Journal of Software for Algebra and Geometry

```
gap> g:= SymmetricGroup( 4 );
Sym( [ 1 .. 4 ] )
i5 : betti(t,Weights=>{1,0})      gap> tbl:= CharacterTable( g );;  HasIrr( tbl );
                                   false
          0 1  2  3 4              gap> tblmod2:= CharacterTable( tbl, 2 );
o5 = total: 1 4 13 14 4           BrauerTable( Sym( [ 1 .. 4 ] ), 2 )
       0: 1 .  .  . .              gap> tblmod2 = CharacterTable( tbl, 2 );
       1: . 2  2  4 2              true
       2: . 2  5  6 .              gap> tblmod2 = BrauerTable( tbl, 2 );
       3: . .  4  . 2              true
       4: . .  .  4 .              gap> tblmod2 = BrauerTable( tbl, 2 );
       5: . .  2  . .              true
o5 : BettiTally                    gap> tblmod2 = BrauerTable( tbl, 2 );
i6 : betti(t,Weights=>{0,1})       true
                                   gap> libtbl:= CharacterTable( "M" );
          0 1  2  3 4              CharacterTable( "M" )
o6 = total: 1 4 13 14 4            gap> CharacterTableRegular( libtbl, 2 );
       0: 1 .  .  . .              BrauerTable( "M", 2 )
       1: . 2  2  4 2              gap> BrauerTable( libtbl, 2 );
       2: . 2  5  6 .              fail
       3: . .  4  . 2              gap> CharacterTable( "Symmetric", 4 );
       4: . .  .  4 .              CharacterTable( "Sym(4)" )
       5: . .  2  . .              gap> ComputedBrauerTables( tbl );
o6 : BettiTally                    [ , BrauerTable( Sym( [ 1 .. 4 ] ), 2 ) ]
i7 : t1 = betti(t,Weights=>{1,1})
          0 1  2  3 4              ring r1 = 32003,(x,y,z),ds;
o7 = total: 1 4 13 14 4            int a,b,c,t=11,5,3,0;
       0: 1 .  .  . .              poly f = x^a+y^b+z^(3*c)+x^(c+2)*y^(c-1)+x^
       1: . .  .  . .                      x^(c-2)*y^c*(y^2+t*x)^2;
       2: . .  .  . .              option(noprot);
       3: . 2  .  . .              timer=1;
       4: . .  .  . .              ring r2 = 32003,(x,y,z),dp;
       5: . 2  .  . .              poly f=imap(r1,f);
       6: . .  1  . .              ideal j=jacob(f);
       7: . .  8  6 .              vdim(std(j));
       8: . .  4  8 4              ==> 536
o7 : BettiTally                    vdim(std(j+f));
i8 : peek t1                       ==> 195
                                   timer=0;  // reset timer
o8 = BettiTally{(0, {0, 0}, 0) => 1 }
              (1, {2, 2}, 4) => 2
              (1, {3, 3}, 6) => 2
              (2, {3, 7}, 10) => 2
              (2, {4, 4}, 8) => 1
              (2, {4, 5}, 9) => 4
              (2, {5, 4}, 9) => 4
              (2, {7, 3}, 10) => 2
              (3, {4, 7}, 11) => 4
              (3, {5, 5}, 10) => 6
              (3, {7, 4}, 11) => 4
              (4, {5, 7}, 12) => 2
              (4, {7, 5}, 12) => 2
```

Sums of squares in Macaulay2

Diego Cifuentes, Thomas Kahle and Pablo Parrilo

Sums of squares in Macaulay2

DIEGO CIFUENTES, THOMAS KAHLE AND PABLO PARRILO

ABSTRACT: The *Macaulay2* package `SumsOfSquares` decomposes polynomials as sums of squares. It is based on methods to rationalize sums-of-squares decompositions due to Parrilo and Peyrl. The package features a data type for sums-of-squares polynomials, support for external semidefinite programming solvers, and optimization over varieties.

1. INTRODUCTION. Let $\mathbb{K} = \mathbb{Q}$ or $\mathbb{K} = \mathbb{R}$ be the rational or real numbers and $R = \mathbb{K}[x_1, \ldots, x_n]$ be the polynomial ring. An element $f \in R$ is *nonnegative* if $f(x) \geq 0$ for all $x \in \mathbb{R}^n$, and f is a *sum of squares* (SOS) if there are polynomials $f_1, \ldots, f_m \in R$ and positive scalars $\lambda_1, \ldots, \lambda_m \in \mathbb{K}$ such that $f = \sum_i \lambda_i f_i^2$. The scalars are not necessary when the field is $\mathbb{K} = \mathbb{R}$. Clearly, a sum of squares is nonnegative, but not every nonnegative polynomial is a sum of squares. Hilbert showed that the nonnegative polynomials of degree d in n variables are sums of squares if and only if: $n = 1$; or $d = 2$; or $n = 2$ and $d = 4$. For an introduction to the area we recommend [Scheiderer 2009, Blekherman et al. 2013].

The `SumsOfSquares` package contains methods to compute sums of squares in [Macaulay2]. A particular focus is on trying to find rational sums-of-squares decompositions of polynomials with rational coefficients (whenever they exist).

Consider the basic problem of deciding whether a polynomial is a sum of squares. Let f be an element of R of degree $2d$, and $v \in R^N$ be a vector whose entries are the $N = \binom{n+d}{d}$ monomials of degree $\leq d$. The following fundamental result holds:

$$f \text{ is SOS} \quad \Longleftrightarrow \quad \text{there exists } Q \in \mathbb{S}_+^N \text{ such that } f = v^T Q v,$$

where $\mathbb{S}_+^N$ is the cone of $N \times N$ symmetric positive semidefinite matrices; see [Blekherman et al. 2013, Section 3.1]. This reduces the problem to finding a *Gram matrix* Q as above, which can be done efficiently with *semidefinite programming* (SDP).

The method `solveSOS` performs the computation above. We use it here to verify that

$$f = 2x^4 + 5y^4 - 2x^2y^2 + 2x^3y$$

is a sum of squares:

```
i1 : R = QQ[x,y];

i2 : f = 2*x^4+5*y^4-2*x^2*y^2+2*x^3*y;

i3 : sol = solveSOS f;
Executing CSDP
Status: SDP solved, primal-dual feasible
```

The "Status" line indicates that a Gram matrix was found, so f is indeed a sum of squares. In the example above the package called an external program to serve as semidefinite programming solver. The default solver is the open source program CSDP [Borchers 1999], which is included in *Macaulay2*. The output of `solveSOS` is an object of type `SDPResult`. It contains, in particular, the Gram matrix Q and the monomial vector v.

```
i4 : (Q,v) = ( sol#GramMatrix, sol#Monomials )

o4 = ( | 2        1      -83/40 |,   | x2 | )
       | 1        43/20  0       |    | xy |
       | -83/40   0      5       |    | y2 |
```

The result of the semidefinite programming solver is a floating point approximation of the Gram matrix. The `SumsOfSquares` package attempts to find a close enough rational Gram matrix by rounding its entries [Peyrl and Parrilo 2008]. If this rounding procedure fails to find a feasible rational matrix, the method returns the floating point solution. The procedure is guaranteed to work when the floating point Gram matrix lies in the interior of $\mathbb{S}^N_+$. See the Appendix for more details about rational rounding.

The method `sosPoly` extracts the sum-of-squares decomposition from the returned `SDPResult`. This is done via an LDL factorization (a variant of Cholesky factorization) of the Gram matrix. For the function f from above we get three squares:

```
i5 : s = sosPoly sol

            83  2     2 2     43  20  2        2     231773   2 2
o5 = (5)(- ---x  + y )  + (--)(--x  + x*y)  + (------)(x )
           200             20  43               344000
```

The output above is an object of type `SOSPoly`. An object of this type stores the coefficients λ_i and polynomials (or generators) f_i such that $f = \sum_i \lambda_i f_i^2$. We can extract the coefficients and generators as follows:

```
i5 : coefficients s

o5 = {5, 43/20, 231773/344000}

i6 : gens s

o6 = {-83/200*x^2 + y^2, 20/43*x^2 + x*y, x^2}
```

The method `solveSOS` can also compute sums-of-squares decompositions in quotient rings. This can be useful to prove nonnegativity of a polynomial on a variety. We take an example from [Parrilo 2005]. Consider proving that $f = 10 - x^2 - y$ is nonnegative on the circle defined by $g = x^2 + y^2 - 1$. To do this, we check if f is a sum of squares in the quotient ring $\mathbb{Q}[x, y]/\langle g \rangle$. For such a computation, an even degree bound must be given by the user, as otherwise it is not obvious how to choose the monomial vector v. In the following example we use $2d = 2$ as the degree bound.

```
i1 : R = QQ[x,y]/ideal(x^2 + y^2 - 1);

i2 : f = 10-x^2-y;   d = 1;

i3 : sosPoly solveSOS (f, 2*d, TraceObj=>true)
Executing CSDP
Status: SDP solved, primal-dual feasible

            1       2     35     2
o3 = (9)(- -- y + 1)  + (--)(y)
           18                 36
```

In the computation above the option `TraceObj=>true` was used to reduce the number of squares in the SOS decomposition (see Section 6).

2. SUMS OF SQUARES IN IDEALS. Let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ be an ideal. Given an even bound $2d$, consider the problem of finding a nonzero sum-of-squares polynomial of degree $\leq 2d$ in the ideal I. If one of the generators of I has degree $\leq d$, then the problem is trivial. But otherwise the problem can be hard. The method `sosInIdeal` can be used to solve it. One of the main motivations for this problem is that it reveals information about the *real radical* of the ideal I, i.e., the vanishing ideal of the real zeros of I. Indeed, if $f = \sum \lambda_i f_i^2 \in I$ then each of the factors f_i must lie in the real radical of I.

Given generators of the ideal $I = \langle h_1, \ldots, h_m \rangle$, we may solve this problem by looking for some polynomial multipliers $l_i(x)$ such that $\sum_i l_i(x)h_i(x)$ is a sum of squares. The method `sosInIdeal` can find these multipliers. The input is a matrix containing the generators, and the degree bound $2d$. We illustrate this for the ideal $I = \langle x^2-4x+2y^2, 2z^2-y^2+2 \rangle$:

```
i1 : R = QQ[x,y,z];   d = 1;

i2 : h = matrix {{x^2-4*x+2*y^2, 2*z^2-y^2+2}};

i3 : (sol,mult) = sosInIdeal (h, 2*d);

i4 : sosPoly sol

        395      1       2    395     2
o4 = (---)(- - x + 1)  + (---)(z)
         2       2              2

i5 : h * mult == sosPoly sol

o5 = true
```

An alternative way to approach this problem is to construct the quotient $S = \mathbb{K}[x_1, \ldots, x_n]/I$ and then write $0 \in S$ as a sum of squares. In this case the input to `sosInIdeal` is simply the quotient ring S.

```
i6 : S = R/ideal h;

i7 : sosPoly sosInIdeal (S, 2*d);

        1031833      1       2    1031833     2
o7 = (-------)(- - x + 1)  + (-------)(z)
         2048        2              2048
```

In both cases we obtained a multiple of the sum-of-squares polynomial $\left(\frac{1}{2}x-1\right)^2+z^2$. This computation reveals that $x-2$ and z lie in the real radical of I. Indeed, we have $\sqrt[\mathbb{R}]{I} = \langle x-2, z, y^2-2 \rangle$.

3. SOS DECOMPOSITIONS OF TERNARY FORMS. Hilbert showed that any nonnegative form $f \in \mathbb{K}[x, y, z]$ can be decomposed as a quotient of sums of squares. We can obtain this decomposition

by iteratively calling `sosInIdeal`. Specifically, one can first find a multiplier q_1 such that $q_1 f$ is a sum of squares. Since q_1 is also nonnegative, we can then search for a multiplier p_1 such that $p_1 q_1$ is a sum of squares, and so on. The main observation is that the necessary degree of p_1 is lower than that of q_1 [de Klerk and Pasechnik 2004]. Hence this procedure terminates, and we can write

$$f = \frac{p_1 \cdots p_s}{q_1 \cdots q_t} \qquad \text{with } p_i, q_i \text{ sums of squares.}$$

As an illustration, we write the Motzkin polynomial as a quotient of sums of squares. We first use the function `library`, which contains a small library of interesting nonnegative forms.

```
i1 : R = QQ[x,y,z]

i2 : f = library ("Motzkin", {x,y,z})
       4 2     2 4      2 2 2    6
o2 = x y   + x y   - 3x y z   + z
```

We now apply the function `sosdecTernary`, which implements the iterative algorithm from above.

```
i3 : (Nums,Dens) = sosdecTernary f;
Executing CSDP

i4 : num = first Nums
      2267   2 2    4 2     2003     1013 3    990    3      2 2
o4 = (----)(x y  - z )  + (----)(- ----x y - ----x*y + x*y*z )  + ...
       64                   64     2003      2003

i5 : den = first Dens
      2267    2    1079    2    33    2
o5 = (----)(z)  + (----)(x)  + (--)(y)
       64           64          2
```

The result consists of two sums of squares, the second being the denominator. We can check the computation as follows.

```
i6 : f*value(den) == value(num)
o6 = true
```

4. PARAMETRIC SOS PROBLEMS. The `SumsOfSquares` package can also solve parametric problems. Assume now that $x \mapsto f(x; t)$ is a polynomial function that depends affinely on some parameters t. The command `solveSOS` can be used to search for values of the parameters such that the polynomial is a sum of squares. In the following example, we change two coefficients of the Robinson polynomial so that it becomes a sum of squares.

```
i1 : R = QQ[x,y,z][s,t];

i2 : g = library("Robinson", {x,y,z}) + s*x^6 + t*y^6;

i3 : sol = solveSOS g;
Executing CSDP
Status: SDP solved, primal-dual feasible

i4 : sol#Parameters
o4 = | 34 |
     | 34 |
```

In the code above, the ring construction (first line) indicates that s, t should be treated as parameters. The values obtained were $s = t = 34$.

It is also possible find the values of the parameters that optimize a given linear function. This allows us to find lower bounds for a polynomial function $f(x)$, by finding the largest t such that $f(x) - t$ is a sum of squares. Here we apply this method to the dehomogenized Motzkin polynomial.

```
i1 : R = QQ[x,z][t];

i2 : f = library ("Motzkin", {x,1,z});

i3 : sol = solveSOS (f-t, -t, RoundTol=>12);
Executing CSDP
Status: SDP solved, primal-dual feasible

i4 : sol#Parameters

o4 = | -729/4096 |
```

Alternatively, the method `lowerBound` can be called with input $f(x)$. The method internally declares a new parameter t and optimizes $f(x) - t$.

```
i1 : R = QQ[x,z];

i2 : f = library ("Motzkin", {x,1,z});

i3 : (t,sol) = lowerBound (f, RoundTol=>12);
Executing CSDP
Status: SDP solved, primal-dual feasible

i4 : t

o4 = - 729/4096
```

5. POLYNOMIAL OPTIMIZATION.

In applications one often needs to find lower bounds for polynomials subject to some polynomial constraints. More precisely, consider the problem

$$\min_{x \in \mathbb{R}^n} f(x) \text{ such that } h_1(x) = \cdots = h_m(x) = 0,$$

where $f, h_1, \ldots, h_m$ are polynomials. The `SumsOfSquares` package provides two ways to compute a lower bound for such a problem. The most elegant approach is to construct the associated quotient ring, and then call `lowerBound`. This will look for the largest t such that $f(x) - t$ is a sum of squares (in the quotient ring). A degree bound $2d$ must be given by the user.

```
i1 : R = QQ[x,y]/ideal(x^2 - x, y^2 - y);

i2 : f = x - y;  d = 1;

i3 : (t,sol) = lowerBound(f,2*d);
Executing CSDP
Status: SDP solved, primal-dual feasible

i4 : t

o4 = -1

i5 : f - t == sosPoly sol

o5 = true
```

Calling `lowerBound` as above is conceptually simple, but requires knowledge of a Gröbner basis, which is computed when constructing the quotient ring. If no Gröbner basis is available there is an

alternative way to call `lowerBound` with just the equations $h_1, \ldots, h_m$ as the input. The method will then look for polynomial multipliers $l_i(x)$ such that $f(x) - t + \sum_i l_i(x)h_i(x)$ is a sum of squares. This may result in larger semidefinite programs and weaker bounds.

```
i1 : R = QQ[x,y];
i2 : f = x - y;  d = 1;
i3 : h = matrix{{x^2 - x, y^2 - y}};
i4 : (t,sol,mult) = lowerBound (f, h, 2*d);
Executing CSDP
Status: SDP solved, primal-dual feasible
i5 : t
o5 = -1
i6 : f - t + h*mult == sosPoly sol
o6 = true
```

Lower bounds for polynomial optimization problems critically depend on the degree bound chosen. While higher degree bounds lead to better bounds, the computational complexity escalates quite rapidly. Nonetheless, low degree SOS lower bounds often perform very well in applications. In some cases, the minimizer might also be recovered from the `SDPResult` with the method `recoverSolution`.

```
i7 : recoverSolution sol
o7 = {x => 1.77345e-9, y => 1}
```

6. OPTIONAL ARGUMENTS.

SDP Solver. The optional argument `Solver` is available for many package methods and a particular semidefinite programming solver can be picked by setting it. These solvers are interfaced via the auxiliary *Macaulay2* package [SemidefiniteProgramming]. The package provides interfaces to the open source solvers CSDP [Borchers 1999] and SDPA [Yamashita et al. 2003], and the commercial solver [MOSEK]. There is also a built-in solver in the *Macaulay2* language. In our experience CSDP and MOSEK give the best results. CSDP is provided as part of *Macaulay2* and configured as the default.

Rounding tolerance. The method `lowerBound` has the optional argument `RoundTol`, which specifies the precision of the rational rounding. Smaller values of `RoundTol` lead to rational matrices with smaller denominators but farther from the numerical solution. The rational rounding may be skipped by setting it to infinity.

Trace objective. The option `TraceObj` tells the semidefinite programming solver to minimize the trace of the Gram matrix. This is a known heuristic to reduce the number of squares in the SOS decomposition.

APPENDIX: RATIONAL ROUNDING. Sums-of-squares problems are solved numerically using a semidefinite programming solver, and afterwards the package attempts to round the floating point solution to rational numbers. We briefly describe the rounding procedure, which was proposed in [Peyrl and Parrilo 2008].

Let $f \in \mathbb{Q}[x_1, \ldots, x_n]$ and consider the affine space $\mathcal{L} := \{Q : v^T Q v = f\}$, where v is a given monomial vector. A Gram matrix is an element of $\mathcal{L} \cap \mathbb{S}_+^N$. The semidefinite programming solver returns a numerical matrix Q_n, an "approximate" Gram matrix, which may not lie exactly on $\mathcal{L}$. The rounding problem consists in finding a nearby Gram matrix Q_r with rational entries.

The procedure from [Peyrl and Parrilo 2008] consists of two steps. First, the entries of Q_n are rounded to a rational matrix Q_r'. Then Q_r is obtained as the orthogonal projection of Q_r' onto $\mathcal{L}$. The image of the projection is rational, lies in $\mathcal{L}$, but need not be positive semidefinite. We may ensure $Q_r \in \mathbb{S}_+^N$ if the numerical matrix Q_n is in the interior of $\mathbb{S}_+^N$ and sufficiently close to $\mathcal{L}$. More precisely, assume λ, the smallest eigenvalue of Q_n, is greater than the distance $\delta := \mathrm{dist}(Q_n, \mathcal{L})$. Then setting the rounding tolerance $\mathrm{dist}(Q_n, Q_r')$ smaller than $\sqrt{\lambda^2 - \delta^2}$ guarantees that $Q_r \in \mathbb{S}_+^N$; see [Peyrl and Parrilo 2008, Proposition 8].

SUPPLEMENT.   The online supplement contains version 2.1 of `SumsOfSquares`.

REFERENCES.

[Blekherman et al. 2013] G. Blekherman, P. A. Parrilo, and R. R. Thomas, *Semidefinite optimization and convex algebraic geometry*, edited by G. Blekherman et al., MOS-SIAM Series on Optimization **13**, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2013. MR Zbl

[Borchers 1999] B. Borchers, "CSDP, a C library for semidefinite programming", pp. 613–623 11/12 **1-4**, 1999. MR Zbl

[de Klerk and Pasechnik 2004] E. de Klerk and D. V. Pasechnik, "Products of positive forms, linear matrix inequalities, and Hilbert 17th problem for ternary forms", *European J. Oper. Res.* **157**:1 (2004), 39–45. MR Zbl

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[MOSEK] MOSEK ApS, "MOSEK Command Line Tools", software, available at https://docs.mosek.com/8.1/cmdtools.pdf.

[Parrilo 2005] P. A. Parrilo, "Exploiting algebraic structure in sum of squares programs", pp. 181–194 in *Positive polynomials in control*, edited by D. Henrion and A. Garulli, Lect. Notes Control Inf. Sci. **312**, Springer, 2005. MR Zbl

[Peyrl and Parrilo 2008] H. Peyrl and P. A. Parrilo, "Computing sum of squares decompositions with rational coefficients", *Theoret. Comput. Sci.* **409**:2 (2008), 269–281. MR Zbl

[Scheiderer 2009] C. Scheiderer, "Positivity and sums of squares: a guide to recent results", pp. 271–324 in *Emerging applications of algebraic geometry*, edited by M. Putinar and S. Sullivant, IMA Vol. Math. Appl. **149**, Springer, 2009. MR Zbl

[SemidefiniteProgramming] D. Cifuentes, T. Kahle, P. A. Parrilo, and H. Peyrl, "SemidefiniteProgramming", Macaulay2 package,, available at https://github.com/Macaulay2/M2/blob/master/M2/Macaulay2/packages/SemidefiniteProgramming/.

[Yamashita et al. 2003] M. Yamashita, K. Fujisawa, and M. Kojima, "Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0)", *Optim. Methods Softw.* **18**:4 (2003), 491–505. The Second Japanese-Sino Optimization Meeting, Part II (Kyoto, 2002). MR Zbl

DIEGO CIFUENTES:

diegcif@mit.edu
Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, United States

THOMAS KAHLE:

thomas.kahle@ovgu.de
Fakultät für Mathematik, Otto-von-Guericke Universität Magdeburg, Institut Algebra und Geometrie, Universitätsplatz 2, Magdeburg, Germany

PABLO PARRILO:

parrilo@mit.edu
Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, United States