OVERVIEW OF DECOMPOLY

THANOS PAPAÏOANNOU

1. Introduction

`decompoly` is a program that takes as input a polynomial in one or several variables with rational coefficients, and attempts to decide if the polynomial is non-negative for all real values of its arguments by way of exhibiting an exact decomposition of the polynomial as a sum-of-squares-of-rational-functions (SOSRF) with rational coefficients, if one can be found. This sum-of-squares decomposition serves as an algebraic certificate that the polynomial is non-negative. It is known by Artin's proof of Hilbert's 17th problem that every real non-negative polynomial has an expression as a SOSRF with real coefficients. The code is available on GitHub.

The main idea of how `decompoly` works is as follows: Given a polynomial $p(x)$, we write

$$p(x) = v^T Q v \tag{1.1}$$

where v is a column vector of monomials, and Q is a symmetric matrix of real scalars, called a *Gram matrix* for p. If such a Q can be found which is positive semidefinite (PSD), then by factorising Q as a product of a permutation matrix, a lower-triangular matrix, a diagonal matrix, and their transposes ($Q = PLDL^T P^T$), then the above expression is equivalent to a SOS decomposition

$$p(x) = \sum_{j=1}^{k} d_j l_j^2 \tag{1.2}$$

where the l_j are linear combinations of the monomials.

Every polynomial may be written in terms of a Gram matrix given an appropriate basis set of monomials, but in most cases there is no unique Gram matrix because of algebraic dependencies among the monomials. The problem is to find a PSD Gram matrix among the symmetric matrices satisfying the appropriate linear equations to match $p(x)$. Such a problem may be phrased as a problem of *semidefinite programming*, the problem of optimisation over a convex set consisting of linear constraints and the constraint of lying inside the cone of positive semidefinite matrices, which is computationally tractable.

2. Details

`decompoly` therefore uses a mixed symbolic and numerical approach. The main step in the program is therefore to set up the linear constraints corresponding the the polynomial p, and implement a numerical semidefinite programming solver 'dsdp' in order to identify a PSD Gram matrix or determine if no such Gram matrix exists. If the dsdp solver returns a PSD Gram matrix $\tilde{Q}$, the program looks for a matrix Q consisting of rational numbers with bounded denominator close to $\tilde{Q}$ and such that Q is also PSD and satisfies the linear constraints. The $PLDL^T P^T$ decomposition applied to Q will then yield a SOS decomposition with rational coefficients.

The setup requires the determination of the vector of monomials. To keep the dimensionality of the problem as low as possible, we only wish to consider monomials that could appear in a SOS decomposition of p. The *Newton polytope* of p is the convex hull of the set of vector multi-indices of monomials appearing in p. It is sufficient to consider a basis of monomials for the terms in a SOS decomposition whose exponent indices lie inside the Newton polytope of p scaled down by a factor of one-half. The program first computes a list of all integer vectors inside this convex hull to within a small tolerance. Let m be the number of integer multi-indices in the Newton polytope of p and n be the number of integer multi-indices in 1/2-times the

E-mail address: `tdp@post.harvard.edu`.

Newton polytope. The Gram matrix Q is then a symmetric n-by-n matrix with $n(n+1)/2$ independent entries satisfying m equations of the form

$$\sum_{\beta+\beta'=\alpha} Q_{\beta,\beta'} = p_\alpha \tag{2.1}$$

where β, β', α are multi-indices corresponding to monomials, and p_α is the coefficient of p on the monomial with index α.

Our program sets up the SDP feasibility problem in the so-called *explicit*, or *image*, representation. We represent the space of allowable Gram matrices as

$$\{G(y) = G_0 + G_1 y_1 + \ldots G_L y_L | y_i \in \mathbf{R}\} \tag{2.2}$$

where the G_i are symmetric matrices, and the dimension $L = n(n+1)/2 - m$. Producing the list of $L+1$ symmetric matrices $\{G_i\}$ is equivalent to solving the set of m linear equations (2.1), but we can take advantage of the special structure of these equations (namely, that each entry of Q appears in exactly one equation) to read off the basis matrices directly.

Having determined the basis matrices $\{G_i\}$, we are ready to implement an SDP solver. There are several adjustable options in the solver function, and some theoretical comments are in order to explain the choice of objective function and tolerance. In principle, the matter of deciding if a polynomial may be represented as a SOS is an SDP feasibility problem. If the problem is strictly feasible, that is, if the intersection of the set of matrices $G(y)$ and the PSD cone has nonempty interior, then there is a k-dimensional set of feasible Gram matrices from which to choose. Given any numerical vector $\tilde{y}$ corresponding to a Gram matrix $G(\tilde{y})$ in the interior, there exists a vector y of rational numbers close to $\tilde{y}$ so that $G(y)$, necessarily in the space of linear constraints, is also PSD. The closer to the boundary of the PSD cone $G(\tilde{y})$ is, the better the approximation y must be, requiring larger denominators. Therefore, for the majority of non-negative polynomials with strictly feasible regions, we choose to solve a semidefinite programming problem with constraint

$$G(y) \geq \varepsilon I \tag{2.3}$$

where ε is a small positive quantity chosen to keep the smallest eigenvalue positive. We first try $\varepsilon = 10^{-3}$. At the same time, we may choose an objective function to minimise. Natural choices in this regime are the zero objective, or we can attempt to *maximise* the trace, as a proxy for keeping the eigenvalues away from 0.

3. Limitations

Such a procedure is theoretically justified in the general, strictly feasible case for producing rational SOS decompositions with not-too-large denominator (as long as the feasible set contains some matrices whose eigenvalues are larger than ε). However, such decompositions are necessarily of full rank, meaning the decomposition involves the maximum n terms. It is a separate question of theoretical interest to find SOS decompositions of minimal rank. Less-than-full-rank Gram matrices lie on the boundary of the PSD cone, and are more difficult to specify exactly. If the initial SDP solver fails to find a nondegenerate solution, a second test is performed with $\varepsilon = 0$ and with objective to *minimise* the trace. There are known examples of non-negative polynomials with rational coefficients whose space of feasible Gram matrices lies entirely on the boundary of the PSD cone, and admit no rational SOS decompositions. For these degenerate cases, the interior-point method SDP solver will not work, even if ε is set to 0. There is no single method for recognising or dealing with such cases. In the current version of the code, we partially address the issue by applying a standard polynomial factoring algorithm to p in advance, and then the problem of decomposing p as a sum-of-squares of polynomials is reduced to finding SOS decompositions for the factors of p occurring with odd exponents.

Another difficulty arises from the known discrepancy between the set of non-negative polynomials and those polynomials having SOS polynomial decompositions. In general, it is only guaranteed that a non-negative polynomial has a decomposition in terms of rational functions. In other words, there exist non-negative polynomials with no rational SOS decomposition, but whose product with a non-negative polynomial (a denominator) is a SOS. In general, it is not known how to find appropriate denominators for any non-negative polynomial. If the SDP solvers fail to converge or provide a certificate of infeasibility, `decompoly`

will try pre-multiplying the polynomial successively by powers of $(1 + x_1^2 + x_2^2 + \cdots + x_j^2)$, where $\{x_i\}_{i=1}^{j}$ are all of the variables up to a default maximal exponent of 3. This choice of denominator is motivated by the result that every strictly positive polynomial (or every homogeneous polynomial positive away from 0) may be written as an SOSRF with denominator a sufficiently large power of the sum of all variable squares. However, there are known examples for which there is no SOSRF with this denominator. Such polynomials are said to have 'bad points'. Currently, `decompoly` cannot recognise such polynomials as non-negative.

It is worth mentioning that although in some sense most non-negative polynomials are nondegenerate, most of the examples of non-negative polynomials that are of theoretical interest or that are readily verified as non-negative (either by appeal to known inequalities of real numbers, or because the polynomial is written as a sum or product of squares) are degenerate.

If the SDP solver returns a numerical solution, `decompoly` first employs a check to ensure that the Gram matrix returned is PSD by employing an eigensolver for symmetric matrices. In order to allow for numerical approximations of Gram matrices with zero eigenvalues, the numerical Gram matrix is accepted as long as its smallest eigenvalue is greater than a small negative tolerance, tol $= 10^{-7}$. We then find a matrix of rational numbers close to the numerically computed Gram matrix by finding the best rational approximation y to the numerical solution $\tilde{y}$ with denominator less than a specified amount. This rational matrix $G(y)$ (which necessarily satisfies the linear constraints required to be a Gram matrix for p) is then checked for being PSD. The method is different for a matrix of rational numbers; here we compute the characteristic polynomial and check that the signs of terms alternate. As currently written, the procedure is to first check for a PSD Gram matrix with small maximum denominator of 100, and then to calculate a rational vector with denominator based on the smallest eigenvalue if the first rational Gram matrix is not PSD. Note that if the output of the SDP solver is an approximation of a degenerate matrix, then there is no guarantee that any particular nearby rational Gram matrix will be PSD.

If a rational, PSD Gram matrix Q is found, then the $PLDL^TP^T$ factorisation can be read off from the LU factorisation, which can be performed symbolically on Q in sympy, which preserves the rationality.