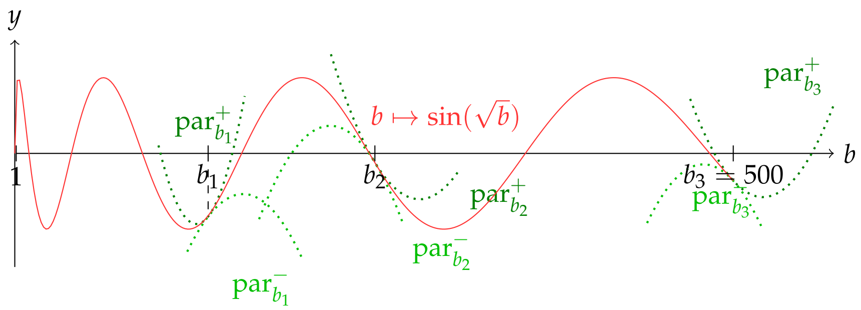


Figure 6.7: Templates based on Maxplus Semialgebraic Estimators for $b \mapsto \sin(\sqrt{b})$: $\max_{j\in\{1,2,3\}}\{\text{par}^-_{b_j}(x_i)\} \leqslant \sin\sqrt{x_i} \leqslant \min_{j\in\{1,2,3\}}\{\text{par}^+_{b_j}(x_i)\}$

Proof. We claim that Assumption 3.4 holds for the particular choice unary_approx = minimax_unary_approx and reduce_lift = build_l1_template. First, the uniform convergence of minimax_unary_approx comes from Theorem 5.3. Next, for sufficiently large relaxation order, the build_l1_template procedure returns the best (for the L_1 norm) degree-d polynomial underestimator of a given semialgebraic function (Theorem 6.10). Applying Proposition 3.8 yields the desired result. □

6.5 Benchmarks

6.5.1 Comparing three certification methods.

We next present numerical results obtained by applying the present template method to examples from the global optimization literature, as well as inequalities from the Flyspeck project. These experiments have been performed by interfacing our tool NLCertify with the SPARSEPOP solver [WKK$^+$08].

In each example, our aim is to certify a lower bound m of a function f on a box K. We use the algorithm template_optim, keeping the SOS relaxation order k sufficiently small to ensure the fast computation of the lower bounds. The algorithm template_optim returns more precise bounds by successive updates of the control points sequence s. Then, we perform a domain subdivision to reduce the relaxation gap (as explained in Section 4.3).

For the sake of comparison, we have implemented a template-free SOS method `ia_sos`, which coincides with the particular case of the algorithm `template_optim` in which $\#s = 0$ and $n_{\text{lifting}} = 0$. It computes the bounds of semialgebraic functions with standard SOS relaxations and bounds the univariate transcendental functions by interval arithmetic. We also tested the MATLAB toolbox algorithm `intsolver` [Mon09], which is based on the Newton interval method [HG83]. Experiments are performed on an Intel Core i5 CPU (2.40 GHz).

6.5.2 Global optimization problems.

Certification of lower bounds of non-linear problems

In Table 6.3, the *time* column indicates the total informal verification time, *i.e.* without the exact certification of the lower bound m with COQ. Each occurrence of the symbol "−" means that m could not be determined within one day of computation by the corresponding solver. We see that `ia_sos` already outperforms the interval arithmetic solver `intsolver` on these examples. However, it can only be used for problems with a moderate number of variables. The algorithm `template_optim` allows us to overcome this restriction, while keeping a similar performance (or occasionally improving this performance) on moderate size examples.

Notice that reducing the number of lifting variables allows us to provide more quickly coarse bounds for large-scale instances of *SWF*. We discuss the results appearing in the two last lines of Table 6.3. Without any box subdivision, we can certify a better lower bound $m = -967n$ with $n_{\text{lifting}} = 2n$ since our semialgebraic estimator is more precise. However the last lower bound $m = -968n$ can be computed twice faster by considering only n lifting variables, thus reducing the size of the POP described in Example 1.4. This indicates that the method is able to avoid the explosion for certain hard sub-classes of problems where a standard (full lifting) POP formulation would involve a large number of lifting variables.

Table 6.3: Comparison results for global optimization examples

Pb	n	m	`template_optim`					`ia_sos`		`intsolver`
			k	$\#s$	n_{lifting}	#boxes	time	#boxes	time	time
H3	3	-3.863	2	3	4	99	101 *s*	1096	247 *s*	3.73 *h*
H6	6	-3.33	2	1	6	113	102 *s*	113	45 *s*	> 4 *h*
MC	2	-1.92	1	2	1	17	1.8 *s*	92	7.6 *s*	4.4 *s*
ML	10	-0.966	1	1	6	8	8.2 *s*	8	6.6 *s*	> 4 *h*
PP	10	-46	1	3	2	135	89 *s*	3133	115 *s*	56 *min*
SBT	2	-190	2	3	2	150	36 *s*	258	0.6 *s*	57 *s*
SWF	10	$-430n$	2	6	$2n$	16	40 *s*	3830	129 *s*	18.5 *min*
	10^2	$-440n$	2	6	$2n$	274	1.9 *h*	$> 10^4$	> 10 *h*	−
$\epsilon = 0$	10^3	$-486n$	2	4	$2n$	1	450 *s*	−	−	−
	10^3	$-488n$	2	4	n	1	250 *s*	−	−	−
SWF	10^3	$-967n$	3	2	$2n$	1	543 *s*	−	−	−
$\epsilon = 1$	10^3	$-968n$	3	2	n	1	272 *s*	−	−	−

High-degree polynomial approximations.

We interfaced our tool with `Sollya` and performed some numerical tests. The minimax approximation based method is eventually faster than the template method for moderate instances. For the examples *H3* and *H6*, the speed-up factor is 8 when the function exp is approximated by a quartic minimax polynomial.

However, this approach is much slower to compute lower bounds of problems involving a large number of variables. It requires 57 times more CPU time to solve *SWF* ($\epsilon = 1$) with $n = 10$ by considering a cubic minimax polynomial approximation of the function $b \mapsto \sin(\sqrt{b})$ on a floating-point interval $I \supseteq [1, \sqrt{500}]$. These experiments indicate that a high-degree polynomial approximation is not suitable for large-scale problems.

6.5.3 Certification of various Flyspeck inequalities.

In Table 6.4, we present some test results for several non-linear Flyspeck inequalities. The information in the columns *time*, #boxes and n_{lifting} is the same as above. The integer $n_{\mathcal{D}}$ represents the number of transcendental univariate nodes in the corresponding abstract syntax trees.

Table 6.4: Results for Flyspeck inequalities using `template_optim` with $n = 6$, $k = 2$ and $m = 0$

Inequality id	$n_{\mathcal{D}}$	#s	n_{lifting}	#boxes	time
9922699028	1	4	9	47	241 *s*
9922699028	1	4	3	39	190 *s*
9922699028	1	1	1	170	1080 *s*
3318775219	1	2	9	338	26 *min*
7726998381	3	4	15	70	43 *min*
7394240696	3	2	15	351	1.8 *h*
4652969746_1	6	4	15	81	1.3 *h*
OXLZLEZ 6346351218_2_0	6	4	24	200	5.7 *h*

These inequalities are known to be tight and involve sum of arctan of correlated functions in many variables, whence we keep high the number of lifting variables to get precise max-plus estimators. However, some inequalities (e.g. 9922699028) are easier to solve by using coarser semialgebraic estimators. For instance, the first line ($n_{\text{lifting}} = 9$) corresponds to the algorithm described in [AGMW13b]. The second and third line illustrate our improved template method. For the former ($n_{\text{lifting}} = 3$), we do not use any lifting variables to represent square roots of univariate functions. For the latter ($n_{\text{lifting}} = 1$), we underestimate the semialgebraic function $\frac{\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}}$ with the `build_l1_template` procedure. Thus, we save two more lifting variables.

Part III

From Certified to Formal Global Optimization

Chapter 7

Formal Nonlinear Global Optimization

In the first two sections, we remind some basics on the Coq system. In particular, we focus on computational reflection (Section 7.1.3), which is used to handle formal polynomial optimization (Section 7.3). Finally, we explain how to derive formal bounds of semialgebraic functions (Section 7.4).

7.1 The Coq Proof Assistant

The aim of this section is to briefly recall some fundamental notions about the mechanisms of theorem proving within the Coq proof assistant. For more details on the topic, we recommend the documentation available in [BC04].

7.1.1 A Formal Theorem Prover

A formal proof assistant is a software which can represent mathematical definitions, theorems, propositions in a formal specification language. The Coq proof assistant provides such a language, called *Gallina*. A crucial point is that the system also has an abstract syntax to represent *proofs*. Because this syntax is cumbersome, proofs are generally built incrementally and interactively by the user using commands called proof tactics.

Once a proof is completed, it is checked by a specific part of Coq, called the *kernel*. The soundness of proofs is thus guaranteed by the consistency of the logical formalism implemented by Coq and the correctness of the kernel. In software engineering terms, the kernel is thus the *trusted computing base* of the whole system.

The Coq system is implemented, using the language (ML) OCaml. An interesting point is that *Gallina* is actually very similar to ML: it is a typed functional language, supporting a form of e.g. pattern-matching, and more generally a precisely defined notion of computation. There are of course differences between Coq's language and ML. On the one hand, the strength of the Coq type system enables to build *dependant types*. On the other hand, side-effects are not allowed and the programs written in Coq always terminate. Thus, Coq's language is essentially a dependently typed purely functional programming language.

7.1.2 Logic and Computation

In COQ, types of terms have also types. The type of types are specific constants called *sorts*. Because propositions (logical formulas) in Coq can be viewed as types, one such sort is `Prop`, the type of propositions. Another sort is `Type`, which is the type of computational types (and also, technically, of `Prop`). For historical reasons, `Type` is also sometimes written `Set`.

The COQ language allows to define complex datatypes, as inductive types. Objects of a given inductive type are built with *constructors*. A simple example is the type `bool` which has two constructors:

```
Inductive bool : Set :=
  | true  : bool
  | false : bool.
```

Since the two constructors `true` and `false` have no arguments, we say that `bool` is an enumerated type. A slightly more complex inductive type is the type of natural numbers:

```
Inductive nat : Set :=
 | O : nat
 | S : nat -> nat.
```

Here, the constructor `O` stands for the integer 0. Given a natural number n represented by `n`, `(S n)` stands for its successor $n+1$. For instance 2 is represented by `S (S O)` and 4 by `S (S (S (S O)))`. It is because the constructor `S` is recursive (its argument is itself a natural number) that this type has infinitively many inhabitants.

We have mentioned above that the language of COQ is at the same time a mathematical language and a programming language, which supports computation. This has consequences on the way proofs can be performed which are absolutely crucial for our work. We can illustrate this feature through a classic example.

In the traditional setting, arithmetic involves the following constant and axioms :

```
+ : nat -> nat -> nat
addO : forall n, O + n = n
addS : forall n m, (S n) + m = n + (S m)
```

The goal "2 + 2 = 4" can be written in COQ as:

```
Lemma two_plus_two : S (S O) + S (S O) = S (S (S (S O))).
```

By using the axiom `addS`, we go from the initial goal to `S O + S (S (S O))= S (S (S (S O)))`. By using `addS` one more time, the goal is rewritten as `O + S (S (S (S O)))= S (S (S (S O)))`. Then, we use the first axiom to obtain the goal `S (S (S (S O)))= S (S (S (S O)))`, that we solve by using the reflexivity of equality (`refl_equal` in COQ). Here the symbol "+" has no computational content.

The good way to define addition in COQ however is construct is as a computable function/program:

```
Fixpoint add n m : nat :=
 match n with
  | O    ⇒ m
  | S n' ⇒ add n' (S m)
```

In consequence, and this is no surprise, “2+2” (or `(plus (S (S O))(S (S O)))`) computes to 4 (or `(S (S (S (S O))))`). This means that these two objects are logically identified, and by congruence, so are the propositions “2+2=4” and “4=4”. In turn, the proposition “2+2=4” can be proved using a single deduction step (reflexivity), the rest by taken care of the computation mechanism.

The computational (Coq) proof is thus shorter than its traditional, purely deductive, counterpart: the formalism of Coq allows to replace some deduction steps unwritten by computation steps. Of course, this short-cut becomes much more important when proving “200+200=400”, or “2000+2000=4000”.

Note that several conversion rules define the internal notion of computation in Coq but for the sake of clarity, we do not detail these rules.

7.1.3 Computational Reflection

The programming language provided inside the formalism of Coq can be used in more sophisticated ways. In particular, it allows to build decision procedures or automatized reasoning, thus providing a way to prove classes of propositions in a systematic and efficient way. Because this involves formalizing a fragment of the logic in Coq itself, this technique is called *computational reflection* and was introduced in [BRB96] (see also [BM81] for details about reflection).

We illustrate this concept with arithmetic expressions, which can be encoded using the following inductive type:

```
Inductive nat_expr :=
 | Cst : nat → nat_expr
 | Var : id → nat_expr
 | Add : nat → nat → nat_expr.
```

where `id` is some data-type representing identifiers. Thus, the expression $x + (y + 1)$ is represented by `Add (Var x)(Add (Var y)(Cst (S O)))`.

Suppose now we have a decidable ordering over the type `id`. This allows us to define a normalization function over expressions:

```
norm_expr : expr → expr.
```

Basically, this function will flatten expressions, group the numerical constants and sort the identifiers. So that $x + (y + 1)$, $y + ((x + 0) + 1)$ and $(1 + y) + x$ are all normalized to $1 + (x + y)$ (that is `Add (Cst (S O))(Add (Var x)(Var y))`).

We can then prove that the values of expressions are unchanged by normalization. To make that statement precise, we need to define an interpretation, which maps expressions to their numerical values :

```
Fixpoint interp_expr e ( ivar : id -> nat ) : nat :=
 match e with
  | Cst n   ⇒ n
  | Var x   ⇒ ivar x
  | Add n m ⇒ interp_expr ivar n + interp_expr ivar m
 end.
```

Because of the symbolic part of expressions, corresponding to the case of the `Var` constructor, this interpretation is parametrized by an arbitrary interpretation of the identifiers.

Given two expressions e1 and e2, one can prove that if the normalization of e1 is equal to the normalization of e2, then e1 and e2 have also equal interpretations, whatever the interpretation of identifiers is. In Coq, one writes this statement as follows:

```
Lemma norm_expr_correct : forall e1 e2 I,
 norm_expr e1 = norm_expr e2 →
 interp_expr e1 I = interp_expr e2 I.
```

Proof. By induction on the structure of arithmetic expressions. □

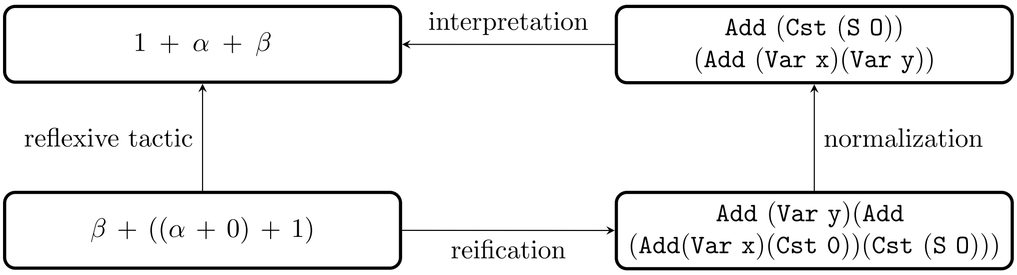
Figure 7.1 illustrates the reflection proof of the proposition "$\beta + ((\alpha + 0) + 1) = 1 + \alpha + \beta$", where α and β are arbitrary expressions of type nat. The trick is to have an interpretation function I which maps x to α and y to β. Then, the left-hand-side of the goal is reified into the expression Add (Var y)(Add (Add (Var x)(Cst 0))(Cst (S O))). Finally, the proof of the equality is:

```
norm_expr_correct (norm_expr (Add (Var y) (Add (Add (Var x)
    (Cst 0))
(Cst (S O))))) I
 (refl_equal (norm_expr (Add (Cst (S
  O)) (Add (Var x)(Var y))))).
```

One understand that such a proof is easy to generate for any given pair of expressions. Let us also point out that the verification of this proof essentially boils down to:

- normalizing the two expressions,
- verifying that the two normal forms are equal.

The rest, that is the proof of the correctness lemma is proved once for all.



$1 + \alpha + \beta$
interpretation
Add (Cst (S O)) (Add (Var x)(Var y))
reflexive tactic
normalization
$\beta + ((\alpha + 0) + 1)$
reification
Add (Var y)(Add (Add(Var x)(Cst 0))(Cst (S O)))

Figure 7.1: An illustration of Computational Reflection for Arithmetic Proofs

The tactics ring [GM05] and micromega [Bes07] are famous examples illustrating this methodology. The implementation of our SOS certificates checker (described in Section 7.3) is inspired from the development libraries of these two tactics. We use computational reflection to check the equality between polynomial expressions and Putinar certificates (Section 7.3.3).

7.2 Polynomial Arithmetic in Coq

As for proofs involving equalities of arithmetic expressions, one first needs to define the data-structure to represent polynomials as well as SOS certificates.

7.2.1 From Binary Integers to Arbitrary-size Rationals

In COQ, the set $\mathbb{N}_0$ of positive binary integers is represented by the following inductive type:

```
Inductive positive : Set :=
 | xI : positive → positive
 | xO : positive → positive
 | xH : positive.
```

The constructor `xH` stands for 1. When x is represented by the positive `x`, then `(xO x)` (resp. `(xI x)`) represents $2x$ (resp. $2x + 1$). For instance, `(xI xH)` stands for 3.

One can extend the positive integers with a zero (represented by the constructor `N0`) to obtain the set of natural numbers $\mathbb{N}$:

```
Inductive N : Set :=
 | N0   : N
 | Npos : positive → N.
```

The set $\mathbb{Z}$ of integers is defined as:

```
Inductive Z : Set :=
  | Z0   : Z
  | Zpos : positive → Z
  | Zneg : positive → Z.
```

However, a more efficient implementation of numbers is mandatory to check the correctness of SOS certificates. As a matter of fact, several theorem proving applications (see Remark 7.1) has required fast integers computation, which is currently available inside COQ.

Remark 7.1 (Checking primality certificates). In May 2013, Helfgott and Platt [HP13] (see also [Hel13]) presented a numerical verification of the ternary Goldbach Conjecture (up to $8.875 \cdot 10^{30}$). The approach described in [GTW06] could be applied to eliminate all uncertainties on this verification.

The type `C` that is used to represent the coefficients of polynomials can be instantiated by an efficient implementation of rational numbers `BigQ`. We give more details about this implementation in the sequel.

Tree representation of integers

The `BigN` library relies on a functional modular arithmetic developed by Grégoire and Théry [GT06]. Now, we recall the basic principles of this arithmetic.

From a given one-word set `w`, the two-words set `w2 w` (that depends on `w`) is defined as follows:

```
Inductive w2 (w : Set) : Set :=
 | W0 : w2 w
 | WW : w → w → w2 w.
```

The constructor `WW` takes two arguments, so we can arbitrary decide that the first (resp. last) one stands for high (resp. low) bits. The empty word constructor `W0` allows to manipulate binary trees, which are not systematically complete.

The type of numbers of height `n` is represented by the inductive type `word`:

```
Fixpoint word (w : Set) (n : nat) : Set :=
 match n with
  | O   ⇒ w
  | S n ⇒ w2 (word w n)
 end.
```

Then, one defines the arithmetic (as well as the comparison function) of the two-word set from the arithmetic of the single-word one. The interested reader can find more details about their implementation in [GT06]. One important feature is the logarithmic complexity for accessing digits. Thus, the order of bytes (the so-called "Endianness" choice) does not matter to perform arithmetic operations. This representation allows to split a number in two for free, which is particularly adapted for divide and conquer algorithms (Karatsuba-Ofman multiplication [KO63], recursive square root and division).

In the current state of the `BigN` library, `w` is the type `int31`. This type has a unique constructor `I31` that expects 31 arguments of type `digits`:

```
Inductive digits : Type := D0 | D1.

Inductive int31 : Type := I31 of digits &
& digits & digits & digits & digits & digits
& digits & digits & digits & digits & digits
& digits & digits & digits & digits & digits
& digits & digits & digits & digits & digits
& digits & digits & digits & digits & digits.
```

This definition of `int31` allows to use a mechanism for hardware efficient computation. Spiwack [Spi06] has modified the virtual machine to handle 31-bits integers natively, so that arithmetic operations are delegated to the CPU. Namely, one can benefit from the machine modular arithmetic (32 bits with `int31`) and perform native arithmetic operations on $\mathbb{Z}/31\mathbb{Z}$ inside COQ. Such developments made possible to deal with cpu-intensive tasks such as handling the proof checking of SAT traces [AGST10].

From integers to rational numbers

This type is built out of the type `BigN` (resp. `BigZ`) of arbitrary-size natural numbers (resp. integers) in base 2^{31} (binary trees with `int31` leaves). More generally, given a type of integers `ZZ.t` (e.g. `bigZ := BigZ.t`) for numerators and natural numbers `NN.t` (e.g. `bigN := BigN.t`) for denominators, the library `QMake` implements the type `t` of rationals. Notice that this inductive type allows multiple representations of the zero.

```
Inductive t :=
 | Qz : ZZ.t → t
 | Qq : ZZ.t → NN.t → t.
```

Let denote the zero of integers by:

```
Notation "0" := ZZ.zero.
```

The expression `(Qz z)` is interpreted as the integer `z` and `(Qq n d)` is interpreted as "n/d". The zero has the representations `(Qq 0 n)`, `(Qz 0)`, `(Qq n 0)` (e.g. 0, 0/4,

$2/0$, *etc*). The inverse function is defined on zero, thus is a total function. The specifications of the rationals library ensure that this definition does not lead to any mathematical inconsistencies.

For the ease of presentation, one defines:

```
Notation "p # q" := Qq p q.
Definition zero: t := Qz 0.
Definition one: t := Qz ZZ.one.
```

By using the comparison function of `NN`, one can check whether a reduced fraction n/d is an integer and return the corresponding rational number. Notice that the `check_int` procedure makes a comparison test but does not return a boolean value.

```
 Definition check_int n d :=
  match NN.compare NN.one d with
   | Lt ⇒ n # d
   | Eq ⇒ Qz n
   | Gt ⇒ zero
  end.
```

When $1 < d$, `check_int` returns the fraction n/d. When $d = 1$, it returns the integer n. Otherwise, $d = 0$ and the pair of arguments represents 0.

The normalization function is defined from `check_int` and the greatest common divisor function gcd of the natural numbers:

```
Definition norm n d : t :=
 let gcd := NN.gcd (Zabs_N n) d in
  match NN.compare NN.one gcd with
   | Lt ⇒ check_int (ZZ.div n (Z_of_N gcd)) (NN.div d gcd)
   | Eq ⇒ check_int n d
   | Gt ⇒ zero
  end.
```

First, one computes $g := \gcd(|n|, d)$. When $1 < g$, the procedure returns the result of `check_int` on (n/g) and (d/g). When n and d are coprime, the result is the same than `check_int`. Otherwise, both numbers n and d are zero.

Now, one can provide an unique representation of rationals, using irreducible fractions:

```
Definition red (x : t) : t :=
 match x with
  | Qz z  ⇒ x
  | n # d ⇒ norm n d
 end.
```

We emphasize the fact that the equality is decidable on rationals, which is crucial for checking our certificates. Indeed, it allows to prove the equality between polynomials built on top of rationals (Section 7.3.3) then to assert the nonnegativity of nonlinear expressions (Section 7.3.4).

7.2.2 The polynomial ring structure

In order to prove the equality between polynomials and sums of squares certificates, we build a so-called "customized" (see the documentation of the `ring` tactic[1]) polynomial ring. This requires to provide a type of coefficients `C` for polynomial expressions (`bigQ` in our current implementation), as well as a ring morphism `IQR`.

Injecting `BigQ` in `R`

The morphism `IQR` injects arbitrary-size rationals into the carrier type `R`. In our development, `R` is the type of Coq classical real numbers.

We often use the following notation in the sequel:

```
Notation "[ c ]" := IQR c.
```

We denote real addition, subtraction, multiplication and division by the respective notations `+`, `-`, `*`, `/`. To avoid confusions, we add the symbol `!` to denote the same operations on `BigQ`.

Let `==` be the propositional equality on real numbers and `?=!` be the boolean equality on rationals.

The injection of arbitrary-size rational constants in `R` relies on the injection of integers (denoted by `IZR`) in `R`. The arbitrary-size rational `q` is first converted into a number of type `Q` (rationals of the Coq standard library) with the function `to_Q`, thus one can apply the procedure `IZR`:

```
Definition IQR (q : bigQ) : R :=
 match (BigQ.to_Q q) with
   n # d ⇒ IZR n / IZR (Zpos d)
 end.
```

The morphism `IQR` has to satisfy the following properties:

```
(* IQR preserves the neutral elements of the ring *)
iqr0    : [zero] == 0;
iqr1    : [one] == 1;
(* IQR respects the operations of addition, subtraction and
   multiplication *)
iqr_add : forall x y, [x +! y] == [x]+[y];
iqr_sub : forall x y, [x -! y] == [x]-[y];
iqr_mul : forall x y, [x *! y] == [x]*[y];
iqr_opp : forall x, [-!x] == -[x];
(* IQR satisfies the following correctness lemma *)
iqr_eqb_eq : forall x y, x?=!y = true → [x] == [y].
```

Two representations for polynomials

For the `ring` tactic [GM05], two representations of polynomials are used, namely `PExpr` and `PolC`. The former is for uninterpreted ring expressions, while the latter is for uninterpreted normalized polynomial expressions.

One defines the inductive type `PExpr`:

[1] http://coq.inria.fr/refman/Reference-Manual027.html

```
 Inductive PExpr : Type :=
  | PEc   : bigQ → PExpr
  | PEX   : positive → PExpr
  | PEadd : PExpr → PExpr → PExpr
  | PEsub : PExpr → PExpr → PExpr
  | PEmul : PExpr → PExpr → PExpr
  | PEopp : PExpr → PExpr
  | PEpow : PExpr → N → PExpr.
```

Moreover, the sparse Horner representation is the normal form:

```
 Inductive PolC : Type :=
  | Pc   : bigQ → PolC
  | Pinj : positive → PolC → PolC
  | PX   : PolC → positive → PolC → PolC.
```

The three constructors `Pc`, `Pinj` and `PX` satisfy the following conditions:

1. The polynomial `(Pc c)` is the constant polynomial that evaluates to `[c]`.

2. The polynomial `(Pinj i p)` is obtained by shifting the index of `i` in the variables of `p`. In other words, when `p` is interpreted as the value of the $(n - i)$ variables polynomial $p(x_1, \dots, x_{n-i})$, then one interprets `(Pinj i p)` as the value of $p(x_i, \dots, x_n)$.

3. Let `p` (resp. `q`) represents p(resp. $q(x_1, \dots, x_{n-1})$). Then `(PX p j q)` evaluates to $px_1^j + q(x_2, \dots, x_n)$.

We define the zero polynomial as well as the polynomial whose values are all equal to one:

```
Definition p0 := Pc zero.
Definition p1 := Pc one.
```

Example 7.2. Consider the polynomials $x_{12} := x_1 - x_2$ and $q_{12} := x_1 - 2x_2$. The rational number 2 is encoded by `c2`. The Horner normal forms that represent x_{12} and q_{12} are:

```
Definition x_12 := PX p1 xH (PX (Pc (-! one)) xH p0).
Definition q_12 := PX p1 xH (PX (Pc (-! c2)) xH p0).
```

Then, the polynomial $p_{12} := x_1^2 - 2x_1x_2 + x_2^2 = x_1(x_1 - 2x_2) + x_2^2 = q_{12}x_1 + x_2^2$ is represented by:

```
Definition p_12 := PX q_12 xH (PX p1 (x0 xH) p0).
```

The boolean equality test `Peq` between two normalized polynomials relies on the positive (resp. `BigQ`) boolean equality test `Pos.eqb` (resp. `?=!`):

```
Fixpoint Peq (p p' : PolC)  {struct p'} : bool :=
 match p, p' with
  | Pc c    , Pc c'      ⇒ c ?=! c'
  | Pinj j q, Pinj j' q' ⇒ Pos.eqb j j' && Peq q q'
  | [...]
 end.
Notation "?==" := Peq.
```

One also defines the basic operations on sparse Horner polynomials. The addition between two polynomials is `padd : PolC → PolC → PolC`. The multiplication of a polynomial by a rational constant is `pmulC : PolC → bigQ → PolC` and the multiplication of two polynomials is `pmul : PolC → PolC → PolC`. The square of `(p : PolC)` is given by `(psquare p)`.

The function `norm` converts a polynomial expression to a normalized form:

```
Definition norm (pe : PExpr) : PolC :=
 match pe with
  | PEc c ⇒ Pc c
  [...]
 end.
```

The environment function `Env := positive → R` binds positive integers to the polynomial real variables. One also needs the `jump` and `tail` functions:

```
Definition jump (j : positive) (l : Env) : Env :=
  fun x ⇒ l (j + x).
Definition tail (l : Env) : Env := jump xH l.
```

The function `PolCeval` (resp. `PEeval`) maps a sparse Horner polynomial (resp. a polynomial expression) to the carrier type `R`:

```
Fixpoint PolCeval (l : Env) (P:PolC) : R :=
 match P with
  | Pc c      ⇒ [c]
  | Pinj j Q ⇒ PolCeval (jump j l) Q
  | PX P i Q ⇒ PolCeval l P * (l xH) ^ i
               + PolCeval (tail l) Q
 end.

Fixpoint PEeval (l : Env) (pe : PExpr) : R :=
 match pe with
  | PEc c          ⇒ [c]
  | PEadd pe1 pe2 ⇒ (PEeval l pe1) + (PEeval l pe2)
  [...]
 end.
```

The following correctness lemma is analogous with `iqr_eqb_eq`:

```
Lemma pol_eqb_eq p1 p2 : (norm p1 ?== norm p2) = true →
   forall (l : Env), PEeval l p1 == PEeval l p2.
```

For instance, one can prove that a polynomial expression `pe` always evaluates to zero by checking the equivalence between `(norm pe)` and `p0`. We next detail how to use this lemma to verify the correctness of SOS certificates.

7.3 Formal Polynomial Optimization

Let us recall the scaled version of the polynomial optimization problem (2.2.1):

$$\begin{cases} \inf_{\mathbf{x}\in[0,1]^n} & f_{\text{pop}}(\mathbf{x}) \ , \\ \text{s.t.} & g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0 \ . \end{cases}$$

Following the procedure described in Section 2.5, we extract a rational certificate $(\tilde{\mu}_k, \tilde{\sigma}_0, \ldots, \tilde{\sigma}_m, \epsilon_{\text{pop}}, \epsilon^*_{\text{pop}})$.

By definition, this certificate satisfies the following, for all $\mathbf{x} \in [0,1]^n$:

$$f_{\text{pop}}(\mathbf{x}) - (\tilde{\mu}_k + \epsilon^*_{\text{pop}}) := \sum_{j=0}^{m} \tilde{\sigma}_j(\mathbf{x}) g_j(\mathbf{x}) + (\epsilon_{\text{pop}}(\mathbf{x}) - \epsilon^*_{\text{pop}}) \ . \tag{7.3.1}$$

Here, we consider the following goal, for all $\mathbf{x} \in [0,1]^n$:

$$g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0 \Rightarrow f_{\text{pop}}(\mathbf{x}) \geqslant \tilde{\mu}_k + \epsilon^*_{\text{pop}}.$$

We solve this goal by proving first the nonnegativity of $\sum_{j=0}^{m} \tilde{\sigma}_j(\mathbf{x}) g_j(\mathbf{x})$ and $(\epsilon_{\text{pop}}(\mathbf{x}) - \epsilon^*_{\text{pop}})$ over $[0,1]^n$. Then, we use computational reflection to check the correctness of the above equality (7.3.1), which allows to conclude.

7.3.1 Encoding Putinar Certificates

Sums of squares can be decomposed as follows (see (2.5.2)):

$$\tilde{\sigma}_j := \sum_{i=1}^{r_j} \lambda_{ij} v_{ij}^2, j = 0, \ldots, m \ .$$

In Coq, we index the rational coefficients λ_{ij} and the inequalities g_j using positive integers. We encode an SOS block using a tuple composed of a positive index and a sparse horner polynomial. An SOS σ is represented by a sequence of SOS blocks `[(1, v_1); ... ; (r, v_r)]`.

```
Definition Lambda_idx := positive.
Definition Ineq_idx := positive.
Definition Sos_block :=  (Lambda_idx * PolC).
Definition Sos := seq Sos_block.
Definition Putinar_psatz := seq (Sos * Ineq_idx).
```

The notation "λ_{ij}" is abusive in our setting, as it does not refer to the eigenvalues of the SDP matrix. A Putinar certificate summand can be defined as a tuple composed of an SOS σ_j and an inequality index j. Then, a Putinar certificate is encoded using a sequence of such tuples: `[(sos_0, 1); ... ; (sos_m, m + 1)]`.

Our certificate also contains a map `lambda : Lambda_idx` → `bigQ` that associates positive indexes to their rational weights. Similarly, the map `ineq : Ineq_idx` → `PolC` returns the polynomials $g_j (j = 0, \ldots, m)$.

Interpretation of Putinar Certificates

SOS blocks are converted into Horner polynomials with `Sos_block_toPolC`:

```
Definition Sos_block_toPolC lambda sos_block :=
   let (lambda_idx, v) := sos_block in
    pmulC  (psquare v) (lambda lambda_idx).
```

Then, Putinar certificates can be interpreted as sparse Horner polynomials. Given a type `A` and an interpretation function `interp : A` → `PolC`, one defines recursively the interpretation of a sequence of type `A` objects:

```
Fixpoint foldr_psatz (s : seq A) : PolC :=
 match s with
  | [::]    ⇒ p0
  | [::hd] ⇒ interp hd
  | x::tl   ⇒ padd (interp x) (foldr_psatz tl)
 end.
```

Hence, sums of squares are converted to polynomials, by instantiating `A` with `Sos_block` and `interp` with `Sos_block_toPolC`.

```
Definition Sos_toPolC lambda sos :=
  foldr_psatz padd (Sos_block_toPolC lambda) sos.

Definition summand_toPolC ineq lambda summand :=
  let (sos, ineq_idx) := summand in
   pmul (Sos_toPolC lambda sos) (ineq ineq_idx).

Definition Psatz_toPolC ineq lambda s :=
  foldr_psatz padd (summand_toPolC ineq lambda) s.
```

Finally, Putinar certificates are converted to polynomials by using, again, the function `foldr_psatz`. In this case, `A` is the product type `(Sos * Ineq_idx)` and `interp` is `summand_toPolC`.

Nonnegativity of Putinar Certificates

To prove that a Putinar certificate `s` is nonnegative, one needs to verify the two conditions:

1. All the coefficients `(lambda lambda_idx)` involved in `s` are nonnegative.

2. All the polynomials `(ineq ineq_idx)` have nonnegative evaluation. This requires to match them with the hypotheses $g_1 \geqslant 0, \ldots, g_m \geqslant 0$.

```
Lemma Putinar_Psatz_Nonnegative l lambda ineq s :
  forall lambda_idx, 0 [<=] lambda lambda_idx →
  forall ineq_idx, 0 <= PolCeval l (ineq ineq_idx) →
  0 <= PolCeval l (Psatz_toPolC ineq lambda s).
```

Proof. By induction on the structure of the Putinar certificate `s`. □

7.3.2 Bounding the Polynomial Remainder

Recall that a coarse lower bound of a polynomial $\epsilon_{\text{pop}} := \sum \epsilon_\alpha \mathbf{x}_\alpha$ over $[0,1]^n$ can be obtained as follows:

$$\epsilon^*_{\text{pop}} := \sum_{\epsilon_\alpha \leqslant 0} \epsilon_\alpha \ .$$

Let `cmin` be the procedure which computes the minimum of two rational numbers. The recursive function `lower_bound_0_1` computes the rational lower bound of a sparse Horner polynomial `eps_pol` on $[0,1]^n$:

```
Fixpoint lower_bound_0_1 (eps_pol : PolC) : bigQ :=
 match eps_pol with
  | Pc c     ⇒ cmin c zero
  | Pinj _ p ⇒ lower_bound_0_1 p
  | PX p _ q ⇒ lower_bound_0_1 p +! lower_bound_0_1 q
 end.
```

Using the function `vars : PolC → seq positive` which returns the variables of a given polynomial, the predicate "$\forall \mathbf{x} \in [0,1]^n$" can be written:

```
forall i, i \in vars eps_pol → 0 <= l i ∧ l i <= 1
```

It allows to state the remainder inequality of Lemma (2.5.3) inside Coq:

```
Lemma remainder_lemma l eps_pol :
  (forall i, i \in vars eps_pol → 0 <= l i ∧ l i <= 1) →
  [lower_bound_0_1 eps_pol] <= PolCeval l eps_pol.
```

Proof. By induction on the structure of sparse Horner polynomials. □

7.3.3 Checking Polynomial Equalities

The procedure which checks the polynomial equality (7.3.1) relies on *computational reflexion*: logical deduction steps are replaced by a conversion rule. Here, we apply the reflexive tactic `ring`, by using the customized polynomial ring described in Section 7.2.2 and the encoding of Putinar certificates of Section 7.3.1. Example 7.3 illustrates the behaviour of this customized ring with computable interpretation and normalization functions.

Example 7.3. We consider the polynomial $p_{12} := x_1^2 - 2x_1x_2 + x_2^2$ defined in Example 7.2 and we prove the goal "$p_{12} = (x_1 - x_2)^2$". This is a simple case of (7.3.1) with $\epsilon_{\text{pop}} = 0, \epsilon^*_{\text{pop}} = \tilde{\mu}_k = 0, m = 0, g_0 = 1, \sigma_0 = (x_1 - x_2)^2$. One defines the following environment function:

```
Definition env (x:positive) : R :=
 match x with
  | xH    ⇒ x1
  | x0 xH ⇒ x2
  | _     ⇒ 0
 end.
```

As depicted in Figure 7.2, the right-hand-side expression $(x_1 - x_2)^2$ is reified into the expression `[([(one, x_12)], xH)]` of type `Putinar_psatz` (`xH` stands for the inequality index, `one` for the rational one and `x_12` represents $x_1 - x_2$). Then, it is normalized into the sparse Horner polynomial `p_12` of type `PolC`. Finally, the interpretation `(PolCeval env p_12)` returns the left-hand-side polynomial expression p_{12}.

7.3.4 Checking Polynomial Nonnegativity

The main correctness lemma

Our main correctness lemma `Putinar_Psatz_correct` relies on the following boolean function `pol_checker`, which computes the coarse lower bound ϵ^*_{pop} of ϵ_{pop} (repre-

$$\begin{tikzcd}
\boxed{x_1^2 - 2x_1x_2 + x_2^2} & \boxed{\texttt{p\_12}} \arrow[l, "\text{interpretation}"'] \\
\boxed{(x_1 - x_2)^2} \arrow[u, "\text{reflexive tactic}"] \arrow[r, "\text{reification}"'] & \boxed{[([(\texttt{one}, \texttt{x\_12})], \texttt{xH})]} \arrow[u, "\text{normalization}"']
\end{tikzcd}$$

Figure 7.2: An illustration of computational reflection

sented by r) and verifies equality 7.3.1:

```
Definition pol_checker obj r ineq lambda s : bool :=
  norm obj ?== padd (Psatz_toPolC ineq lambda s)
                    (psubC r (lower_bound_0_1 r)).

Lemma Putinar_Psatz_correct l obj r ineq lambda s :
  (forall i, i \in vars r → 0 <= l i ∧ l i <= 1 ) →
   forall lambda_idx, 0 [<=] lambda lambda_idx →
   forall ineq_idx, 0 <= PolCeval l (ineq ineq_idx) →
   pol_checker obj r ineq lambda s = true →
     0 <= PEeval l obj.
```

Proof. Assuming that `pol_checker = true`, one can apply Lemma `pol_eqb_eq` and obtain the following equality:

```
PEeval l obj = PolCeval l (Psatz_toPolC ineq lambda s)
             + PolCeval l r - [lower_bound_0_1 r].
```

The nonnegativity of `(Psatz_toPolC ineq lambda s)` comes from the application of Lemma `Putinar_Psatz_Nonnegative`. Then, Lemma `remainder_lemma` implies the nonnegativity of `(PolCeval l r - [lower_bound_0_1 r])`. □

Benchmarks

We tested our formal verification procedure on the following polynomial inequalities ($K = [4, 6.3504]^3 \times [6.3504, 8] \times [4, 6.3504]^2$, see Appendix A for the definition of $\Delta\mathbf{x}$):

- *POP1* : $\forall \mathbf{x} \in K, \partial_4 \Delta\mathbf{x} \geqslant -41$.
- *POP2* : $\forall \mathbf{x} \in K, \Delta\mathbf{x} \geqslant 0$.

We also compared our time results with the `micromega` tactic. This tactic, available with Coq, uses a more general version of the Positivstellensatz (due to Stengle) to find witnesses of unfeasibility of a set of polynomial constraints [Bes07]. The tactic relies on the external SDP solver Csdp. To deal with the numerical errors of Csdp, a projection algorithm is performed in such a way that $\epsilon^*_{\text{pop}} = 0$. Thus, the procedure returns a rational SOS certificate that matches exactly $f_{\text{pop}} - \tilde{\mu_k}$.

Table 7.1 indicates that our tool outperforms the `micromega` decision procedure, thanks to the sparse variant of Lasserre relaxation and a simpler projection method. Notice that for *POP2*, we considered the projection of $\Delta\mathbf{x}$ with respect to the first n coordinates on the box K (fixing the other variables to 6.3504). The symbol "$-$" means that the inequality could not be checked by `micromega` within one hour of computation. A preliminary phase consists in scaling the POP to apply the correctness Lemma `Putinar_Psatz_Nonnegative`.

Table 7.1: Comparing our formal POP checker with `micromega`

Problem	n	`NLCertify`	`micromega`
POP1	6	$0.08\,s$	$9\,s$
POP2	2	$0.09\,s$	$0.36\,s$
	3	$0.39\,s$	–
	6	$13.2\,s$	–

Remark 7.4 (Improving `micromega`). A part of our work could form the basis of an improved version of `micromega`:

1. The modularity of the tactic `micromega` allows to call the external library described in [MC11] to handle degenerate situations (when the SDP formulation of the POP is not strictly feasible). We interfaced our solver with this library and successfully solved small size instances of POP. The formal verification is much slower than the informal procedure (see the results presented in Section 6.5). As an example, for the *MC* problem, it is 36 times slower to generate exact SOS certificates and 13 times slower to prove its correctness in COQ. This track was not further pursued, since computation were faster with the algorithms described in this section. However, we mention that this library could handle the formal verification of unconstrained POP, which is not possible with our current implementation.

2. The relaxation based on Stengle Positivstellensatz can be replaced by the sparse variant described in Section 2.4.

As explained in the sequel, the formal bounds obtained for POP are mandatory to certify semialgebraic optimization problems.

7.4 Beyond Polynomial Inequalities

This section describes how to solve semialgebraic inequalities inside COQ (see (2.3.1)):

$$\forall \mathbf{x} \in K, f_{\mathrm{sa}}(\mathbf{x}) \geqslant f_{\mathrm{sa}}^* \ , \qquad (7.4.1)$$

where $f_{\mathrm{sa}} \in \mathcal{A}$ and $K := \{\mathbf{x} \in \mathbb{R}^n : g_1(\mathbf{x}) \geqslant 0, \dots, g_m(\mathbf{x}) \geqslant 0\}$ is a basic semialgebraic set.

A proper encoding of both intervals and semialgebraic expressions is required to describe the certificates of such inequalities.

7.4.1 Intervals

Intervals are encoded using two rational coefficients:

```
Inductive itv : Type := | Itv : bigQ → bigQ → itv.
```

For instance, the interval $[0,1]$ is represented by (Itv zero one). We define several routines to handle interval arithmetic. For instance, (itv_without_0 (Itv i1 i2)) verifies that 0 belongs to $[i_1, i_2]$.

7.4.2 Semialgebraic expressions

Semialgebraic functions are represented by the inductive type Fsa:

```
Inductive Fsa : Type :=
  | Poly   : PExpr → itv → Fsa
  | Fdivp  : PExpr → Fsa → itv → Fsa
  | Fdiv   : Fsa → Fsa → itv → Fsa
  | Fmul   : Fsa → Fsa → itv → Fsa
  | Fsub   : Fsa → Fsa → itv → Fsa
  | Fopp   : Fsa → itv → Fsa
  | Fadd   : Fsa → Fsa → itv → Fsa
  | Fsqrt  : Fsa → itv → Fsa.
```

The interpretation of semialgebraic functions relies on the polynomial expressions evaluation procedure PEeval:

```
Fixpoint Feval (l:Env) fsa : R :=
  match fsa with
   | Poly pe _     ⇒ PEeval l pe
   | Fdivp pe f _  ⇒ (PEeval l pe) / (Feval l f)
   | Fdiv f1 f2 _  ⇒ (Feval l f1) / (Feval l f2)
   | Fmul f1 f2 _  ⇒ (Feval l f1) * (Feval l f2)
   | Fsub f1 f2 _  ⇒ (Feval l f1) - (Feval l f2)
   | Fopp f1 _     ⇒ - (Feval l f1)
   | Fadd f1 f2 _  ⇒ (Feval l f1) + (Feval l f2)
   | Fsqrt f1 _    ⇒ rsqrt (Feval l f1)
  end.
```

An fsa object is a certificate obtained from one of the informal certification procedures described in Part II (e.g. template_optim). The function get_itv returns an interval, which should enclose the range of values of fsa. In the sequel, we explain how to prove that an interval obtained with (get_itv fsa) satisfies this specification.

7.4.3 POP relaxations

To relax semialgebraic optimization problems into POP inside Coq, we first define the pop record construction:

```
Record pop := mk_pop {
      cstr : seq PExpr;
       obj : PExpr;
  lift_idx : positive}.
```

The entry lift_idx indicates the total number of variables of a given pop. One builds objects of type pop by using the constructor mk_pop. From this definition, three projections are automatically available:

```
cstr p : seq PExpr := let (cstr, _, _) := p in cstr
obj p :  PExpr := let (obj, _, _) := p in obj
lift_idx p  := let (lift_idx, _, _) := p in lift_idx
```

Example 7.5. Let a POP be defined by a finite sequence of polynomials $g_1, \dots, g_m$ (represented by hyps), an objective polynomial p (encoded by pe). Let var be the number of variables involved in $p, g_1, \dots, g_m$. Then, we build this POP with (mk_pop hyps pe var).

Let p be a polynomial represented by pe and $[i_1, i_2]$ an interval represented by (Itv i1 i2). The procedure gen_cstr_itv : PExpr → itv → seq PExpr returns the two polynomials $p - i_1$ and $i_2 - p$. The function pol_in_itv : Env → PExpr → itv → Prop returns the proposition which states that these two polynomial expressions are nonnegative. Similarly, (fsa_in_itv l f itv) returns the proposition which states that the values of a semialgebraic function are enclosed by the interval $[i_1, i_2]$.

The following inductive procedure is the implementation in Coq of the algorithm sa_lift depicted in Figure 2.2. For the sake of simplicity, we only show the case analysis for the constructors Poly, Fadd and Fdiv:

```
Fixpoint sa_lift (hyps : seq PExpr) f  var : pop :=
 match f with
 | Poly pe itv ⇒ mk_pop hyps pe var
 | Fadd f1 f2 _ ⇒
    let pop1 := sa_lift hyps f1 var in
    let pop2 := sa_lift hyps f2 (lift_idx pop1) in
    let (obj1, obj2)  := (obj pop1, obj pop2) in
    let (cstr1, cstr2):= (cstr pop1, cstr pop2) in
     mk_pop (cstr1 ++ cstr2) (PEadd obj1 obj2) (lift_idx
        pop2)
 | Fdiv f1 f2 itv ⇒
   let (itv1, itv2)  := (get_itv f1, get_itv f2) in
   let pop1 := sa_lift hyps f1 var in
   let pop2 := sa_lift hyps f2 (lift_idx pop1) in
   let (obj1, obj2)  := (obj pop1, obj pop2) in
   let (cstr1, cstr2):= (cstr pop1, cstr pop2) in
   let lift_idx_div  := lift_idx pop2 + 1 in
   let obj_div       := PEX lift_idx_div in
   let lhs_cstr  := PEsub (PEmul obj2 obj_div) obj1 in
   let rhs_cstr  := PEsub obj1 (PEmul obj2 obj_div) in
   let cstr_div := lhs_cstr :: rhs_cstr :: cstr1 ++ cstr2
                 ++ gen_cstr_itv obj1 itv1
                 ++ gen_cstr_itv obj2 itv2 in
     mk_pop cstr_div obj_div lift_idx_div
   | [...]
  end.
```

- When f is a polynomial, the basic semialgebraic lifting is constructed as in Example 7.5.

- When f is the addition of two semialgebraic functions f_1 and f_2, the function sa_lift is applied recursively to f1 (resp. f2) to obtain the polynomial constraints sequence cstr1 (resp. cstr2) and the objective function obj1 (resp. obj2). The sequence of constraints of the resulting POP is obtained by merging the polynomial inequality constraints cstr1 and cstr2.

- When f is the division of two semialgebraic functions f_1 and f_2, we also apply recursively sa_lift to f1 and f2. Moreover, we use gen_cstr_itv to add bounds on the polynomial obj1 (resp. obj2) that represents f_1 (resp. f_2). The polynomial expression obj_div is built from the lifting variable which represents f. Finally, we add the polynomial (PEsub (PEmul obj2 obj_div)obj1) and its opposite to the sequence of constraints in order to encode the polynomial equality (obj2 obj_div = obj_1).

For the sequel, one also needs the auxiliary procedures:

```
(* Propositional equality between a semialgebraic function
   and a polynomial *)
Definition fsa_pol_eq l (f : Fsa) (p : PExpr) : Prop :=
  Feval l f = PEeval l p.

(* Access to the fields of a pop record, built after a
   semialgebraic lifting *)
Definition get_obj hyps f var := obj (sa_lift hyps f var).
Definition get_cstr hyps f var := [...].
Definition get_var hyps f var := [...].
```

By using Lemma Putinar_Psatz_correct (see Section 7.3), one can prove that a conjunction of polynomial constraints (returned by conj_PExpr_nonneg l hyps) implies the nonnegativity of a polynomial expression. Hence, one can prove that the range of a polynomial p is enclosed by an interval itv. In Coq, this implication is stated as follows:

```
Definition pop_valid l hyps p itv :=
  conj_PExpr_nonneg l hyps → pol_in_itv l p itv.
```

7.4.4 Formal Bounds for Semialgebraic Functions

Given a semialgebraic expression (f : Fsa), the next algorithm returns a conjunction of propositions that involve either polynomials (e.g. pop_valid l cstr obj itv), intervals (e.g. itv_without_0 (get_itv f2)) or environments (e.g. fsa_pol_eq l f).

```
Fixpoint fsa_valid l hyps f var :=
 let cstr := get_cstr hyps f var in
 let obj  := get_obj hyps f var in
  match f with
   | Poly p itv ⇒ pop_valid l hyps p itv
   | Fdiv f1 f2 itv ⇒
        fsa_valid l hyps f1 var
     ∧ fsa_valid l hyps f2 (get_var hyps f1 var)
     ∧ fsa_pol_eq l f obj ∧ itv_without_0 (get_itv f2)
```

```
      ∧ pop_valid l cstr obj itv
    | [...]
  end.
```

Our main correctness lemma states that Problem 7.4.1 can be solved by proving the conjunction of propositions obtained with `fsa_valid`:

```
Lemma fsa_lifting_correct l hyps f var:
  fsa_valid l hyps f var →
  (conj_PExpr_nonneg l hyps → fsa_in_itv l f (get_itv f)).
```

Proof. We prove the lemma by induction on the structure of semialgebraic expressions. Actually, we need an auxiliary lemma with a stronger conclusion than the one of `fsa_lifting_correct` in order to obtain stronger induction hypotheses. This conclusion `fsa_lifting_goal` is built with the conjunction of three propositions:

```
Definition fsa_lifting_goal l hyps f var :=
   pol_in_itv l (get_obj hyps f var) (get_itv f)
 ∧ fsa_pol_eq l f (get_obj hyps f var)
 ∧ conj_PExpr_nonneg l (get_cstr hyps f var)).
```

We define `fobj := get_obj hyps f var` to be the objective function that represents `f`. Then, the first proposition states that `fobj` is enclosed by the interval `( get_itv f)`. The second one denotes the equality between `f` and `fobj`. The last one states that all the constraints generated by `sa_lift` must hold.

□

Numerical Results

As previously explained, formal lower bounds for semialgebraic problems can be obtained by proving propositions built with `fsa_valid`. The most cpu-intensive task is checking propositions built with `pop_valid`.

Table 7.2 presents the results obtained when proving the correctness of lower bounds for POP relaxations of Flyspeck inequalities. The execution time is compared with the informal verification time when using either `minimax_optim` (results from Table 4.2) or `template_optim` (results from Table 6.4).

Table 7.2: Formal Bounds Computation Results for POP relaxations of Flyspeck Inequalities

Inequality	Method	#boxes	Informal Nonlinear Optimization Time	Formal Polynomial Optimization Time
9922699028	`minimax_optim`	14	244 *s*	972 *s*
	`template_optim`	39	190 *s*	2218 *s*
3318775219	`minimax_optim`	266	4423 *s*	18255 *s*
	`template_optim`	338	1560 *s*	19136 *s*

The formal verification of SOS certificates is the bottleneck of the computational certification task. Indeed, it is 22 times slower to prove the correctness of POP lower bounds for such inequalities. Moreover, the number of domain subdivisions increases when one certifies the inequalities with `template_optim`.

At a first glance, the `minimax_optim` algorithm seems more efficient for computing formal bounds of medium-scale problems. However, these results do not take into account the time required to check the correctness of the semialgebraic estimators (either minimax or maxplus) for univariate functions. Validating these estimators could be handled with the `interval` tactic [Mel12]. We also mention that recent techniques have been developed to obtain rigorous error bounds for Chebyshev interpolation polynomial [BJ10]. As far as our knowledge, these techniques are not yet available in COQ.

Chapter 8

Conclusion and Perspectives

8.1 Achievements

The present nonlinear template method computes certified lower bounds for global optimization problems. It can provide tight minimax or maxplus semialgebraic estimators to certify non-linear inequalities involving transcendental multivariate functions. Our algorithms can solve both small and intermediate size inequalities of the Flyspeck project as well as global optimization problems issued from the literature, with a moderate order of SDP relaxation.

The proposed approach bears some similarity with the "cutting planes" proofs in combinatorial optimization, the cutting planes being now replaced by nonlinear inequalities. It also allows one to limit the growth of the number of lifting variables as well as of polynomial constraints to be handled in the POP relaxations, at the price of a coarser approximation.

Thus, our method is helpful when the size of optimization problems increases. Indeed, the coarse lower bounds obtained (even with a low SDP relaxation order) are better than those obtained with interval arithmetic or high-degree polynomial approximation.

We also derived a hierarchy of semidefinite relaxations to approximate semialgebraic functions with multivariate polynomials that converge to the best (for the L_1 norm) degree-d polynomial underestimators. Thus, we build more accurate nonlinear templates by constructing a sequence of semialgebraic estimators, which can be arbitrary close to the "best" maxplus semialgebraic estimators.

Furthermore, the formal part of our implementation, currently can check medium size semialgebraic certificates. The SOS certificates checker was significantly improved by a careful implementation of informal and formal libraries:

1. Our informal certification tool exploits the system properties of the problems (sparsity) to derive semialgebraic relaxations involving less SDP variables, thus yields smaller SOS certificates.

2. On the COQ side, we took benefit from the machine modular arithmetic by defining a customized ring of polynomials with arbitrary-size rational coefficients.

8.2 Perspectives

This work provides many directions for further investigation of research. In the sequel, we briefly outline some of these tracks.

First, we explain how to derive some error bounds for the estimators obtained from the nonlinear template method (Section 8.2.1). Then, we present some extensions of our nonlinear template method to handle a more general class of nonlinear optimization, such as nonlinear optimization with transcendental constraints (Section 8.2.2), continuous/discrete time optimal control problems (Section 8.2.3). Finally, we highlight some improvements of our formal certification framework that would allow to perform more advanced nonlinear reasoning (Section 8.2.4).

8.2.1 Complexity of the nonlinear template method

For the sake of simplicity, we consider the following sub-case of Problem (1.1.2):

$$f^* := \inf_{\mathbf{x} \in [0,1]^n} u(p(\mathbf{x})) \ , \tag{8.2.1}$$

where $u \in \mathcal{D}$ is a smooth univariate function and p is a degree-d_p multivariate polynomial, with $d_p \geqslant 2$. For the sake of simplicity, we suppose that the range of values of p is enclosed by the interval $[0, 1]$. Then, let us call f_d the degree-d minimax polynomial approximation of u on $[0, 1]$ and consider the following POP:

$$m_d := \inf_{\mathbf{x} \in [0,1]^n} f_d(p(\mathbf{x})) \ . \tag{8.2.2}$$

One also has $M_d := \sup_{\mathbf{x} \in [0,1]^n} f_d(p(\mathbf{x}))$.

Let $k \geqslant \lceil (dd_p)/2 \rceil$. One can obtain a lower bound μ_k of m_d by solving the associated SDP relaxation Q_k (see Section 2.2).

The following conjecture allows to derive error bounds for the Lasserre hierarchy of approximations.

Conjecture 8.1 (De Klerk, Laurent [dKL10]). *Let $M_n(g)$ be the n-truncated quadratic module generated by the polynomials $g_1 := x_1 - x_1^2, \ldots, g_n := x_n - x_n^2$. Then, for n even, one has:*

$$\prod_{i=1}^{n} x_i + \frac{1}{n(n+2)} \in M_n(g) \ .$$

Assume that Conjecture 8.1 holds. Then, it would be interesting to extend the result given in [dKL10] to the polynomial $f_d \circ p$ (see the conclusion for the case of quadratic polynomials). In this case, we could derive an error bound of $m_d - \mu_k$, corresponding to the relaxation gap between the optimal values of the SDP (Q_k) and Problem (8.2.2). This error bound would typically depend on n, d, d_p, m_d, M_d and $L(f_d \circ p)$, where L is defined as follows:

$$\begin{aligned} L : \mathbb{R}[\mathbf{x}] &\to \mathbb{R} \\ q &\mapsto \max_{\boldsymbol{\alpha}} |q_{\boldsymbol{\alpha}}| \frac{\prod_{i=1}^{n} \alpha_i!}{(\Sigma_{i=1}^{n} \alpha_i)!} \ . \end{aligned}$$

Now, consider the SDP relaxation Q_k. The number of moment variables in Q_k is in $O(n^{2k})$ and the size of the SDP matrix is in $O(n^{k+1})$.

Let $\epsilon > 0$ be fixed and $\text{Compl}(f^*, \epsilon)$ be the number of real arithmetic operations needed to obtain an ϵ-solution of f^* with the `minimax_optim` algorithm. We note $\text{Digits}(f_d^*, \epsilon)$ the number of accuracy digits in an $\epsilon/2$-solution of m_d (when solving the SDP relaxation Q_k at sufficiently large order). Applying (2.1.9), we have:

$$\text{Compl}(f^*, \epsilon) = O(1) n^{5/2+7\times 2^{nr}} \text{Digits}(f_d^*, \epsilon) \ . \tag{8.2.3}$$

Finally, we could combine the error bound of $m_d - \mu_k$ and Theorem 8.2 to derive an upper bound on r.

Theorem 8.2. *Let $I := [m, M], u \in \mathcal{C}^{d+1}(I)$ and denote by f_d the degree-d minimax polynomial of u on I. Then, the sequence of best uniform polynomial approximations $(f_d)_{d\in\mathbb{N}}$ of u on I satisfies:*

$$\|u - f_d\|_\infty \leqslant \frac{(M-m)^{d+1} \|u^{(d+1)}\|_\infty}{2^{2d+1}(d+1)!} \ .$$

To study the complexity of the `samp_optim` algorithm, it seems promising to use the error bounds derived by Nie and Schweighofer [NS07, Sch04] for constrained POP (not restricted to the optimization over the hypercube). However, these bounds involve some constants, that depend on the POP data and are not trivial to compute in general.

8.2.2 Extension to global optimization with transcendental constraints

We can generalize our modular templates algorithm to solve optimization problems with non-polynomial constraints. Given $f, f_1, \dots, f_l \in \langle \mathcal{D} \rangle^{\text{sa}}$ and $h_1, \dots, h_q \in \mathcal{A}$, we formulate Problem (1.1.1) as follows:

$$\inf_{\mathbf{x} \in K_{\text{tr}}} f(\mathbf{x}) \ , \tag{8.2.4}$$

where,

$$K_{\text{tr}} := \{\mathbf{x} \in S : f_1(\mathbf{x}) \geqslant 0, \dots, f_l(\mathbf{x}) \geqslant 0\} \ , \tag{8.2.5}$$

and

$$S := \{\mathbf{x} \in \mathbb{R}^n : h_1(\mathbf{x}) \geqslant 0, \dots, h_q(\mathbf{x}) \geqslant 0\} \ . \tag{8.2.6}$$

First, we mention that one could optimize a semialgebraic function over such a compact set S, whose inequalities constraints are defined with functions in $\mathcal{A}$ (rather than polynomials). Following the approach described in [Put93], one rewrites S as the projection of a basic semialgebraic set in a lifted space and solves the POP relaxation with the `min_pop` and `max_pop` procedures. This allows to use the `template_approx` algorithm with a compact set S as in (8.2.6).

We assume that $f, f_1, \dots, f_l$ are defined on S. Given a finite sequence of control points s, the `template_approx` method (see Section 6.4 and Figure 6.5) can be used as an auxiliary procedure to obtain a hierarchy of outer compact approximation sets K_s^+ for the feasible set K_{tr}. This leads to the generalized approximation algorithm depicted in Figure 8.1.

The generalized optimization algorithm also chooses the finite sequence of control points s dynamically. After each call to the approximation procedure, we compute a

```
Input: tree $t$, constraints set $K_{\mathrm{tr}}$, semidefinite relaxation order $k$, control points sequence $s = \{\mathbf{x}_1, \dots, \mathbf{x}_p\} \subset K_{\mathrm{tr}}$
Output: lower bound $m$, upper bound $M$, lower semialgebraic estimator $t^-$, upper semialgebraic estimator $t^+$
For $c \in \{1, \dots, l\}$: $m_i, M_i, f_i^-, f_i^+$ := template_approx($f_i, S, k, s$)
$K^+ := \{\mathbf{x} \in S : f_1^-(\mathbf{x}) \geqslant 0, \dots, f_l^-(\mathbf{x}) \geqslant 0\}$
return template_approx($t, K^+, k, s$)
```

Figure 8.1: An extension of `template_approx` for non-polynomial constraints

minimizer candidate $\mathbf{x}_{opt} \in K_s^+$ of the underestimator tree t^-. Then, the projection of $\mathbf{x}_{opt}$ into K_{tr} is added to the set of control points s.

Let t_p^- be the semialgebraic underestimator obtained at step p of the optimization procedure. We call $\mathbf{x}_p^*$ the projection of one minimizer of t_p^- on K_{tr}.

Corollary 8.3 (Convergence of the Generalized Nonlinear Optimization Algorithm)**.** *Suppose that Assumption (3.4) holds. Then, every limit point of the sequence $(\mathbf{x}_p^*)_{p \in \mathbb{N}}$ is a global minimizer of t over K.*

Proof. It follows from Theorem 6.18 that the semialgebraic estimators obtained with `template_approx` uniformly converge to the nonlinear functions which define the constraints inequalities of K_{tr}. It implies that each limit point of the sequence K_s^+ of outer compact approximation sets is K_{tr}. □

8.2.3 Extension to nonlinear optimal control problems

Our method can also be applied to nonlinear optimal control problems (OCP), for which the problem data are transcendental. Two different approaches can be considered.

Nonlinear continuous-time OCP

As a classical example, let us consider the minimal time OCP. Given two integers $n, m \in \mathbb{N}$, let $t \mapsto \mathbf{x}(t) \in \mathbb{R}^n$ be the state trajectory and $t \mapsto \mathbf{u}(t) \in \mathbb{R}^m$ be a bounded and measurable input function. Let $X_{\mathrm{tr}}, K_{\mathrm{tr}}$ (resp. U_{tr}) be some compact subsets of $\mathbb{R}^n$ (resp. $\mathbb{R}^m$), defined by a finite number of inequalities constraints, involving functions in $\langle \mathcal{D} \rangle^{\mathrm{sa}}$ (as in (8.2.5)). Let x_0 be the initial position, so that $\mathbf{x}(0) = x_0$. Let $T > 0$ be the *final time* real variable. Given a smooth endpoint cost function $\phi : \mathbb{R}^n \to R$ and a smooth dynamics $g : [0, +\infty) \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, we define the Mayer OCP:

$$\min_{\mathbf{x},\mathbf{u}} \quad \phi(x_T)$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = g(t, \mathbf{x}, \mathbf{u}) \ , \tag{8.2.7}$$

$$(\mathbf{x}(t), \mathbf{u}(t)) \in X_{\mathrm{tr}} \times U_{\mathrm{tr}} \text{ a.e. on } [0, T] \ , \tag{8.2.8}$$

$$\mathbf{x}(T) \in K_{\mathrm{tr}} \ , \tag{8.2.9}$$

$$\mathbf{x}(t) \text{ is well defined on } [0, T] \ . \tag{8.2.10}$$

The constraint (8.2.7) is the differential equation satisfied by a trajectory solution that starts from $\mathbf{x}(0)$. The input, state (resp. the terminal input $\mathbf{x}(T)$) satisfy the constraints (8.2.8) (resp. (8.2.9)).

When all the data $(\phi, g, X_{\text{tr}}, U_{\text{tr}}, K_{\text{tr}})$ are polynomial, Lasserre et al.(see [LHPT08]) derived a convergent hierarchy of linear matrix inequalities (LMI) relaxations to solve continuous-time OCP, using the so called *occupation measures*. Some of these techniques are implemented in the software POCP [HLS08]. They could be combined with the extension of `template_approx` (see Figure 8.1) to solve OCP involving transcendental data.

So far, for polynomial data, the combined LMI/occupation measures has been applied only to instances of relatively small size (dimension 3). Possibly coarse certified bounds on the value function could be obtained for higher dimensional instances, by the nonlinear template method - after a time discretization scheme. We next sketch this method starting from a discrete-time OCP.

Nonlinear template applied to discrete-time OCP

Here, we focus on the discrete-time Mayer problem:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \phi(x_T) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = g(\mathbf{x}_k, \mathbf{u}_k) \ , && (8.2.11) \\ & (\mathbf{x}_k, \mathbf{u}_k) \in X_{\text{tr}} \times U_{\text{tr}} \ (0 \leqslant k \leqslant T) \ , && (8.2.12) \\ & \mathbf{x}(T) \in K_{\text{tr}} \ , && (8.2.13) \\ & \mathbf{x}(0) = x_0 \ , && (8.2.14) \end{aligned}$$

where $\phi : \mathbb{R}^n \to R$ is the endpoint cost function, g is the discrete-time dynamics and $X_{\text{tr}}, U_{\text{tr}}, K_{\text{tr}}$ are compact sets defined as in the previous paragraph. This problem can be cast as a high-dimensional optimization problem of the form considered in Chapter 3, in which one minimizes a function f given by an abstract syntax tree. For a control problem, the variables are of course $x_1, \dots, x_T, u_0, \dots, u_T$ and the syntax tree has the shape of a "gourmand de la vigne" (we borrow the term to Viennot [Vie90]), *i.e.* to a very thin binary tree (of linear shape). In this case, for every k, the set of reachable vectors x_k is abstracted by a template. These sets can be computed in a forward fashion, by exploiting the dynamics, whereas the templates can be refined in a backward fashion. Hence, the template method includes as a special case a set theoretical version of the familiar state/co-state optimality conditions in control.

8.2.4 Formal procedures for nonlinear reasoning

Formal non-commutative Polynomial Optimization

Given $n \in \mathbb{N}$, we consider the monoid $\langle \underline{X} \rangle$ freely generated by $\underline{X} := (X_1, \dots, X_n)$. In other words, this monoid is defined with words in the n non-commutative letters $X_1, \dots, X_n$. Let consider the free algebra $\mathbb{R}\langle \underline{X} \rangle$ of non-commutative polynomials (NC polynomials) and equip $\mathbb{R}\langle \underline{X} \rangle$ with the *involution* $*$, that reverses words. We call a *hermitian square* an NC polynomial of the form $f^* f$. Let $\Sigma\langle \underline{X} \rangle$ be the set of sums of hermitian squares. A *commutator* is an element of the form $[f, g] := fg - gf$, for $f, g \in \mathbb{R}\langle \underline{X} \rangle$.

Given $f \in \mathbb{R}\langle \underline{X} \rangle$, consider the following *trace-minimum* problem:

$$f^* := \inf_{\underline{A} \in \mathcal{S}_d^n} \mathbf{Tr}(f(\underline{A})) \ , \tag{8.2.15}$$

where $\mathcal{S}_d^n$ is the set of n-tuples of symmetric $d \times d$ matrices.

The MATLAB `NCSOStools` toolbox [CKP11] computes lower bounds of Problem (8.2.15) by solving the following hierarchy of SDP relaxations:

$$f_k^{\text{ncsos}} := \sup\{a \mid f - a \in \Theta_k\} \ , \tag{8.2.16}$$

where Θ_k is the convex cone of all degree-k NC polynomials that can be written as sums of hermitian squares and commutators. We refer the reader to [BCKP13] for more details on these relaxations. Under certain assumptions, the Parrilo-Peyrl rounding and projection method applies for rational NC polynomials. Hence, as in the commutative case, a hybrid numeric-symbolic approach may allow one to verify in COQ the correctness of NC certificates.

This would require to prove equalities in a non-commutative ring in COQ, by extending the features of some existing reflexive tactics libraries (for instance, the tactic `ring` already deals with the commutative case). This development could be mandatory to handle computational proofs arising from important open problems such as Connes's embedding Conjecture [Con76] or the generalized Lax Conjecture [NT12]. More details on the connection between NC polynomials and polynomial differential operators can be found in [Cim09].

Towards efficient formal procedures for nonlinear reasoning

The implementation of polynomial arithmetic still needs some streamlining, as checking ring equalities in COQ remains the bottleneck of our verification procedure. An external tool could reformulate these equalities, so that our problems only involve polynomials with integer coefficients (using the implementation of arbitrary-size integers `bigZ`). Thus, we would get rid of almost all gcd operations inside COQ. One could consider another type of coefficients, such as dyadic rationals, that can be seen as arbitrary precision floating-point numbers. Alternative sparse representation of polynomials might also provide significant improvements.

Appendix A

Flyspeck Nonlinear Inequalities

We recall the following definitions [Hal03]:

$$
\begin{array}{rcl}
h_0 & := & 1.26 \ , \\
h_+ & := & 1.3254 \ , \\
\Delta \mathbf{x} & := & x_1 x_4 (-x_1 + x_2 + x_3 - x_4 + x_5 + x_6) \\
 & & + x_2 x_5 (x_1 - x_2 + x_3 + x_4 - x_5 + x_6) \\
 & & + x_3 x_6 (x_1 + x_2 - x_3 + x_4 + x_5 - x_6) \\
 & & - x_2 x_3 x_4 - x_1 x_3 x_5 - x_1 x_2 x_6 - x_4 x_5 x_6, \\
\text{rho\_x}(\mathbf{x}) & := & -x_1 x_1 x_4 x_4 - x_2 x_2 x_5 x_5 - x_3 x_3 x_6 x_6 \\
 & & + 2 x_1 x_2 x_4 x_5 + 2 x_1 x_3 x_4 x_6 + 2 x_2 x_3 x_5 x_6 \ , \\
\text{rad2\_x}(\mathbf{x}) & := & \frac{\text{rho\_x}(\mathbf{x})}{4 \Delta \mathbf{x}} \ , \\
\text{dih}(\mathbf{x}) & := & \pi / 2 + \arctan \frac{-\partial_4 \Delta \mathbf{x}}{\sqrt{4 x_1 \Delta \mathbf{x}}} \ , \\
\text{perm}_2(\mathbf{x}) & := & (x_2, x_1, x_3, x_5, x_4, x_6) \ , \\
\text{perm}_3(\mathbf{x}) & := & (x_3, x_1, x_2, x_6, x_4, x_5) \ , \\
\text{sol}(\mathbf{x}) & := & \text{dih}(\mathbf{x}) + \text{dih}(\text{perm}_2(\mathbf{x})) + \text{dih}(\text{perm}_3(\mathbf{x})) - \pi \ , \\
\text{const}_1 & := & \text{sol}(2,2,2,2,2,2)/\pi \ , \\
\text{ly}(x) & := & 1 + \frac{2-x}{0.52} \ , \\
\text{lnazim}(\mathbf{x}) & := & \text{ly}(\sqrt{x_1}) \, \text{dih}(\mathbf{x}) \ , \\
\text{taum}(\mathbf{x}) & := & \text{sol}(\mathbf{x})(1 + \text{const}_1) - \text{const}_1 [\text{lnazim}(\mathbf{x}) \\
 & & + \text{lnazim}(\text{perm}_2(\mathbf{x})) + \text{lnazim}(\text{perm}_3(\mathbf{x}))] \ .
\end{array}
$$

In the sequel, we present some inequalities issued from the nonlinear part of Flyspeck. The classification of these inequalities (semialgebraic, small-sized and medium-size) is based on the number of transcendental functions that are involved.

A.1 Semialgebraic Flyspeck Inequalities

Lemma A.1 (JNTEFVP 1). $\forall \mathbf{x} \in [4, 4h_0^2]^5 \times [8, 16h_0^2], \partial_4 \Delta \mathbf{x} \geqslant 0.$

Lemma A.2 (TSKAJXY TADIAMB). $\forall \mathbf{x} \in [4h_+^2, 8]^2 \times [4, 8]^4, \text{rad2\_x}(\mathbf{x}) \geqslant 2.$

A.2 Transcendental Flyspeck Inequalities

A.2.1 Small-sized Flyspeck Inequalities

Lemma A.3 (7067938795).

$$\forall \mathbf{x} \in [4, 6.3504]^6, \mathrm{dih}(\mathbf{x}) + \pi/2 - 0.46 > 0.$$

Lemma A.4 (9922699028).

$$\forall \mathbf{x} \in [4, 6.3504]^3 \times [6.3504, 8] \times [4, 6.3504]^2, 1.6294 - \mathrm{dih}(\mathbf{x}) - 0.2213(\sqrt{x_2} + \sqrt{x_3} + \sqrt{x_5} + \sqrt{x_6} - 8.0) + 0.913(\sqrt{x_4} - 2.52) + 0.728(\sqrt{x_1} - 2.0) \geqslant 0.$$

Lemma A.5 (3318775219).

$$\forall \mathbf{x} \in [4, 6.3504]^3 \times [6.3504, 8] \times [4, 6.3504]^2, \mathrm{dih}(\mathbf{x}) - 1.629 + 0.414(\sqrt{x_2} + \sqrt{x_3} + \sqrt{x_5} + \sqrt{x_6} - 8.0) - 0.763(\sqrt{x_4} - 2.52) - 0.315(\sqrt{x_1} - 2.0) \geqslant 0.$$

A.2.2 Medium-size Flyspeck Inequalities

Lemma A.6 (7394240696).

$$\forall \mathbf{x} \in [4, 6.3504]^6, \mathrm{sol}(\mathbf{x}) - 0.55125 - 0.196(\sqrt{x_4} + \sqrt{x_5} + \sqrt{x_6} - 6.0) + 0.38(\sqrt{x_1} + \sqrt{x_2} + \sqrt{x_3} - 6.0) \geqslant 0.$$

Lemma A.7 (46529697461).

$$\forall \mathbf{x} \in [4, 6.3504]^4 \times [4.73976441, 6.3504] \times [4, 6.3504], \mathrm{taum}(\mathbf{x}) \geqslant 0.004.$$

Appendix B

The NLCertify Package

NLCertify is a software package for handling certification of nonlinear inequalities involving transcendental multivariate functions. Given a box $K := [a_1, b_1] \times \cdots \times [a_n, b_n]$ and a multivariate transcendental function f, our aim is to assert the inequality: $\forall \mathbf{x} \in K, f(\mathbf{x}) \geqslant 0$.

B.1 Source Code Organization

The source code of the tool can be downloaded on the web page of the author at the following url: www.lix.polytechnique.fr/~magron/nlcertify.tar.gz. The code includes OCAML files (.ml, mll, mly), COQ vernacular files (.v). Some HOL-LIGHT files (.hl) issued from the Flyspeck repository [Hal13] are also available.

The main directory nlcertify contains several OCAML source files, implementing the algorithms described in Part II. The main function is written in nlcertify.ml. It also contains the following subdirectories:

- Sphere_parser: it contains some parsing and translation files.
- flyspeck_dir : some HOL-LIGHT files contain the definitions and inequalities issued from the nonlinear part of Flyspeck (see Appendix A for examples). Their translations from HOL-LIGHT to OCAML (resp. COQ) are also available.
- coq : it contains the source code of the formal checker for SOS certificates.
- log : this folder is created after the first execution of the nlcertify executable. It contains I/O files (for SDPA, Sollya) and log files.

B.2 Installation of the NLCertify Package

B.2.1 Compilation Dependencies

NLCertify needs external software libraries to be compiled. Optional packages are also included in the following list and we recommend their installation.

- OCAML : http://caml.inria.fr/download.fr.html
- OPAM (optional) : http://opam.ocamlpro.com/

- **Mandatory** OCAML **libraries** : ocamlfind, ocamlbuild, ocamlgraph, zarith and lacaml. We highly recommend to install them with OPAM.
- SDPA : http://sdpa.sourceforge.net/download.html
- COQ (optional) : http://coq.inria.fr/download with SSREFLECT : https://gforge.inria.fr/frs/download.php/31453/ssreflect-1.4-coq8.4.tar.gz
- Sollya (optional) : http://sollya.gforge.inria.fr/

B.2.2 Installation

The NLCertify tool can be compiled using the make command. If no error is displayed, the main executable file nlcertify is created in the main directory.

If the SSREFLECT libraries are installed and if the COQ compiler coqc is present in the path, then one can compile the vernacular files inside the coq folder, using the following commands:

```
% cd coq
% coqc Env.v sos_horner.v remainder.v
% cd ..
```

B.3 User Parameters

The user can tune the parameters of the nonlinear certification scheme, by editing the param.transc file, whose content looks like:

```
******* User Parameters *******
[...]
* Number of control points for maxplus approximation
   samp_iters = 1

* Sollya parameters
   approx_minimax = true
   minimax_degree_sqrt = 4
[...]
```

Now, we detail the purpose of each parameter. Note that the parameters are typed. For instance, the type of the parameter scale_pol is bool, so the user can edit the file by writing either scale_pol = true or scale_pol = false. The type of the relaxation order relax_order is int, thus the user may write relax_order = 2 to solve SDP at first or second relaxation order (when the degree of the minimal relaxation order is not greater than 2).

B.3.1 General Parameters

- input_ineqs_filename (string) : the file where the inequalities are defined (the default setting is test.ineq).

- `xconvert_variables (bool)` : when `xconvert_variables = true`, then the objective function $\mathbf{x} \mapsto f(x_1, \ldots, x_n)$ is replaced by $\mathbf{y} \mapsto f(y_1, \ldots, y_n)$ with $y_i := \sqrt{x_i}$ $(i = 1, \ldots, n)$. The input box $K := [a_1, b_1] \times \cdots \times [a_n, b_n]$ is replaced by $K' := [a_1^2, b_1^2] \times \cdots \times [a_n^2, b_n^2]$.

- `samp_iters (int)` : maximal length of the sequence of control points for max-plus approximation (corresponding to #s in the numerical experiments). The case `samp_iters = 0` coincides with `ia_sos` (Section 6.5)

- `approx_minimax (bool)` : when `approx_minimax = true`, minimax approximations are used to estimate univariate transcendental functions.

- `minimax_sqrt (bool)` : when `minimax_sqrt = true`, minimax approximations are used to estimate (even if `approx_minimax = false`) the square roots of univariate functions . As an example, the objective function of the inequalities presented in Appendix A contains such semialgebraic functions.

- `minimax_degree (int)` : the degree of minimax approximations (parameter d in Section 4.3)

- `minimax_degree_sqrt (int)` : the degree of minimax estimators for square roots of univariate functions.

- `minimax_precision (int)` : the working precision used inside the `Sollya` tool (see the documentation in [CJL10]). The default value is 165.

- `bb (bool)` : when `bb = true`, we perform domain subdivision until we succeed to certify the inequality.

B.3.2 POP/SDP Relaxations Parameters

Let $K := [a_1, b_1] \times \cdots \times [a_n, b_n]$ be a box and let $f_{\text{pop}}, g_1, \ldots, g_m \in \mathbb{R}[\mathbf{x}]$. We recall the general constrained POP:

$$(POP) \begin{cases} \inf_{\mathbf{x} \in K} & f_{\text{pop}}(\mathbf{x}) \ , \\ \text{s.t.} & g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0 \ . \end{cases}$$

Several options are available to handle numerical issues when solving SDP relaxations of (POP). In the sequel, we will often refer to this problem to describe the usage of these options. We also recall that the minimal SDP relaxation order of Problem (POP) is $k_0 := \max(\lceil \deg f_{\text{pop}}/2 \rceil, \max_{1 \leqslant j \leqslant m} \lceil \deg g_j/2 \rceil)$. Note that there exists $M > 0$ such that $M - \sum_{i=1}^n x_i^2 \geqslant 0$. Let $(POP)'$ be the scaled POP version of Problem (POP):

$$(POP)' \begin{cases} \inf_{\mathbf{x}' \in [0,1]^n} & f'_{\text{pop}}(\mathbf{x}') \ , \\ \text{s.t.} & g'_1(\mathbf{x}') \geqslant 0, \ldots, g'_m(\mathbf{x}') \geqslant 0 \ , \end{cases}$$

where

$$x'_i := (x_i - a_i)/(b_i - a_i) \ (i = 1, \ldots, n) \ ,$$

$$f'_{\text{pop}} := f_{\text{pop}}/\|f_{\text{pop}}\|_\infty, \quad g'_j := g_j/\|g_j\|_\infty \ (j = 1, \ldots, m) \ .$$

Note that the function $\| \cdot \|_\infty$ returns the maximum magnitude of the coefficients of polynomials in $\mathbb{R}[\mathbf{x}']$.

After solving Problem (POP), one obtains the following decomposition from the output of SDPA:

$$f_{\text{pop}}(\mathbf{x}) - \mu_k = \sum_{j=0}^{m} g_j(\mathbf{x}) \sum_{i=1}^{r_j} \lambda_{ij} v_{ij}^2(\mathbf{x}) \ . \tag{B.3.1}$$

- `check_certif_coq (bool)` : when `check_certif_coq = true`, the correctness of the SOS certificates is checked in COQ

- `sos_verb (bool)` : when `sos_verb = true`, several information about SOS relaxations (moment and localizing matrices, supports of polynomials, *etc*). For each POP, the polynomial data $(f_{\text{pop}}, g_1, \dots, g_m)$ are also displayed.

- `pop_verb (bool)` : when `pop_verb = true`, the objective functions of semialgebraic problems are displayed.

- `sdp_verb (bool)` : when `sdp_verb = true`, execution time and I/O file names of SDPA are displayed.

- `relax_order (int)` : when `relax_order =` k, SDP relaxations of (POP) are solved using a relaxation order not greater than $\max(k, k_0)$.

- `reduce_sos (bool)` : this option allows to eliminate the redundant vectors for any SOS representation of $f_{\text{pop}} - \sum_{j=1}^{m} \sigma_j g_j$ (see Figure 2.4).

- `scale_pol (bool)` : when `scale_pol = true`, one solves Problem $(POP)'$ instead of Problem (POP).

- `bound_squares_variables (bool)` : this option adds the polynomial inequalities $(b_i - x_i)(x_i - a_i) \geqslant 0$ $(i = 1, \dots, n)$ to the constraints set of (POP).

- `mk_archimedean (bool)` : this option adds the single polynomial inequality constraint $\sum_{i=1}^{n} M - x_i^2 \geqslant 0$ to the constraints set of (POP).

- `eig_tol (float)` : This option replaces each λ_{ij} by 0 in (B.3.1) when `eig_tol` $\geqslant \lambda_{ij}$.

- `eq_tol (float)` : when `eq_tol =` $\epsilon > 0$, then each polynomial equality constraint $h(\mathbf{x}) = 0$ is relaxed into two polynomial inequalities $h(\mathbf{x}) \geqslant 0$ and $h(\mathbf{x}) \geqslant \epsilon$.

- `sdp_solver_epsilon (int)` : the accuracy of the SDP solver SDPA (for more details, see [YFN+10]).

- `sdp_solver_print (int)` : the number of digits displayed in the output of SDPA files.

- `erase_sdpa_files (bool)` : when `erase_sdpa_files = true`, the I/O SDPA files are erased. Otherwise, they are stored in the `log` folder.

B.4 Certification of Nonlinear Inequalities

B.4.1 Input Settings

The user can define the input box $K := [a_1, b_1] \times \cdots \times [a_n, b_n]$ and the multivariate transcendental objective function f in the file `input_ineqs`. The inequality $\forall \mathbf{x} \in K, f(\mathbf{x}) \geqslant m$ is encoded as follows:

```
let box_ineq x1 ... xn = [(a1 , b1); ... ; (an , bn)];;
let obj_ineq x1 ... xn = [( f, m )];;
```

Note that it is mandatory to separate each definition (either for a box or an objective function) with a double semicolon "`;;`". Let us give a concrete example for Problem *MC*. We recall the formulation of the problem:

$$\min_{\substack{-1.5 \leqslant x_1 \leqslant 4 \\ -3 \leqslant x_2 \leqslant 3}} f(\mathbf{x}) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5 x_1 + 2.5 x_2 + 1 \ .$$

Hence we encode the inequality "$\forall \mathbf{x} \in K, f(\mathbf{x}) \geqslant -1.92$" in `NLCertify` as follows:

```
let box_MC x1 x2 = [ (-1.5, 4) ; (-3, 3) ];;
let obj_MC x1 x2 = [ (sin(x1 + x2) + (x1 - x2)**2  - 1.5 *
   x1 + 2.5 * x2 + 1, -1.92 )];;
```

B.4.2 `NLCertify` Execution

Given a inequality (defined as above) identified with `ineq`, the following command line allows to execute the main program:

```
% ./nlcertify ineq
```

For instance, without domain subdivision (setting the option `bb = false`), and with the maxplus method (`approx_minimax = false`) the program returns the following output, after a single iteration (`samp_iter = 1`):

```
% ./nlcertify MC
start Program
1 problem remaining, 0 cuts done
[(-1.5, 4.0) ; (-3.0, 3.0)]
...
min = -62.1516382708
[3.9999984600 ; 2.9999984400]

1 problem solved, 0 cuts done
End of maxplus algorithm
-62.1516382708 <= 0.0000000000
Failed to verify the inequality MC
Total time: 1.149512 seconds
```

Note that the box is displayed, as well as the lower bound m of the objective function on this box ($m \simeq -62$). Here, the minimizer guess is obtained at $\mathbf{x}_{opt} \simeq (4, 3)$. In this case, the relaxation gap is too high to certify the inequality. After 3 iterations, (`samp_iter = 3`), the output is:

```
% ./nlcertify MC
...
min = -0.8304673056
[-1.1951372495 ; -2.3854570200]

-0.8304673056
1 problem solved, 0 cuts done
End of maxplus algorithm
-0.8304673056 <= 0.0000000000
Failed to verify the inequality MC
Total time: 1.828811 seconds
```

The lower bound is more precise ($m \simeq -0.830$) but one has to switch the `bb` option to `true` to solve the inequality:

```
% ./nlcertify MC
...
13 problems solved, 12 cuts done
End of maxplus algorithm
0.0017452960 >= 0.0000000000
Inequality MC verified
Total time: 19.966771 seconds
```

Here, the lower bound obtained after the 12 subdivisions is greater than 0 ($m \simeq 0.002$). The verbosity options can be switched to display more information when the algorithms implemented in `NLCertify` are called on each sub-box (e.g. the sequence of control points, the equation of the estimators, the I/O data of the external solvers, *etc*). These information are saved in the `log/mc.log` file.

Bibliography

[AGG12] A. Adje, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. *Logical methods in computer science*, 8(1):1–32, 2012.

[AGK05] Marianne Akian, Stéphane Gaubert, and Vassili Kolokoltsov. Set coverings and invertibility of Functional Galois Connections. In G.L. Litvinov and V.P. Maslov, editors, *Idempotent Mathematics and Mathematical Physics*, volume 377 of *Contemporary Mathematics*, pages 19–51. American Mathematical Society, 2005. Also ESI Preprint 1447, http://arXiv.org/abs/math.FA/0403441.

[AGL08a] M. Akian, S. Gaubert, and A. Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. *SIAM J. Control Optim.*, 47(2):817–848, 2008.

[AGL08b] Marianne Akian, Stéphane Gaubert, and Asma Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: Basic properties and convergence analysis. *SIAM J. Control Optim.*, 47(2):817–848, February 2008.

[AGMW13a] Xavier Allamigeon, Stéphane Gaubert, Victor Magron, and Benjamin Werner. Certification of bounds of non-linear functions : the templates method, 2013. To appear in the Proceedings of Conferences on Intelligent Computer Mathematics, CICM 2013 Calculemus, Bath.

[AGMW13b] Xavier Allamigeon, Stéphane Gaubert, Victor Magron, and Benjamin Werner. Certification of inequalities involving transcendental functions: combining sdp and max-plus approximation, 2013. To appear in the Proceedings of the European Control Conference, ECC'13, Zurich.

[AGST10] Michaël Armand, Benjamin Grégoire, Arnaud Spiwack, and Laurent Théry. Extending coq with imperative features and its application to sat verification. In Matt Kaufmann and LawrenceC. Paulson, editors, *Interactive Theorem Proving*, volume 6172 of *Lecture Notes in Computer Science*, pages 83–98. Springer Berlin Heidelberg, 2010.

[AH77] K. Appel and W. Haken. Every planar map is 4-colorable. *Illinois Journal of Mathematics*, 21:429–567, 1977.

[AKZ05] M. Montaz Ali, Charoenchai Khompatraporn, and Zelda B. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *J. of Global Optimization*, 31(4):635–672, April 2005.

[AP10] Behzad Akbarpour and Lawrence Charles Paulson. Metitarski: An automatic theorem prover for real-valued special functions. *J. Autom. Reason.*, 44(3):175–205, March 2010.

[BC04] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[BCKP13] Sabine Burgdorf, Kristijan Cafuta, Igor Klep, and Janez Povh. The tracial moment problem and trace-optimization of polynomials. *Mathematical Programming*, 137(1-2):557–578, 2013.

[BEFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[Bes07] Frédéric Besson. Fast reflexive arithmetic tactics the linear case and beyond. In *Proceedings of the 2006 international conference on Types for proofs and programs*, TYPES'06, pages 48–62, Berlin, Heidelberg, 2007. Springer-Verlag.

[BJ10] Nicolas Brisebarre and Maria Joldes, Mioara. Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 147–154, Münich, Germany, July 2010. ACM New York, NY, USA. 17 pages RRLIP2010-13 RRLIP2010-13.

[BM81] R. S. Boyer and J. S. Moore. Metafunctions: Proving them correct and using them efficiently as new proof procedures. In *The Correctness Problem in Computer Science*, pages 103–84. Academic Press, New York, 1981.

[BM09] Martin Berz and Kyoko Makino. Rigorous global search using taylor models. In *Proceedings of the 2009 conference on Symbolic numeric computation*, SNC '09, pages 11–20, New York, NY, USA, 2009. ACM.

[Bor97] Brian Borchers. Csdp, a c library for semidefinite programming., 1997.

[BRB96] Gilles Barthe, Mark Ruys, and Henk Barendregt. A two-level approach towards lean proof-checking, 1996.

[BTN01] Aharon Ben-Tal and Arkadi Semenovich Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[CBBH08] A. Cuyt, F. Backeljauw, and C. Bonan-Hamada. *Handbook of Continued Fractions for Special Functions*. SpringerLink: Springer e-Books. Springer, 2008.

[CD08] G. Calafiore and F. Dabbene. Reduced vertex set result for interval semidefinite optimization problems. *Journal of Optimization Theory and Applications*, 139:17–33, 2008. 10.1007/s10957-008-9423-1.

[CFS07] Sylvain Conchon, Jean-Christophe Filliâtre, and Julien Signoles. Designing a Generic Graph Library using ML Functors. In *The Eighth Symposium on Trends in Functional Programming*, volume TR-SHU-CS-2007-04-1, pages XII/1–13, New York, USA, April 2007. Seton Hall University.

[CGT11] Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Math. Program.*, 127(2):245–295, 2011.

[Che82] E.W. Cheney. *Introduction to Approximation Theory*. AMS Chelsea Publishing Series. AMS Chelsea Pub., 1982.

[Che09] Sylvain Chevillard. *Évaluation efficace de fonctions numériques - Outils et exemples*. These, Université de Lyon; Ecole normale supérieure de lyon - ENS LYON, July 2009.

[Cim09] J. Cimpric. A method for computing lowest eigenvalues of symmetric polynomial differential operators by semidefinite programming. *ArXiv e-prints*, June 2009.

[CJL09] Sylvain Chevillard, Mioara Joldes, and Christoph Lauter. Certified and fast computation of supremum norms of approximation errors. In *Proceedings of the 2009 19th IEEE Symposium on Computer Arithmetic*, ARITH '09, pages 169–176, Washington, DC, USA, 2009. IEEE Computer Society.

[CJL10] S. Chevillard, M. Joldes, and C. Lauter. Sollya: An environment for the development of numerical codes. In K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 28–31, Heidelberg, Germany, September 2010. Springer.

[CKP11] Kristijan Cafuta, Igor Klep, and Janez Povh. Ncsostools: a computer algebra system for symbolic and numerical computation with noncommutative polynomials. *Optimization Methods and Software*, 26(3):363–380, 2011.

[Con76] Alain Connes. Classification of injective factors. Cases II_1, II_∞, III_λ, $\lambda \neq 1$. *Ann. Math. (2)*, 104:73–115, 1976.

[Coq] The Coq Proof Assistant. `http://coq.inria.fr/`.

[dKL10] Etienne de Klerk and Monique Laurent. Error bounds for some semidefinite programming approaches to polynomial minimization on the hypercube. *SIAM J. on Optimization*, 20(6):3104–3120, October 2010.

[FM00] W. H. Fleming and W. M. McEneaney. A max-plus-based algorithm for a Hamilton-Jacobi-Bellman equation of nonlinear filtering. *SIAM J. Control Optim.*, 38(3):683–710, 2000.

[GAA$^+$] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A Machine-Checked Proof of the Odd Order Theorem.

[GAC12] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. Delta-complete decision procedures for satisfiability over the reals. *CoRR*, abs/1204.3513, 2012.

[GLV09] Nebojsa Gvozdenovic, Monique Laurent, and Frank Vallentin. Block-diagonal semidefinite programming hierarchies for 0/1 programming. *Oper. Res. Lett.*, 37(1):27–31, 2009.

[GM05] Benjamin Grégoire and Assia Mahboubi. Proving equalities in a commutative ring done right in coq. In Joe Hurd and Thomas F. Melham, editors, *TPHOLs*, volume 3603 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2005.

[GMQ11] Stephane Gaubert, William M. McEneaney, and Zheng Qu. Curse of dimensionality reduction in max-plus based approximation methods: Theoretical estimates and improved pruning algorithms. In *CDC-ECC*, pages 1054–1061. IEEE, 2011.

[Gon08] Georges Gonthier. Computer mathematics. chapter The Four Colour Theorem: Engineering of a Formal Proof, pages 333–333. Springer-Verlag, Berlin, Heidelberg, 2008.

[GST07] Amparo Gil, Javier Segura, and Nico M. Temme. *Numerical Methods for Special Functions*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1 edition, 2007.

[GT06] B. Grégoire and L. Thery. A purely functional library for modular arithmetic and its application for certifying large prime numbers. In U. Furbach and N. Shankar, editors, *Proceedings of IJCAR'06*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 423–437. Springer-Verlag, 2006.

[GTW06] Benjamin Grégoire, Laurent Théry, and Benjamin Werner. A computational approach to pocklington certificates in type theory. In Masami Hagiya and Philip Wadler, editors, *Functional and Logic Programming*, volume 3945 of *Lecture Notes in Computer Science*, pages 97–113. Springer Berlin Heidelberg, 2006.

[Hal94] Thomas C. Hales. A proof of the kepler conjecture. *Math. Intelligencer*, 16:47–58, 1994.

[Hal03] Thomas C. Hales. Some algorithms arising in the proof of the kepler conjecture. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and Computational Geometry*, volume 25 of *Algorithms and Combinatorics*, pages 489–507. Springer Berlin Heidelberg, 2003.

[Hal05] Thomas C. Hales. A proof of the Kepler conjecture. *Ann. of Math. (2)*, 162(3):1065–1185, 2005.

[Hal06] Thomas C. Hales. Introduction to the flyspeck project. In Thierry Coquand, Henri Lombardi, and Marie-Françoise Roy, editors, *Mathematics, Algorithms and Proofs*, number 05021 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[Hal13] Thomas C. Hales. The flyspeck project, 2013.

[Han06] Eldon R. Hansen. Sharpening interval computations. *Reliable Computing*, 12(1):21–34, 2006.

[Har07] John Harrison. Verifying nonlinear real formulas via sums of squares. In Klaus Schneider and Jens Brandt, editors, *Proceedings of the 20th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2007*, volume 4732 of *Lecture Notes in Computer Science*, pages 102–118, Kaiserslautern, Germany, 2007. Springer-Verlag.

[Hel13] H. A. Helfgott. Major arcs for Goldbach's theorem. *ArXiv e-prints*, May 2013.

[HG83] E.R. Hansen and R.I. Greenberg. An interval newton method. *Applied Mathematics and Computation*, 12(2-3):89 – 98, 1983.

[HLS08] Didier Henrion, Jean-Bernard Lasserre, and Carlo Savorgnan. POCP: a package for polynomial optimal control problems. Rapport LAAS n° 08647, September 2008.

[HP13] H. A. Helfgott and D. J. Platt. Numerical Verification of the Ternary Goldbach Conjecture up to 8.875e30. *ArXiv e-prints*, May 2013.

[KKW04a] Sunyoung Kim, Masakazu Kojima, and Hayato Waki. Generalized lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems, 2004.

[KKW04b] Masakazu Kojima, Sunyoung Kim, and Hayato Waki. Sparsity in sums of squares of polynomials, 2004.

[KLYZ12] Erich L. Kaltofen, Bin Li, Zhengfeng Yang, and Lihong Zhi. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients. *JSC*, 47(1):1–15, jan 2012. In memory of Wenda Wu (1929–2009).

[KO63] Anatolii Karatsuba and Yuri Ofman. Multiplication of Multidigit Numbers on Automata. *Soviet Physics-Doklady*, 7:595–596, 1963.

[KS08] I. Klep and M. Schweighofer. Sums of Hermitian Squares and the BMV Conjecture. *Journal of Statistical Physics*, 133:739–760, November 2008.

[Las] Jean B. Lasserre. Convergent sdp-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 17:822–843.

[Las01] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[Lau09] Monique Laurent. Matrix completion problems. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1967–1975. Springer, 2009.

[Lax58] P. D. Lax. Differential equations, difference equations and matrix theory. *Communications on Pure and Applied Mathematics*, 11(2):175–194, 1958.

[LHPT08] Jean-Bernard Lasserre, Didier Henrion, Christophe Prieur, and Emmanuel Trélat. Nonlinear Optimal Control via Occupation Measures and LMI-Relaxations. *Siam Journal on Control and Optimization*, 47:1643–1666, 2008.

[LP10] Jean B. Lasserre and Mihai Putinar. Positivity and optimization for semi-algebraic functions. *SIAM Journal on Optimization*, 20(6):3364–3383, 2010.

[LS04] E. H. Lieb and R. Seiringer. Equivalent Forms of the Bessis Moussa Villani Conjecture. *Journal of Statistical Physics*, 115:185–190, April 2004.

[LT13] JeanB. Lasserre and TungPhan Thanh. Convex underestimators of polynomials. *Journal of Global Optimization*, 56(1):1–25, 2013.

[Mas93] G.D. Maso. *An Introduction to Gamma-Convergence*. Birkhäuser, 1993.

[MC11] David Monniaux and Pierre Corbineau. On the generation of Positivstellensatz witnesses in degenerate cases. In Marko Van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving (ITP)*, volume 6898 of *Lecture Notes in Computer Science*, pages 249–264. Springer Verlag, August 2011.

[McE06] W. M. McEneaney. *Max-plus methods for nonlinear control and estimation*. Systems & Control: Foundations & Applications. Birkhäuser Boston Inc., Boston, MA, 2006.

[McE07] W. M. McEneaney. A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs. *SIAM J. Control Optim.*, 46(4):1239–1276, 2007.

[MDG08] W. M. McEneaney, A. Deshpande, and S. Gaubert. Curse-of-complexity attenuation in the curse-of-dimensionality-free method for HJB PDEs. In *Proc. of the 2008 American Control Conference*, pages 4684–4690, Seattle, Washington, USA, June 2008.

[Mel12] Guillaume Melquiond. Floating-point arithmetic in the coq system. *Information and Computation*, 216(0):14 – 23, 2012. <ce:title>Special Issue: 8th Conference on Real Numbers and Computers</ce:title>.

[Mes99] Frédéric Messine. Extensions of affine arithmetic: Application to unconstrained global optimization, 1999.

[MH03] J.C. Mason and D.D.C. Handscomb. *The Chebyshev Polynomials*. Chapman and Hall/CRC, 2003.

[Mon09] Tiago M. Montanher. Intsolver: An interval based toolbox for global optimization, 2009. Version 1.0, available from `www.mathworks.com`.

[Nag74] J. Nagata. *Modern general topology*. Bibliotheca mathematica. North-Holland Pub. Co., 1974.

[NN94] Y. Nesterov and A. Nemirovski. *Interior Point Polynomial Methods in Convex Programming: Theory and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[NS07] Jiawang Nie and Markus Schweighofer. On the complexity of putinar's positivstellensatz. *Journal of Complexity*, 23(1):135 – 150, 2007.

[NT12] T. Netzer and A. Thom. Hyperbolic Polynomials and Generalized Clifford Algebras. *ArXiv e-prints*, July 2012.

[PP08] Helfried Peyrl and Pablo A. Parrilo. Computing sum of squares decompositions with rational coefficients. *Theor. Comput. Sci.*, 409(2):269–281, 2008.

[PS03] Pablo A. Parrilo and Bernd Sturmfels. *Minimizing polynomial functions*, volume 60 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 83–99. Amer. Math. Soc., Providence, RI, 2003.

[Put93] M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[Roc70] R. T. Rockafellar. *Convex Analysis*. Princeton mathematical series. Princeton University Press, 1970.

[RTAL11] Cordian Riener, Thorsten Theobald, Lina Jansson Andrén, and Jean B. Lasserre. Exploiting symmetries in sdp-relaxations for polynomial optimization. *CoRR*, abs/1103.0486, 2011.

[Sch04] Markus Schweighofer. On the complexity of schmüdgen's positivstellensatz. *Journal of Complexity*, 20(4):529 – 543, 2004.

[Sch05] Markus Schweighofer. Optimization of polynomials on compact semialgebraic sets. *SIAM Journal on Optimization*, 15(3):805–825, 2005.

[SGJM10] Srinivas Sridharan, Mile Gu, Matthew R. James, and William M. McEneaney. Reduced-complexity numerical method for optimal gate synthesis. *Phys. Rev. A*, 82:042319, Oct 2010.

[SH13] Alexey Solovyev and Thomas C. Hales. Formal verification of nonlinear inequalities with taylor interval approximations. *CoRR*, abs/1301.1702, 2013.

[Sol] Alexey Solovyev. Formal computations and methods.

[Spi06] Arnaud Spiwack. Ajouter des entiers machine à coq. 2006.

[SSM05] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Scalable analysis of linear systems using mathematical programming. In Radhia Cousot, editor, *Proc. of Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 3385, pages 21–47, Paris, France, January 2005. Springer Verlag.

[Sta11] H. R. Stahl. Proof of the BMV Conjecture. *ArXiv e-prints*, July 2011.

[Stu98] Jos F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones, 1998.

[VB94] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1994.

[Vie90] G. X. Viennot. Trees. In *Mots*, Lang. Raison. Calc., pages 265–297. Hermès, Paris, 1990.

[WKK$^+$08] Hayato Waki, Sunyoung Kim, Masakazu Kojima, Masakazu Muramatsu, and Hiroshi Sugimoto. Algorithm 883 : Sparsepop — a sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Trans. Math. Softw.*, 35(2), 2008.

[WKKM06] Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17:218–242, 2006.

[YFN$^+$10] Makoto Yamashita, Katsuki Fujisawa, Kazuhide Nakata, Maho Nakata, Mituhiro Fukuda, Kazuhiro Kobayashi, and Kazushige Goto. A high-performance software package for semidefinite programs : Sdpa7. Technical report, Dept. of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 2010.

[Zum08] Roland Zumkeller. *Rigorous Global Optimization*. PhD thesis, Ècole Polytechnique, 2008.

Index

Abstract

The aim of this work is to certify lower bounds for real-valued multivariate functions, defined by semialgebraic or transcendental expressions and to prove their correctness by checking the certificates in the COQ proof system.

The application range for such a tool is widespread; for instance Hales' proof of Kepler's conjecture involves thousands of nonlinear inequalities.

The functions we are dealing with are nonlinear and involve semialgebraic operations as well as some transcendental functions like cos, arctan, exp, etc. Our general framework is to use different approximation methods to relax the original problem into a semialgebraic optimization problem. It leads to polynomial optimization problems which we solve by sparse sums of squares relaxations.

First, we implement a classical scheme in global optimization. Namely, we approximate univariate transcendental functions with best uniform degree-d polynomial estimators.

Then, we present an alternative method, which consists in bounding some of the constituents of the nonlinear function by suprema or infima of quadratic polynomials (max-plus approximation method, initially introduced in optimal control) with a carefully chosen curvature.

Finally, we improve this approximation algorithm, by combining the ideas of the maxplus estimators and of the linear template method developed by Manna et al. (in static analysis). The nonlinear templates control the complexity of the semialgebraic estimators at the price of coarsening the maxplus approximations. In that way, we arrive at a new - template based - global optimization method, which exploits both the precision of sums of squares/SDP relaxations and the scalability of abstraction methods.

We successfully implemented these approximation methods in a software package named `NLCertify`. This tool interleaves semialgebraic approximations with sums of squares witnesses to form certificates. It is interfaced with COQ and thus benefits from the trusted arithmetic available inside the proof assistant. This feature is used to produce, from the certificates, both valid underestimators and lower bounds for each approximated constituent.

We illustrate the efficiency of `NLCertify` with various examples from the global optimization literature, as well as tight inequalities issued from the Flyspeck project.