Manual of SARAG (Some Algorithms in Real Algebraic Geometry)

SARAG was developed by Fabrizio Caruso under the scientific guidance of Marie-Françoise Roy at the University of Rennes 1, France, with support from the RAAG network, further developed at the University of Pisa, Italy, modified by Alexandre Le Meur, Mathieu Kohli and Marie-Françoise Roy at the University of Rennes 1. This manual has been written by Alexandre Le Meur and later modified by Marie-Françoise Roy.

Please report bugs to: fabrizio.caruso@posso.dm.unipi.it or marie-francoise.roy@univ-rennes1.fr

SARAG is closely connected to "Algorithms in Real Algebraic Geometry" by Saugata Basu, Richard Pollack, Marie-Françoise Roy Springer-Verlag, Berlin, 2003 (second edition 2006) which together with SARAG is available for download at http://name.math.univ-rennes1.fr/marie-francoise.roy/bpr-ed2-posted3.html

1 Introduction

1.1 The SARAG Library

Free open source Maxima library for some algorithms of real algebraic geometry.

SARAG provides functions for linear algebra, theory of subresultants, Cauchy Index and its application to real root counting, isolation of real roots, sign determination, Thom encodings as well as certificates of positivity in the univariate and multivariate case.

1.2 Requirements

– MAXIMA (5.9.2 or above) The library has been initially developed and tested with Maxima version 5.9.2 and 5.9.3. SARAG with the only exception of plotting (e.g., "drawTopology") also works on Maxima 5.9.1. The latest version of Maxima tested so far is Maxima 5.20.1. The latest version of Maxima is available on line at http://maxima.sourceforge.net/.

– GNUPLOT (3.7.x, 4.0.x or above) The library uses Gnuplot for plotting graphs and it has been tested successfully on Gnuplot 3.7.x and 4.0.x.

1.3 Loading the files

You can work directly in Maxima with the command load(sarag).

Another option is to download the last version on line that can be found on
http://perso.univ-rennes1.fr/marie-francoise.roy/bpr-ed2-posted3.html
in the compressed directory saragcur and loaded by load(saragcur).

The library is contained in the following files:

sarag.mac (it loads all the files, settings.mac (general settings), constants.mac, sarag_initialization.mac ,aliases.mac (name conventions), lowLevel.mac (low level routines), sarag_linear_algebra.mac (linear algebra and matrix manipulation) ,rootCounting.mac (real root counting), rootIsolation.mac (root isolation by De Casteljau method), signDetermination.mac (sign determination), intervalArithmetic.mac (interval arithmetic), topology.mac (topology of curves), certificatesOfPositivity (certificates of positivity in the univariate case), multiCertificatesOfPositivity (certificates of positivity in the multivaritate case), arag_test.mac (test file for the book "Algorithms in Real Algebraic Geometry") hard_test.mac (test files with mostly topology computations) and saragmanual.pdf (this manual).

ATTENTION:The variables of the different functions are explicited in the file named « variables ».

In order to load the library either load the file "saragcur.mac" that will load all the files if they are in the same directory indicated by path : load("path/saragcur.mac") or load each single file with the "load" Maxima command.

Example: if the library is contained in a folder named saragcur in the directory where you are working type
load("./saragcur/saragcur.mac");

- Load the library

```
(%i1) load("./saragcur/saragcur.mac");
```

In order to test the library you can use:
batch("path/arag_test.mac",test);
and
batch("path/hard_test.mac",test);
path must be replace by the path of the folder containing the library :
exemple :
batch("./saragcur/arag_test.mac",test)
batch("./saragcur/hard_test.mac",test);
If there are errors they are reported in arag_test.ERR and hard_test.ERR.
Dont pay attention to the output you get in Maxima, just open the files arag_test.ERR and hard_test.ERR.

- Test the library

```
(%i3) batch("./saragcur/arag_test.mac",test);
```

```
(%i5) batch("./saragcur/hard_test.mac",test);
```

1.4 Manual

file saragmanual.pdf

This manual describes the high level functions ("main functions") of the SARAG library and the most important auxiliary functions.

For more details and for the theory behind the algorithms we refer to "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy which together with SARAG is available for download at http://name.math.univ-rennes1.fr/marie-francoise.roy/bpr-ed2-posted3.html. For univariate certificates of positivity, to F. Boudaoud, F. Caruso M.-F. Roy Certificates of positivity in the Bernstein basis, Discrete and Computational Geometry 39 4 639-655 (2008), for multivariate certificates of positivity to R. Leroy, Certificats de positivité et minimisation polynomiale dans la base de Bernstein multivariée http://tel.archives-ouvertes.fr/tel-00349444/fr/. For zero-nonzero sign determination see D. Perrucci, M.-F. Roy. Zero-nonzero and real-nonreal sign determination, Linear Algebra and Its Applications 439 (2013), no. 10, pp. 3016-3030 (preliminary version,arXiv:1305.4131).

1.5 Naming conventions

file : aliases.mac

The names of main functions are formed by adding prefixes to specify the method/algorithm which is used and suffixes to specify an alternative version of the function or output.

When no prefix or no suffix is used the function with the default method/algorithm will be called as set in the file: "aliases.mac".

Auxiliary functions may not follow such conventions.

1.6 Form of the output

The output of all functions of the library is a Maxima expression. Maxima uses brakets "[", "]" to describe couples and lists (e.g. an open inteval (a,b) is described by a couple containing the ends, which in Maxima is "[a,b]" (rather unfortunately)).

2 Functions and syntax

In this Section, all functions are described, with syntax and arguments. For each function we use an example to show how the function works and what it returns.

For several functions the library provides different methods for computing the result. This is indicated in the syntax part of a descrition in the paragraph Method. The default method, sert is aliases.mac is indicated with a *. Methods not implemented yet are indicated in grey.

2.1 Linear algebra

The corresponding results are explained in "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy, mainly in Chapter 8.

file : sarag_linear_algebra.mac

This file contains functions related to Gaussian elimination and matrix manipulations (products, determinant, etc.)

2.1.1 Matrix product

Matrix Product:. Computes the product of two matrices

- Syntax
  Name : matrixProd(A,B)
  Input : A and B are lists of lists representing the rows of two matrices such that A has the same number of columns as B as rows
  Output : the row-column product matrix of A, B
- Example

```
(%i2) matrixProd([[1,0,1],[0,1,0],[1,1,0]],[[1,0,0],[0,1,0],[0,0,1]]);
```

(%o2) $[[1,0,1],[0,1,0],[1,1,0]]$

- Code
  The code can be found in `./saragcur/sarag_linearAlgebra.mac` □

2.1.2 Determinant, elimination

Main Functions :

Determinant:. Computes the determinant of a square matrix

- Syntax
  Name : det(M)
  Input : M is a list of lists representing the rows of a matrix
  Method :
  - Gauss [Gaussian elimination with no pivot optimization]
  - Bareiss [Bareiss method]*

  Output : the determinant of M
- Example

```
(%i2) det([[1,0,0],[0,1,0],[0,0,1]]);
(%o2) 1
(%i3) gaussDet([[1,0,0],[0,1,0],[0,0,1]]);
(%o3) 1
(%i4) bareissDet([[1,0,0],[0,1,0],[0,0,1]]);
(%o4) 1
```

- Code
  The code can be found in `./saragcur/sarag_linearAlgebra.mac` □

Elimination:. Computes the upper triangular equivalent form of a matrix

- Syntax
  Name : elim(M)
  Input : M is a list of lists representing the rows of a matrix
  Method :
  - Gauss [Gaussian elimination with no pivot optimization]
  - Bareiss [Bareiss method]

  Output : [t,c,z] where :
  i. t is an upper triangular equivalent form of M with possibly some zero rows computed by Gaussian elimination with divisions and columns exchanges
  ii. c is the list of couples describing the columns exchanges
  iii. z is the list of zero rows
- Example

```
(%i7) elim([[0,1,0],[0,0,1],[1,0,0]]);
```

(%o7) $[[[1,0,0],[0,1,0],[0,0,1]],[[0,1],[1,2]],[]]$

- Code
  The code can be found in `./saragcur/sarag_linearAlgebra.mac` □

2.1.3 Characteristic polynomial

Auxiliary Functions :

Polynomial From Newton :. Computes the polynomial corresponding to given Newton Sums

- Syntax
  Name : polyFromNewton(ns,var)
  Input : ns is a list containing the Newton sums of a given polynomial in variable var
  Output : the corresponding polynomial
- Example

```
(%i13) polyFromNewton([3,0,2,0],x);
```

(%o13) $x^3 - x$

- Code
  The code can be found in `./saragcur/sarag_linearAlgebra.mac`

□

Newton From Polynomial :. Computes the Newton Sums corresponding to a given Polynomial.

- Syntax
  Name : newtonFromPoly(P,var,len)
  Input : P is a polynomial in variable var, len is a non negative integer

Output : array containing the n+1 newton sums of P

- Example

```
(%i4) newtonFromPoly(x^3-x,x,3);
```

(%o4) $[3, 0, 2, 0]$

- Code

The code can be found in `./saragcur/sarag_linearAlgebra.mac` □

Main Functions :

Characteristic Polynomial:. Computes the characteristic polynomial of a square matrix

- Syntax

Name : charPol(A,var)
Input : A is a list of lists representing the rows of a matrix
var is the variable of the polynomial (for example x)
Method :

  - Gauss [Gaussian elimination with no pivot optimization]
  - Bareiss [Bareiss method]
  - Baby Giant [baby step, giant step trace-based method]*

Output : the characteristic polynomial of A in the indeterminate var

- Example

Note that the command string can be used to output the polynomial in a form more convenient for further computations

```
(%i6) charPol([[1,2,3,4], [2,3,4,5], [3,4,5,6], [5,6,7,8]],x);
```

(%o6) $x^4 - 17\,x^3 - 26\,x^2$

```
(%i6) string(charPol([[1,2,3,4], [2,3,4,5], [3,4,5,6], [5,6,7,8]],x));
```

```
(%o6) x^4-17*x^3+26*x^2
(%i7) babyGiantCharPol([[1,2,3,4], [2,3,4,5], [3,4,5,6], [5,6,7,8]],x);
(%o7) x^4 - 17 x^3 - 26 x^2
(%i8) gaussCharPol([[1,2,3,4], [2,3,4,5], [3,4,5,6], [5,6,7,8]],x);
(%o8) x^4 - 17 x^3 - 26 x^2
```

- Code
  The code can be found in `./saragcur/sarag_linearAlgebra.mac` □

Descartes Signature:. Computes the signature of a symmetric matrix by the Descartes' Rule applied to the characteristic polynomial

- Syntax
  Name : descartesSignature(M)
  Input : M is a list of lists representing a square symmetric matrix
  Output : the signature of M
- Example

```
(%i10) descartesSignature([[1,0,0],[0,1,0],[0,0,1]]);
(%o10) 3
```

- Code
  The code can be found in `./saragcur/sarag_linearAlgebra.mac` □

2.2 Root counting

file : rootCounting.mac

2.2.1 Signes changes

The corresponding results are explained in "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy, mainly in Chapter 2 and 9.

Sign Changes:. Counts the number of sign changes in a sequence of real numbers

- Syntax
  Name : signChanges(seq)
  Input : seq is a sequence of number
  Output : number of sign changes in the sequence
- Example

```
(%i36) signChanges([1,0,0,1,2,3,-8,0,1]);
(%o36) 2
```

- Code
  The code can be found in `./saragcur/rootCounting.mac`

□

Modified Sign Changes:. Slight modification of the number of sign changes in a sequence of real numbers taking into account existing zeroes in the sequence.

- Syntax
  Name : modifiedSignChanges(seq)
  Input : seq is a sequence of number
  Output : number of sign changes in the sequence. Warning! Some annulations count for modified sign changes!
- Example

```
(%i42) modifiedSignChanges([1,0,0,1,2,3,-8,0,1]);
(%o42) 4
```

- Code

The code can be found in `./saragcur/rootCounting.mac` □

Generalized Permancences Minus Variations:. Computes a generalization of the difference between Permanences and Variations of a sequence

- Syntax
  Name: genPermMVar(seq)
  Input: seq a sequence of elements in an ordered integral domain
  Output: PmV(seq)
- Example

```
(%i23) genPermMVar([1,0,0,1,2,3,-8,0,1]);
(%o23) 0
(%i24)
```

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

2.2.2 Signed remainder sequence

The corresponding results are explained in "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy, mainly in Chapter 2 and 8.

Signed Remainder Sequence:. Computes the signed remainder sequence of two polynomials

- Syntax
  Name : sRem(P,Q,var)
  Input : P and Q are polynomials
  var is the variable of the polynomials (for example x)

  Output : list containing signed remainder sequence

- Example

```
(%i38) sRem(x^11-x^10+1,11*x^10-10*x^9,x);
```

$$(\%o38)\quad \left[x^{11}-x^{10}+1,\,11\,x^{10}-10\,x^{9},\,\frac{10\,x^{9}-121}{121},\,-\frac{1331\,x-1210}{10},\,\frac{275311670611}{285311670611}\right]$$

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

Sturm Sequence:. Computes the Sturm sequence of a polynomial

- Syntax
  Name : sturmSequence(P,var)
  Input : P polynomial
  var is the variable of the polynomial (for example x)
  Output : the Sturm sequence of P (signed remainder sequence of P and its derivative)
- Example

```
(%i74) sturmSequence(x^4+x+2,x);
```

$$(\%o74)\quad \left[x^{4}+x+2,\,4\,x^{3}+1,\,-\frac{3\,x+8}{4},\,\frac{2021}{27}\right]$$

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

Extended Signed Remainder Sequence. Same as previous function but include also the corresponding sequence of cofactors □

2.2.3 Subresultants

The corresponding results are explained in "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy, mainly in Chapter 4 and 8.

Signed Subresultant Polynomials:. Computes the list of the signed subresultant polynomials of two polynomials

- Syntax
  Name : sSubResPol(P,Q,var)
  Input : P and Q are polynomials
  var is the variable of the polynomials (for example x)

  Output : [sSubRes] where sSubRes is a list containing the signed subresultant sequence

- Example

```
(%i44) sSubResPol(x^4+a*x^2+b*x+c, diff(x^4+a*x^2+b*x+c,x),x);
```

(%o44) $[x^4+a\,x^2+b\,x+c, 4\,x^3+2\,a\,x+b, -8\,a\,x^2-12\,b\,x-16\,c, 32\,a\,c\,x-36\,b^2\,x-8\,a^3\,x-48\,b\,c-4\,a^2\,b, 256\,c^3-128\,a^2\,c^2+144\,a\,b^2\,c+16\,a^4\,c-27\,b^4-4\,a^3\,b^2]$

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

Signed Subresultant Coefficients. Computes the signed subresultant coefficients of two polynomials

- Syntax
  Name : sSubResCoeff(P,Q,var)
  Input : P is a polynomial in variable var
  Output : the signed subresultant coefficients of the polynomials P and Q
- Example

```
(%i52) sSubResCoeff(x^3+a*x+b,diff(x^3+a*x+b,x),x);
```

(%o52) $[1, 3, -6\,a, -27\,b^2 - 4\,a^3]$

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

Extended Signed Subresultant Sequence:. Same as previous but includes also the corresponding sequence of cofactors

- Syntax
  Name : sSubResExt(P,Q,var)
  Input : P and Q are polynomials
  var is the variable of the polynomials (for example x)

  Output : [sSubRes, s, u, v] where

  1. sSubRes is a list containing the signed subresultant sequence
  2. u, v are the corresponding cofactors

- Example

```
(%i54) sSubResExt(x^4+b*x+c,4*x^3+b,x);
```

(%o54) $[[x^4 + b\,x + c, 4\,x^3 + b, -12\,b\,x - 16\,c, -36\,b^2\,x - 48\,b\,c, 256\,c^3 - 27\,b^4], [1, 0, -16, -48\,b, 144\,b^2\,x^2 - 192\,b\,c\,x + 256\,c^2, -1024\,c^3\,x^3 + 108\,b^4\,x^3 - 256\,b\,c^3 + 27\,b^5], [0, 1, 4\,x, 12\,b\,x, -36\,b^2\,x^3 + 48\,b\,c\,x^2 - 64\,c^2\,x - 27\,b^3, 256\,c^3\,x^4 - 27\,b^4\,x^4 + 256\,b\,c^3\,x - 27\,b^5\,x + 256\,c^4 - 27\,b^4\,c]]$

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

GCD free part. Computes the GCD of two polynomials P and Q as well as P divided by this GCD

- Syntax
  Name : gcdFreePart(P,Q,var)
  Input : P and Q are polynomials
  var is the variable of the polynomials (for example x)
  Output : [g,f] where
  1. g is a gcd(P,Q)
  2. f is a gcd-free part of P with respect to Q

  Name : gcdFreePartWithZ(P,Q,var)
  Input : P and Q are polynomials wiht integer coefficients
  var is the variable of the polynomials (for example x)
  Output : [g,f] where
  1. g is a gcd(P,Q)
  2. f is a gcd-free part of P with respect to Q

- Example

```
(%i58) gcdFreePart((x^2-1)*(x+2),(x+1)*x,x);
```

(%o58) $[2\,x+2, 2\,x^2+2\,x-4]$

```
(%i57) gcdFreePartWithZ((x^2-1)*(x+2),(x+1)*x,x);
```

(%o57) $[x+1, x^2+x-2]$

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

Sylvester Resultant. Computes the Sylvester Resultant of two polynomials

- Syntax
  Name : sylvesterResultant(P,Q,var) or resultant(P,Q,var)
  Input : P and Q are polynomials
  var is the variable of the polynomials (for example x)
  Output : the resultant of P and Q (i.e. the determinant of the Sylvester Matrix of P and Q)
- Example

```
(%i5) resultant(a*x^2+b*x+c,2*a*x+b,x);
```

(%o5) $4\,a^2\,c - a\,b^2$

```
(%i6) sylvesterResultant(a*x^2+b*x+c,2*a*x+b,x);
```

(%o6) $4\,a^2\,c - a\,b^2$

- Code
  The code can be found in `./saragcur/rootCounting.mac`

□

Discriminant. Computes the discriminant of a polynomial

- Syntax
  Name : discriminant(P,var)
  Input : P is a polynomial in variable var
  Output : the discriminant of the polynomial P (product of the square of the differences of the roots)
- Example

```
(%i2) discriminant(x^3+p*x+q,x);
```

(%o2) $-27\,q^2 - 4\,p^3$

- Code

The code can be found in `./saragcur/rootCounting.mac` □

Subdiscriminant. Computes the subdiscriminant coefficients of a polynomial

- Syntax
  Name : subDiscrCoeff(P,var)
  Input : P is a polynomial in variable var
  Output : the subdiscriminants of the polynomial P
- Example

```
(%i23) subDiscrCoeff(x^3+p*x+q,x);
```

(%o23) $[1, 3, -6\,p, -27\,q^2 - 4\,p^3]$

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

2.2.4 Root counting

The corresponding results are explained in "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy, mainly in Chapter 9.

file : rootcounting.mac
rootIsolation.mac (for De Casteljau-based's root counting method)

Main Functions :

Cauchy Index. Computes the Cauchy Index of a rational function

- Syntax
  Name : cauchyIndex(Q,P,var)
  Input : Q, P polynomials

var is the variable of the polynomials (for example x)

Method :

i. sRem [signed remainder sequence]

ii. sSubRes [signed subresultants]*

Output : Cauchy index of Q/P in the whole real line

- Example

```
(%i76) cauchyIndex( (x-5)*(x-4)*(x-2)*(x+1)*(x+2)*(x+4), (x-3)^2*(x-1)*(x+3),x);
(%o76) 0
(%i78) sRemCauchyIndex( (x-5)*(x-4)*(x-2)*(x+1)*(x+2)*(x+4), (x-3)^2*(x-1)*(x+3),x);
(%o78) 0
(%i79) sSubResCauchyIndex( (x-5)*(x-4)*(x-2)*(x+1)*(x+2)*(x+4), (x-3)^2*(x-1)*(x+3),x);
(%o79) 0
```

- Code

The code can be found in `./saragcur/rootCounting.mac` □

Tarski Query. Computes the Tarski Query of two polynomials which is the difference between the number of (real) roots of P where Q is positive and the number of roots of P where Q is negative □

Number of Roots. Computes the number of real roots of a polynomial

- Syntax

Name : numberOfRoots(P,var)

Input : P polynomial

var is the variable of the polynomial (for example x)

Method :

i. sRem [signed remainder sequence]

ii. sSubRes [signed subresultants]

iii. deCasteljau [De Casteljau method for isolation]*

iv. monomial [monomial method for isolation]

Output : number of roots of P in the whole real line

- Example

```
(%i2) numberOfRoots(x^2-2*x+1,x);
(%o2) 1
(%i4) sRemNumberOfRoots(x^2-2*x+1,x);
(%o4) 1
(%i5) sSubResNumberOfRoots(x^2-2*x+1,x);
(%o5) 1
(%i3) deCasteljauNumberOfRoots(x^2-2*x+1,x);
(%o3) 1
```

- Code

  The code can be found in `./saragcur/rootCounting.mac` for sSubRes and sRem methods and in `./saragcur/rootIsolation.mac` for Decasteljau's method □

Cauchy Index Between. Computes the Cauchy Index of a rational function between two real numbers

- Syntax

  Name : cauchyIndexBetween(Q,P,var,a,b)

Input : Q, P polynomials
var is the variable of the polynomials (for example x)
a and b are reals (a<b) or -INFINITY or +INFINITY
Method :

i. sRem [signed remainder sequence]

ii. sSubRes [signed subresultants]*

Output : Cauchy index of Q/P in the interval]a,b[

- Example

```
(%i95) cauchyIndexBetween( (x-5)*(x-4)*(x-2)*(x+1)*(x+2)*(x+4), (x-3)^2*(x-1)*(x+3),x,1,-1);
(%o95) 0
(%i98) sSubResCauchyIndexBetween( (x-5)*(x-4)*(x-2)*(x+1)*(x+2)*(x+4), (x-3)^2*(x-1)*(x+3),x,1,-1);
(%o98) 0
(%i99) sRemCauchyIndexBetween( (x-5)*(x-4)*(x-2)*(x+1)*(x+2)*(x+4), (x-3)^2*(x-1)*(x+3),x,1,-1);
(%o99) 0
```

- Code
  The code can be found in `./saragcur/rootCounting.mac` □

Tarski Query Between. Computes the Tarski Query of two polynomials between two real numbers

- Syntax
  Name : tarskiQueryBetween(Q,P,var,a,b)
  Input : Q, P polynomials
  var is the variable of the polynomials (for example x)

a and b are reals (a<b) or -INFINITY or +INFINITY

Method :

i. sRem [signed remainder sequence]

ii. sSubRes [signed subresultants]*

Output : Tarski query of Q,P in]a,b[

- Example

```
(%i103) tarskiQueryBetween(1,x^11-x^10+1,x,0,1);
(%o103) 0
(%i104) sSubResTarskiQueryBetween(1,x^11-x^10+1,x,0,1);
(%o104) 0
(%i105) sRemTarskiQueryBetween(1,x^11-x^10+1,x,0,1);
(%o105) 0
```

- Code

  The code can be found in `./saragcur/rootCounting.mac` □

Number of Roots Between. Computes the number of real roots of a polynomial between two real numbers

- Syntax

  Name : numberOfRootsBetween(P,var,a,b)

  Input : P polynomial

  var is the variable of the polynomial (for example x)

  a and b are reals (a<b) or -INFINITY or +INFINITY

  Method :

  i. sRem [signed remainder sequence]

ii. sSubRes [signed subresultants]

iii. deCasteljau [De Casteljau method for isolation]*

iv. monomial [monomial method for isolation]

Output : number of roots of P in the interval]a,b[

- Example

```
(%i29) numberOfRootsBetween(x^2-2,x,0,10);
(%o29) 1
(%i30) sSubResNumberOfRootsBetween(x^2-2,x,0,10);
(%o30) 1
(%i31) sRemNumberOfRootsBetween(x^2-2,x,0,10);
(%o31) 1
(%i32) deCasteljauNumberOfRootsBetween(x^2-2,x,0,10);
(%o32) 1
```

- Code

  The code can be found in `./saragcur/rootIsolation.mac`

□

Hankel Matrix. Computes the Hankel quadratic form associated to a sequence of numbers

- Syntax

  Name : hankelMatrix(seq)

  Input : sequence seq of odd length of elements of an integral domain

  Output : Hankel quadratic form for seq (represented by a square symmetric matrix)

- Example

```
(%i34) hankelMatrix([0,1,2,3,4]);
```

(%o34) $[[0,1,2],[1,2,3],[2,3,4]]$

- Code
  The code can be found in `./saragcur/rootCounting.mac`

□

Hankel Signature. Computes the signature of the Hankel quadratic form associated to a sequence

- Syntax
  Name : hankelSignature(seq)
  Input : sequence seq of odd length of elements of an integral domain
  Output : signature of the Hankel quadratic form for seq
- Example

```
(%i118) hankelSignature([0,1,2,3,4]);
```

(%o118) 0

```
(%i36) hankelSignature([0,0,0,0,-1]);
```

(%o36) -1

```
(%i38) hankelSignature([0,0,0,-1,a]);
```

(%o38) 0

- Code
  The code can be found in `./saragcur/rootCounting.mac`

□

Difference of positive and negative real part of the roots. Computes the difference between the number of roots a polynomail with positive real part and those with negative real part

- Syntax
  Name : posNegDiff(P,var)
  Input : P is a polynomial in variable var
  Output : difference between the number of roots of P with positive real part and those with negative real part
- Example

```
(%i115) posNegDiff((x^2-1)*(x^3-1),x);
(%o115) −1
```

- Code
  The code can be found in `./saragcur/rootCounting.mac`

□

2.2.5 Isolation of roots

The corresponding results are explained in Chapter 10 of "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy

file : rootIsolation.mac
The routines contained in this files deal with the problem of isolation of real roots by using the conversion to the Bernstein basis and De Casteljau's method.
Auxiliary Functions :

Cauchy Upper Bound. Computes the difference between the number of roots a polynomail with positive real part and those with negative real part

- Syntax
  Name : cauchyRootUpperBound(P,var)
  Input : P is a polynomial in variable var
  Output : the upper bound for the absolute values of all real roots of P
- Example

```
(%i8) cauchyRootUpperBound(x+2,x);
(%o8) 3
```

- Code
  The code can be found in `./saragcur/rootIsolation.mac`

□

Cauchy Lower Bound. Computes the lower bound for the absolute values of all real roots of a polynomial

- Syntax
  Name : cauchyRootLowerBound(P,var)
  Input : P is a polynomial in variable var
  Output : the lower bound for the absolute values of all non-zero real roots of P
- Example

```
(%i10) cauchyRootLowerBound(x+2,x);
```

$$\text{(\%o10)}\ \frac{2}{3}$$

- Code
  The code can be found in `./saragcur/rootIsolation.mac`

□

Prime Cauchy Upper Bound. Computes an (alternative) upper bound for the absolute values of all real roots of a polynomial

- Syntax
  Name : primeCauchyRootUpperBound(P,var)
  Input : P is a polynomial in variable var
  Output : (alternative) upper bound for the absolute values of all real roots of P
- Example

```
(%i41) primeCauchyRootUpperBound(x+2,x);
(%o41) 10
```

- Code
  The code can be found in `./saragcur/rootIsolation.mac` □

Prime Cauchy Lower Bound. Computes an (alternative) lower bound for the absolute values of all real roots of a polynomial P

- Syntax
  Name : primeCauchyRootLowerBound(P,var)
  Input : P is a polynomial in variable var
  Output : (alternative) lower bound for the absolute values of all non-zero real roots of P
- Example

```
(%i12) primeCauchyRootLowerBound(x+2,x);
```

(%o12) $\frac{2}{5}$

- Code
  The code can be found in `./saragcur/rootIsolation.mac`

□

Convert to Berstein. Computes the coefficients of a polynomial in the Berstein's Basis

- Syntax
  Name : convert2Bernstein(P,d,var,l,r)
  Input : P is a polynomial in variable var of degree at most d.
  l and r are the parameters of the Basis. (l<r)
  Output : list containing the coefficients of P in the Bernstein basis of degree d for l,r

- Example

```
(%i44) convert2Bernstein(x^2,2,x,0,1);
(%o44) [0, 0, 1]
```

- Code
  The code can be found in `./saragcur/rootIsolation.mac` □

Berntein Split. Computes the coefficients of P in the Berstein's Basis for l,m and mr, knowing those for l,r

- Syntax
  Name : bernsteinSplit(coeffList, l,r,m)
  Input : coeffList is the list of coefficients of a polynomial in Berstein basis for l,r, m is another number
  Output : [bern_lm, bern_mr] where
  1. bern_lm is a list containing the coefficients in the Bernstein basis of degree d for l,m
  2. bern_mr is a list containing the coefficients in the Bernstein basis of degree d for m,r
- Example

```
(%i1) string(bernsteinSplit([4,-6,7,10],0,1,1/2));
(%i2)  [[4,-1,-1/4,17/8],[17/8,9/2,17/2,10]]
```

- Code
  The code can be found in `./saragcur/rootIsolation.mac` □

Special Berntein Split. Computes integer multiples of the coefficients of P in the Berstein's Basis for l,(l+r)/2 and (l+r)/2,r knowing those for l,r

- Syntax
  Name : specialBernsteinSplit(coeffList, l,r)
  Input : coeffList is the list of coefficients of a polynomial in berstein basis for parameters l,r

Output : [bern_first,bern_second] where :

1. bern_first is a list containing the coefficients in the Bernstein basis for l,(l+r)/2 of 2^deg(P) P
2. bern_second is a list containing the coefficients in the Bernstein basis for (l+r)/2,r of 2^deg(P) P

- Example

```
(%i51) specialBernsteinSplit([4,-6,7,10],0,1);
```

(%o51) $[[32, -8, -2, 17], [17, 36, 68, 80]]$

- Code

  The code can be found in `./saragcur/rootIsolation.mac` □

Main Functions :

Isolate Roots. Computes a list containing points or open intervals which contain the roots of a given polynomial

- Syntax

  Name : isolateRoots(P,var)
  Input : P is a polynomial in variable var.
  Method :

  i. deCasteljau [De Casteljau method for root isolation]

  ii. monomial [root isolation in the monomial basis]

  Output : characterization of the roots of P by a list of elements of the form either

  1. [pt] describing a real root pt of P
  2. [a,b] describing the open interval (a,b) containing exactly one root of P

- Example

```
(%i7) isolateRoots(-33*x^3+69*x^2-30*x+4,x);
```

```
(%o7) [[1, 2]]
(%i8) isolateRoots(x^2-2*x+1,x);
(%o8) [[−4, 4]]
```

- Code
  The code can be found in `./saragcur/rootIsolation.mac` □

Isolate Roots Between. Computes a list containing points or open intervals which contain the roots of a given polynomial between two real numbers

- Syntax
  Name : isolateRootsBetween(P,var,a,b)
  Input : P is a polynomial in variable var.
  a,b two real numbers
  Method :
  i. deCasteljau [De Casteljau method for root isolation]
  ii. monomial [root isolation in the monomial basis]
  Output : list of elements of the form either
  1. [pt] describing the real root pt which is a root of P
  2. [c,d] describing the open interval (c,d) containing exactly a root of P
- Example

```
(%i50) isolateRootsBetween((x-2)^2-1,x,-4,4);
(%o50) [[0, 2], [2, 4]]
```

- Code
  The code can be found in `./saragcur/rootIsolation.mac`

□

Find Roots. Computes a list containing points of open intervals smaller than threshold which contain exactly one root of a given polynomial □

Find Roots Between. Computes a list containing points of open intervals smaller than threshold which contain exactly one root of a given polynomial and are between two real numbers

- Syntax
  Name : findRoots(P,var,threshold)
  Input : P is a polynomial in variable var.
  threshold is the interval in which the function search the roots.
  Method :
  i. deCasteljau [De Casteljau method for root isolation]
  ii. monomial [root isolation in the monomial basis]
  Output : list of elements of the form either
  1. [pt] describing the real root pt of P
  2. [a,b] describing the open interval "(a,b)" smaller than threshold and containing exactly a root of P
- Example

```
(%i53) findRootsBetween(x^3-1,x,1,-2,2);
(%o53) [[1]]
(%i54)
```

- Code

The code can be found in `./saragcur/rootIsolation.mac` □

Roots Sign. Computes a list containing couples describing the real roots of P and the sign of Q at these real roots

- Syntax

  Name : signsAtRoots(isInt,P,Q,var)

  Input : P,Q are polynomials in variable var.

  isolating list isInt for the real roots of P in the same form as in the output of "isolateRealRoots"

  Method :

  i. deCasteljau [De Casteljau method for root isolation]

  ii. monomial [root isolation in the monomial basis]

  Output : lists of couples describing a real root and the sign of Q at this real root

- Example

```
(%i55) polR:(x-2)*(x^2-1);
```

(%o55) $(x-2)\,(x^2-1)$

```
(%i56) isInt : deCasteljauIsolateRoots(polR,x);
```

(%o56) $[[-8,0],[0,2],[2]]$

```
(%i57) polQ:x^2-4;
```

(%o57) x^2-4

```
(%i58) signsAtRoots(isInt,polR,polQ,x);
```

(%o58) $[[[-2,0],-1],[[0,2],-1],[[2],0]]$

```
(%i59) deCasteljauSignsAtRoots(isInt,polR,polQ,x);
```

(%o59) $[[[-2,0],-1],[[0,2],-1],[[2],0]]$

- Code
  The code can be found in `./saragcur/rootIsolation.mac`

□

Evaluate signs. Evaluates the signs of two polynomials at their roots

- Syntax
  Name : evaluateSignsAtRoots(P,Q,var)
  Input : P,Q are polynomials in variable var.

  Method :

  i. deCasteljau [De Casteljau method for root isolation]

  ii. monomial [root isolation in the monomial basis]

  Output : [com,signNComP,signNComQ] where

  1. com is an isolating list for the common real roots of P and Q
  2. signNComP is an isolating list with signs of Q for the real roots of P that are not roots of Q
  3. signNComQ is an isolating list with signs of P for the real roots of Q that are not roots of P

- Example

```
(%i61) evaluateSignsAtRoots(x+1,x,x);
```

(%o61) $[[], [[[-2, 0], -1]], [[[-1, 1], 1]]]$

```
(%i62) deCasteljauEvaluateSignsAtRoots(x+1,x,x);
```

(%o62) $[[], [[[-2, 0], -1]], [[[-1, 1], 1]]]$

- Code

The code can be found in `./saragcur/rootIsolation.mac` □

2.2.6 Sign Determination

The corresponding results are explained in Chapter 10 of "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy edition 2 and edition 2016 on line http://name.math.univ-rennes1.fr/marie-francoise.roy/bpr-ed2-posted3.html.

file : signDetermination.mac and *quickSignDetermination.mac*
NOTE: When we refer to an algorithm that computes the Tarski query of polynomials we assume that it has the same input/output format as "tarskiQuery" (ses Section Root Counting)

Sign Determination. Computes the signs of list of polynomials in a given finite set (typically defined as the zero set of a polynomial)

- Syntax
  Name : signDetermination(polList,ptSet,sqAlg,var)
  Input : list polList of polynomials in var, a description of a finite set of points, an algorithm sqAlg to compute the Tarski query of polynomials
  Method :
  i. naive [brute force method on a huge matrix of signs]
  ii. smart [a method that uses much smaller matrices of signs]
  iii. quick[an improvement of smart based on BPR edition 2016 on line]

  Output : a list representing a subset of the all elements of {0,-1,1}^polList describing the possible signs of the polynomials in polList at ptSet
- Example

```
(%i10) signDetermination( [x,x+1,x-2], x^5-9*x^3-x^2+9, tarskiQuery,x);
(%o10) [[1, 1, 1], [1, 1, −1], [−1, −1, −1]]
```

```
(%i17) naiveSignDetermination( [x,x+1,x-2], x^5-9*x^3-x^2+9, sRemTarskiQuery,x);
```

(%o17) $[[-1,-1,-1],[1,1,-1],[1,1,1]]$

```
(%i18) smartSignDetermination( [x,x+1,x-2], x^5-9*x^3-x^2+9, tarskiQuery,x);
```

(%o18) $[[1,1,1],[1,1,-1],[-1,-1,-1]]$

```
(%i8) quickSignDetermination( [x,x+1,x-2], x^5-9*x^3-x^2+9, tarskiQuery,x);
```

(%o8) $[[1,1,1],[[1,1,1],[1,1,-1],[-1,-1,-1]]$

- Code
  The code can be found in `./saragcur/signDetermination.mac` and `./saragcur/quickSignDetermination.mac`

It is also possible to obtain the cardinals of realizable sign conditions together with their list.

- Syntax
  Name : signDeterminationwithcardinals(polList,ptSet,sqAlg,var)
  Input : list polList of polynomials in var, a description of a finite set of points, an algorithm sqAlg to compute the Tarski query
  Method :

    quick*

  Output : a list representing the cardinals of the realizable sign conditions of polList at pSet, a subset of the all elements of {0,-1,1}^polList describing the realizable signs of the polynomials in polList at ptSet and

□

- Example

```
(%i8) signDeterminationwithcardinals( [x,x+1], x^5-9*x^3-x^2+9, tarskiQuery,x);
```

(%o8) $[[2,1],[[1,1],[-1,-1]]]$

```
(%i3) quickSgnDeterminationwithcardinals( [x,x+1], x^5-9*x^3-x^2+9, tarskiQuery,x);
```

(%o8) $[[2,1],[[1,1],[-1,-1]]]$

- Code
  The code can be found in `./saragcur/quickSignDetermination.mac`

2.2.7 Zerononzero Determination

The corresponding results are explained in D. Perrucci, M.-F. Roy. Zero-nonzero and real-nonreal sign determination, Linear Algebra and Its Applications 439 (2013), no. 10, pp. 3016-3030 (preliminary version,arXiv:1305.4131).

It is based on a blackbox similar to the Tarski query computing the number of roots of a polynomial where the value of another polynomial is non zero.

- Syntax
  Name : invertibility(Q,P,var)
  Input : two univariate polynomials P and Q
  Method :

  gcd*

  Output : the number of complex roots of P where Q is nonzero
- Example

```
(%i10) invertibilityQuery(x,x*(x^5-9*x^3-x^2+9),x);
```

(%o17) 5

```
(%i17) gcdInvertibilityQuery( [x,x+1,x-2], x*(x^5-9*x^3-x^2+9), gcdInvertibilityQuery,x);
```

(%o17) 5

- Code
  The code can be found in `./saragcur/quickSignDetermination.mac`

- Syntax
  Name : zerononzeroDetermination(polylist,P,Qu,var)
  Input : list polList of polynomials in var, a description of a finite set of points, an algorithm Qu to compute the Invertibility query
  Method :

    quick*

  Output : a list representing a subset of the all elements of {0,-1,1}^polList describing the possible zero nonzero conditions of the polynomials in polList at the complex zeroes of P
- Example

```
(%i10) zerononzeroDetermination( [x,x+1,x-2], x*(x^5-9*x^3-x^2+9), invertibilityQuery,x);
```

(%o17) $[[0,1,1],[1,1,1]]$

```
(%i17) quickZerononzeroDetermination( [x,x+1,x-2], x*(x^5-9*x^3-x^2+9), gcdInvertibilityQuery,x);
```

(%o17) $[[0,1,1],[1,1,1]]$

- Code
  The code can be found in `./saragcur/quickSignDetermination.mac`
  The same thing can be done with the information on the cardinals
- Syntax
  Name : zerononzeroDeterminationwithcardinals(polylist,P,Qu,var)
  Input : list polList of polynomials in var, a description of a finite set of points, an algorithm Qu to compute the Invertibility query
  Method :

    quick*

  Output : a list representing a subset of the all elements of {0,-1,1}^polList describing the possible zero nonzero conditions of the polynomials in polList at the zeroes of P

- Example

```
(%i10) zerononzeroDeterminationwithcardinals( [x,x+1,x-2], x*(x^5-9*x^3-x^2+9), invertibilityQuery,x);
```

(%o17) $[[1,5],[[0,1,1],[1,1,1]]]$

```
(%i17) quickZerononzeroDeterminationwithcardinals( [x,x+1,x-2], x*(x^5-9*x^3-x^2+9),
       gcdInvertibilityQuery,x);
```

(%o17) $[[1,5],[[0,1,1],[1,1,1]]]$

- Code

  The code can be found in `./saragcur/quickSignDetermination.mac`

2.2.8 Thom encoding

The corresponding results are explained in Chapter 10 of "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy

file : signDetermination.mac and *quickSignDeterminaton.mac*

Thom Encoding. Computes the Thom encodings of the real roots of a polynomial, which is the list of signs taken by the derivatives

- Syntax

  Name : thomEncoding(P,var)

  Input : polynomial P in variable var

  Output : a list containing the Thom encoding of the real roots of P

- Example

```
(%i15) thomEncoding(x^2-2,x);
```

(%o15) $[[-1,1],[1,1]]$

- Code
  The code can be found in `./saragcur/signDetermination.mac` and `./saragcur/quickSignDetermination.mac`

□

Thom Compare. Compares the Thom encodings of the real roots of two polynomials

- Syntax
  Name : thomCompare(P,Q,var)
  Input : polynomials P,Q in variable var
  Output : a list containing elements of the form [owner,thomInf] with the following possibilities :
  1. [0,thomP,thomQ] for a common root of P with Thom encoding thomP and of Q with Thom encoding thomQ
  2. [1,thomP] for a root of P not of Q with Thom encoding thomP
  3. [2,thomQ] for a root of Q not of P with Thom encoding thomQ

  Moreover the corresponding real numbers are ordered.
- Example

```
(%i14) thomCompare( x^2-4, expand((x-2)*(x^3-1)),x);
```

(%o14) $[[1,[-1,1]],[2,[-1,0,1,1]],[0,[1,1],[1,1,1,1]]]$

- Code
  The code can be found in `./saragcur/signDetermination.mac`

□

Thom Encoding with signs. Computes the Thom encoding of P at each root and the sign of Q for each root

- Syntax

Name : thomEncodingWithSigns(P,Q,var)
Input : polynomials P,Q in variable var
Output : a list of couples containing for each root of P its Thom encodings and the corresponding sign of Q

- Example

```
(%i58) thomEncodingWithSigns(x^2-2,x-1,x);
(%o58) [[[1, 1], 1], [[−1, 1], −1]]
(%i59)
```

- Code
  The code can be found in `./saragcur/signDetermination.mac`

□

2.3 Topology

The corresponding results are explained in Chapter 11 of "Algorithms in Real Algebraic Geometry" by S. Basu, R. Pollack, M.-F. Roy

Topology:. Computes the topology of a plane curve

- Syntax
  Name: topology(curve,x,y).
  Input: the equation of a curve in two variables with rational coefficients and the name of the two variables.
  Output: a couple whose first element gives the value of a in the initial change of coordinates, the second describes the topology with respect to the new coordinates. It gives for each interval between the roots of the discriminant the number of branches above it and for each root x of the discriminant, a couple containing the number of points of the curve above x and the position of the unique critical point over x.
  More precisely, the output is $[a, [m_0, [n_1, c_1], ..., m_{r-1}, [n_r, c_r], m_r]]$.

  Name: drawTopology(T[2],x,y).

Input: the output T to tolology(curve,x,y).

Output: a piecewise linear curve homeomorpic to the zero set of the polynomial curve.

- Example

```
(%i70) examplecurve:2*y^3+(3*x-3)*y^2+(3*x^2-3*x)*y+x^3;
```

(%o70) $2\,y^3+(3\,x-3)\,y^2+(3\,x^2-3\,x)\,y+x^3$

```
(%i71) T:topology(examplecurve,x,y);
```

(%o71) $[0,[1,[2,2],3,[2,1],3,[2,2],1]]$

```
(%i72) drawTopology(T[2]);
```

(%o72) **true**

```
(%i73) vide:topology(x^2+y^2+1,x,y);
```

(%o73) $[0,[0]]$

```
(%i74) drawTopology(vide[2]);
plot2d: nothing to plot.
```

(%o74) **true**

```
(%i75) notgeneric:(y^2+x)*(y^2-x+1);
```

(%o75) $(y^2-x+1)\,(y^2+x)$

```
(%i76) B:topology(notgeneric,x,y);
```

(%o76) $[1,[2,[1,1],0,[1,1],2]]$

```
(%i77) drawTopology(B[2]);
```

(%o77) **true**

- Code

The code can be found in `./saragcur/topology.mac` □

2.4 Certificate of positivity

a) In the univariate case, the corresponding reference is F. Boudaoud, F. Caruso M.-F. Roy Certificates of positivity in the Bernstein basis, Discrete and Computational Geometry 39 4 639-655 (2008)
file : certificateOfPositivity.mac

Certificate of positivity. Decides whether a polynomial is positive or negative on an interval and provides a certificate

- Syntax
  Name : certificate(P,var,a,b)
  Input : P is a polynomial in variable var. a and b are real numbers
  Output :
  1. [positive, certificate] if P is positive
  2. [negative, certificate] if P is negative
  3. [false, [a], sqPol] if P(a) equal 0 (where sqPol is the separable part of P)
  4. [false, [c,d], sqPol] if sqPol(c)*sqPol(d)<0 (where sqPol is the separable part of P)
- Example

```
(%i79) certificate(x^4+(8*x-1)^2,x,-1,1);
```

(%o79) $\left[\text{positive}, \left[[[-1, 0], 3, [3936, 2160, 944, 240, 48]], \left[\left[0, \frac{1}{8}\right], 3, [196608, 98304, 32768, 0, 48]\right], \left[\left[\frac{1}{8}, 1\right], 3, [48, 384, 1608704, 4841472, 9830400]\right]\right]\right]$

```
(%i80) certificate((x-2)^2-1,x,-1,1);
```

(%o80) $[\textbf{false}, [[1], x^2 - 4\,x + 3]]$

```
(%i81) certificate((2*x^2-1)^2,x,-1,1);
```

(%o81) $[\mathbf{false}, [-1, 0], 4\,x^2 - 2]$

```
(%i82) certificate(-x^4-(8*x-1)^2,x,-1,1);
```

(%o82) $\left[\text{negative}, \left[[-1, 0], 3, [-3936, -2160, -944, -240, -48]\right], \left[\left[0, \frac{1}{8}\right], 3, [-196608, -98304, -32768, 0, -48]\right], \left[\left[\frac{1}{8}, 1\right], 3, [-48, -384, -1608704, -4841472, -9830400]\right]\right]$

- Code
  The code can be found in `./saragcur/certificateOfPositivity.mac`

□

b) In the multivariate case, the corresponding reference is R. Leroy, Certificats de positivité et minimisation polynomiale dans la base de Bernstein multivariée http://tel.archives-ouvertes.fr/tel-00349444/fr/
file : multiCertificateOfPositivity.mac

Bernstein coefficients on a simplex. Various annex functions expressing a polynomial in the Bernstein basis on a simplex.

- Syntax
  Name : compositions(d,k+1)
  Input : d and k are natural numbers
  Output : all the multiindices of sum d and length k+1

  Name : simplex2bar(V,vars)
  Input : a simplex V given by k+1 points, each of them described by its k coordinates
  Output : the k+1 barycentric coordinates definining the simplex

  Name : monom2bern(P,V,vars,d)

Input : P is a multivariate polynomial in a list of k variable vars, d is at least the degree of P, V is a simplex described by k+1 points with k coordinate

Output : the list of coefficients of P in the Bernstein basis of degree d on V(in the order corresponding to compositions d,k for the barycentric coordinates given by simplex2bar)

Name : deCasteljau(simpl,ber,deg,M)

Input : a simplex defined by k+1 points with k coordinates, Bernstein coefficients on this simplex of a polynomial of degree at most deg and the k coordinates of a point

Output : the list of coefficients of P in the Bernstein basis of degree d on the k+1 subsimplices defined by the initial simplex and the point

- Example

```
(%i84) Comp:compositions(2,3);
```

(%o84) $[[2,0,0],[1,1,0],[1,0,1],[0,2,0],[0,1,1],[0,0,2]]$

```
(%i85) L:simplex2bar([[0,0],[1,0],[0,1]],[x,y]);
```

(%o85) $[-y-x+1, x, y]$

```
(%i86) expand(sum(apply(multinomial_coeff,Comp[i])*prod(L[j]^Comp[i][j],j,1,length(L)),i,1,length(Comp)));:monom2bern
```

(%o86) 1

```
(%i88) B:monom2bern(9*y^2-24*x*y+12*y+16*x^2+16*x+5,[[0,0],[1,0],[0,1]],[x,y],2);
```

(%o88) $[5, 13, 11, 37, 7, 26]$

```
(%i89) expand(sum(B[i]*apply(multinomial_coeff,Comp[i])*prod(L[j]^Comp[i][j],j,1,length(L)),i,1,length(Comp)));:monom
       ;
```

(%o89) $9\,y^2-24\,x\,y+12\,y+16\,x^2+16\,x+5$

```
(%i91) deCasteljau([[0,0],[1,0],[0,1]],B,2,[1/2,1/2]);
```

(%o91) $\left[\left[\left[\left[\frac{1}{2},\frac{1}{2}\right],[1,0],[0,1]\right],\left[\frac{77}{4},22,\frac{33}{2},37,7,26\right]\right],\left[\left[[0,0],\left[\frac{1}{2},\frac{1}{2}\right],[0,1]\right],\left[5,12,11,\frac{77}{4},\frac{33}{2},26\right]\right],\left[\left[[0,0],[1,0],\left[\frac{1}{2},\frac{1}{2}\right]\right],\left[5,13,12,37,22,\frac{77}{4}\right]\right]\right]$

- Code
  The code can be found in `./saragcur/multiCertificateOfPositivity.mac`

□

Certificate of positivity in the multivariate case. Decides whether a polynomial is positive on a simplex and provides a certificate

- Syntax
  Name : multiCertificate(P,V,vars,d,sub,cert)
  Input : P is a multivariate polynomial in a list of k variable vars, d is at least the degree of P, V is a simplex described by k+1 points with k coordinates, sub is a subdivision method (bisection, bisection_castel, standard_triang2, bisection_max_pos)
  Output :
  1. [certificate] if P is positive i.e. a sequence of subsimplexes and Bernstein coefficients on the corresponding subsimplex
  2. [a, sign(P(a))] if P(a)$\leqslant$0
  3. in other cases the output is not satisfactory yet

  Name : drawMultiCertificate(list)
  Input : the certficate output by multiCertificate, when the polynomial is positive, in the bivariate case
  Output : a picture of the corresponding subdivision
- Example

```
(%i93) f:9*y^2-24*x*y+12*y+16*x^2-16*x+5;
```

(%o93) $9\,y^2-24\,x\,y+12\,y+16\,x^2-16\,x+5$

```
(%i94) C1:multiCertificate(f,[[0,0],[1,0],[0,1]],[x,y],2,bisection,pos);
```

(%o94) $$\left[\left[\left[[0,0],\left[\frac{1}{2},\frac{1}{2}\right],[0,1]\right],\left[5,4,11,\frac{13}{4},\frac{17}{2},26\right]\right],\left[\left[[0,0],\left[\frac{1}{2},0\right],\left[\frac{1}{2},\frac{1}{2}\right]\right],\left[5,1,4,1,1,\frac{13}{4}\right]\right],\left[\left[\left[\frac{1}{2},0\right],\left[\frac{3}{4},\frac{1}{4}\right],\left[\frac{1}{2},\frac{1}{2}\right]\right],\left[1,1,1,\frac{17}{16},\frac{5}{8},\frac{13}{4}\right]\right],\left[\left[\left[\frac{1}{2},0\right],[1,0],\left[\frac{3}{4},\frac{1}{4}\right]\right],\left[1,1,1,5,\frac{3}{2},\frac{17}{16}\right]\right]\right]$$

```
(%i95) drawMultiCertificate(C1,500);
```

(%o95) [gr2*d(polygon, polygon, polygon, polygon)]

```
(%i43) C2:multiCertificate(f,[[0,0],[1,0],[0,1]],[x,y],2,bisection_max_pos,pos);
```

(%o43) $$\left[\left[\left[[0,0],\left[\frac{7}{10},\frac{3}{10}\right],[0,1]\right],\left[5,\frac{6}{5},11,\frac{101}{100},\frac{3}{2},26\right]\right],\left[\left[\left[\frac{2}{5},0\right],[1,0],\left[\frac{7}{10},\frac{3}{10}\right]\right],\left[\frac{29}{25},\frac{1}{5},\frac{26}{25},5,\frac{4}{5},\frac{101}{100}\right]\right],\left[\left[[0,0],\left[\frac{2}{5},0\right],\left[\frac{7}{10},\frac{3}{10}\right]\right],\left[5,\frac{9}{5},\frac{6}{5},\frac{29}{25},\frac{26}{25},\frac{101}{100}\right]\right]\right]$$

```
(%i44) drawMultiCertificate(C2,500);
```

(%o44) [gr2d(polygon, polygon, polygon)]

```
(%i96) C3:multiCertificate(f,[[0,0],[1,0],[0,1]],[x,y],2,standard_triang2,pos);
```

(%o96) $$\left[\left[\left[\left[0,\frac{1}{2}\right],\left[\frac{1}{2},\frac{1}{2}\right],[0,1]\right],\left[\frac{53}{4},\frac{25}{4},\frac{37}{2},\frac{13}{4},\frac{17}{2},26\right]\right],\left[\left[\left[\frac{1}{2},0\right],\left[0,\frac{1}{2}\right],\left[\frac{1}{2},\frac{1}{2}\right]\right],\left[1,1,1,\frac{53}{4},\frac{25}{4},\frac{13}{4}\right]\right],\left[\left[[0,0],\left[\frac{1}{2},0\right],\left[0,\frac{1}{2}\right]\right],\left[5,1,8,1,1,\frac{53}{4}\right]\right],\left[\left[\left[\frac{1}{2},\frac{1}{4}\right],\left[\frac{3}{4},\frac{1}{4}\right],\left[\frac{1}{2},\frac{1}{2}\right]\right],\left[\frac{25}{16},\frac{13}{16},\frac{17}{8},\frac{17}{16},\frac{5}{8},\frac{13}{4}\right]\right],\left[\left[\left[\frac{3}{4},0\right],[1,0],\left[\frac{3}{4},\frac{1}{4}\right]\right],\left[2,3,\frac{5}{4},5,\frac{3}{2},\frac{17}{16}\right]\right],\left[\left[\left[\frac{3}{4},0\right],\left[\frac{1}{2},\frac{1}{4}\right],\left[\frac{3}{4},\frac{1}{4}\right]\right],\left[2,\frac{1}{4},\frac{5}{4},\frac{25}{16},\frac{13}{16},\frac{17}{16}\right]\right],\left[\left[\left[\frac{1}{2},0\right],\left[\frac{3}{4},0\right],\left[\frac{1}{2},\frac{1}{4}\right]\right],\left[1,1,1,2,\frac{1}{4},\frac{25}{16}\right]\right]\right]$$

```
(%i97) drawMultiCertificate(C3,500);
```

(%o97) [gr2*d(polygon, polygon, polygon, polygon, polygon, polygon, polygon)]

```
(%i98) multiCertificate(x^2-y^2-1,[[0,0],[1,0],[0,1]],[x,y],2,bisection,pos);
```

(%o98) $[[0,0],-1]$

```
(%i99) g:9*y^2-24*x*y+12*y+16*x^2-16*x+5+z^2;
```

(%o99) $z^2+9\,y^2-24\,x\,y+12\,y+16\,x^2-16\,x+5$

```
(%i100) M1:multiCertificate(g,[[0,0,0],[1,0,0],[0,1,0],[0,0,1]],[x,y,z],2,standard_triang2,pos);
```

(%o100) $$\left[\left[\left[\left[\left[0,0,\frac{1}{2}\right],\left[\frac{1}{2},0,\frac{1}{2}\right],\left[0,\frac{1}{2},\frac{1}{2}\right],[0,0,1]\right],\left[\frac{21}{4},\frac{5}{4},\frac{33}{4},\frac{11}{2},\frac{5}{4},\frac{5}{4},\frac{3}{2},\frac{27}{2},\frac{17}{2},6\right]\right],\left[\left[\left[0,\frac{1}{2},0\right],\left[\frac{1}{2},\frac{1}{2},0\right],[0,1,0],\left[0,\frac{1}{2},\frac{1}{2}\right]\right],\left[\frac{53}{4},\frac{25}{4},\frac{37}{2},\frac{53}{4},\frac{13}{4},\frac{17}{2},\frac{25}{4},26,\frac{37}{2},\frac{27}{2}\right]\right],\left[\left[\left[0,\frac{1}{2},0\right],\left[\frac{1}{2},\frac{1}{2},0\right],\left[\frac{1}{2},0,\frac{1}{2}\right],\left[0,\frac{1}{2},\frac{1}{2}\right]\right],\left[\frac{29}{4},\frac{9}{4},6,\frac{29}{4},\frac{5}{4},2,\frac{9}{4},\frac{21}{4},\frac{25}{4},\frac{15}{2}\right]\right],\left[\left[\left[0,\frac{1}{2},0\right],\left[0,0,\frac{1}{2}\right],\left[\frac{1}{2},0,\frac{1}{2}\right],\left[0,\frac{1}{2},\frac{1}{2}\right]\right],\left[\frac{29}{4},6,6,\frac{29}{4},\frac{21}{4},\frac{21}{4},\frac{25}{4},\frac{21}{4},\frac{25}{4},\frac{15}{2}\right]\right],\left[\left[\left[\frac{1}{2},0,0\right],\left[0,\frac{1}{2},0\right],\left[\frac{1}{2},\frac{1}{2},0\right],\left[\frac{1}{2},0,\frac{1}{2}\right]\right],\left[1,1,1,1,\frac{53}{4},\frac{25}{4},1,\frac{13}{4},1,\frac{5}{4}\right]\right],\left[\left[\left[\frac{1}{2},0,0\right],\left[0,\frac{1}{2},0\right],\left[0,0,\frac{1}{2}\right],\left[\frac{1}{2},0,\frac{1}{2}\right]\right],\left[1,1,1,1,\frac{85}{4},1,1,\frac{5}{4},\frac{5}{4},\frac{5}{4}\right]\right],\left[\left[[0,0,0],\left[\frac{1}{2},0,0\right],\left[0,\frac{1}{2},0\right],\left[0,0,\frac{1}{2}\right]\right],\left[5,1,8,5,1,1,1,\frac{53}{4},8,\frac{21}{4}\right]\right],\left[\left[\left[\frac{1}{2},0,\frac{1}{4}\right],\left[\frac{3}{4},0,\frac{1}{4}\right],\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right],\left[\frac{1}{2},0,\frac{1}{2}\right]\right],\left[\frac{17}{16},\frac{17}{16},\frac{17}{16},\frac{9}{8},\frac{33}{16},\frac{5}{16},\frac{9}{8},\frac{13}{8},\frac{9}{8},\frac{5}{4}\right]\right],\left[\left[\left[\frac{1}{2},\frac{1}{4},0\right],\left[\frac{3}{4},\frac{1}{4},0\right],\left[\frac{1}{2},\frac{1}{2},0\right],\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]\right],\left[\frac{25}{16},\frac{13}{16},\frac{17}{8},\frac{25}{16},\frac{17}{16},\frac{5}{8},\frac{13}{16},\frac{13}{4},\frac{17}{8},\frac{13}{8}\right]\right],\left[\left[\left[\frac{1}{2},\frac{1}{4},0\right],\left[\frac{3}{4},\frac{1}{4},0\right],\left[\frac{3}{4},0,\frac{1}{4}\right],\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]\right],\left[\frac{5}{4},\frac{3}{4},1,\frac{5}{4},\frac{5}{4},1,\frac{3}{4},\frac{17}{16},\frac{17}{16},\frac{21}{16}\right]\right],\left[\left[\left[\frac{3}{4},0,0\right],[1,0,0],\left[\frac{3}{4},\frac{1}{4},0\right],\left[\frac{3}{4},0,\frac{1}{4}\right]\right],\left[2,3,\frac{5}{4},2,5,\frac{3}{2},3,\frac{17}{16},\frac{5}{4},\frac{33}{16}\right]\right],\left[\left[\left[\frac{1}{2},\frac{1}{4},0\right],\left[\frac{1}{2},0,\frac{1}{4}\right],\left[\frac{3}{4},0,\frac{1}{4}\right],\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]\right],\left[\frac{5}{4},1,1,\frac{5}{4},\frac{17}{16},\frac{17}{16},\frac{17}{16},\frac{17}{16},\frac{17}{16},\frac{21}{16}\right]\right],\left[\left[\left[\frac{3}{4},0,0\right],\left[\frac{1}{2},\frac{1}{4},0\right],\left[\frac{3}{4},\frac{1}{4},0\right],\left[\frac{3}{4},0,\frac{1}{4}\right]\right],\left[2,\frac{1}{4},\frac{5}{4},2,\frac{25}{16},\frac{13}{16},\frac{1}{4},\frac{17}{16},\frac{5}{4},\frac{33}{16}\right]\right],\left[\left[\left[\frac{3}{4},0,0\right],\left[\frac{1}{2},\frac{1}{4},0\right],\left[\frac{1}{2},0,\frac{1}{4}\right],\left[\frac{3}{4},0,\frac{1}{4}\right]\right],\left[2,0,2,2,2,0,0,\frac{33}{16},\frac{33}{16},\frac{33}{16}\right]\right],\left[\left[\left[\frac{1}{2},0,0\right],\left[\frac{3}{4},0,0\right],\left[\frac{1}{2},\frac{1}{4},0\right],\left[\frac{1}{2},0,\frac{1}{4}\right]\right],\left[1,1,1,1,2,\frac{1}{4},1,\frac{25}{16},1,\frac{17}{16}\right]\right]\right]$$

```
(%i101) M2:multiCertificate(g,[[0,0,0],[1,0,0],[0,1,0],[0,0,1]],[x,y,z],2,bisection_max_pos,pos);
```

(%o101) $$\left[\left[\left[[0,0,0],\left[\frac{7}{10},\frac{3}{10},0\right],[0,1,0],[0,0,1]\right],\left[5,\frac{6}{5},11,5,\frac{101}{100},\frac{3}{2},\frac{6}{5},26,11,6\right]\right],\left[\left[[0,0,0],\left[\frac{3}{5},0,\frac{2}{5}\right],\left[\frac{7}{10},\frac{3}{10},0\right],[0,0,1]\right],\left[5,\frac{1}{5},\frac{6}{5},5,\frac{33}{25},\frac{24}{25},\frac{3}{5},\frac{101}{100},\frac{6}{5},6\right]\right],\left[\left[\left[\frac{2}{5},0,0\right],[1,0,0],\left[\frac{7}{10},\frac{3}{10},0\right],\left[\frac{3}{5},0,\frac{2}{5}\right]\right],\left[\frac{29}{25},\frac{1}{5},\frac{26}{25},\frac{21}{25},5,\frac{4}{5},\frac{9}{5},\frac{101}{100},\frac{24}{25},\frac{33}{25}\right]\right],\left[\left[[0,0,0],\left[\frac{2}{5},0,0\right],\left[\frac{7}{10},\frac{3}{10},0\right],\left[\frac{3}{5},0,\frac{2}{5}\right]\right],\left[5,\frac{9}{5},\frac{6}{5},\frac{1}{5},\frac{29}{25},\frac{26}{25},\frac{21}{25},\frac{101}{100},\frac{24}{25},\frac{33}{25}\right]\right]\right]$$

- Code
  The code can be found in `./saragcur/multiCertificateOfPositivity.mac` □