

SMRSOFT V1.2 Guide

Graziano Chesi

Department of Electrical and Electronic Engineering
University of Hong Kong

March 19, 2012

Contents

Abbreviations	4
Introduction	6
1 Preliminaries	9
1.1 Input Formats	9
1.2 Auxiliary Functions	12
1.2.1 smrplot	12
1.2.2 smrprint	13
Part I: Basic Optimization Problems	15
2 Positivity and Non-Positivity	17
2.1 smrsos	17
2.2 Examples: Positivity	18
2.3 Examples: Non-Positivity	20
2.4 Examples: Positivity of PNS Polynomials	20
2.5 Examples: Positivity and Non-Positivity over Semialgebraic Sets	22
2.6 Examples: Positivity of Homogeneous Polynomials	23
3 Minimum of Rational Functions	25
3.1 smropt	25
3.2 Examples: Unconstrained Variable	26
3.3 Examples: Variable Constrained over Semialgebraic Sets	27
4 Solving Systems of Polynomial Equations and Inequalities	29
4.1 smrtps	29
4.2 Examples: Systems of Polynomial Equations	30
4.3 Examples: Systems of Polynomial Equations and Inequalities	31
Part II: Dynamical Systems	33
5 Analysis and Control of the Domain of Attraction	35
5.1 smrda	35
5.2 Examples	38

Abbreviations

BMI Bilinear matrix inequality

DA Domain of attraction

EVP Eigenvalue problem

GEVP Generalized eigenvalue problem

LEDA Largest estimate of the domain of attraction

LERDA Largest estimate of the robust domain of attraction

LF Lyapunov function

LMI Linear matrix inequality

PNS Positive non sum of squares of polynomials

RDA Robust domain of attraction

SDP Semidefinite program, semidefinite programming

SMR Square matricial representation

SOS Sum of squares of polynomials

Introduction

SMRSOFT is a Matlab toolbox for solving basic optimization problems over polynomials and studying dynamical systems via SOS programming. This guide describes the functions available in SMRSOFT version 1.2, which are based on the techniques described in the book

G. Chesi, “Domain of Attraction: Analysis and Control via SOS Programming”, Springer, 2011.

For the part “basic optimization problems”, the main functions currently available are:

- **smrsos** : computes the generalized SOS index of a polynomial or a matrix polynomial, and investigates positivity and non-positivity of a polynomial or a matrix polynomial with unconstrained variables or variables constrained over a semialgebraic set.
- **smropt** : computes a lower bound of the minimum of a rational function with unconstrained variables or variables constrained over a semialgebraic set, and verifies the tightness of the found lower bound.
- **smrsps** : determines the solutions of a system of polynomial equations and inequalities.

For the part “dynamical systems”, SMRSOFT currently allows one to investigate the domain of attraction (DA) of an equilibrium point of a polynomial system through the following function:

- **smrda** : computes a lower bound of the largest estimate of the DA (LEDA) of an equilibrium point of a polynomial system for a given polynomial Lyapunov function (LF), verifies the tightness of the found lower bound, searches for optimal polynomial LFs maximizing the size of the LEDA, computes a polynomial output controller for enlarging the LEDA, and plots the found results in the case of second or third order polynomial systems.

For using SMRSOFT , just add the location of the folder “smrsoftV12” to the Matlab path (the subfolders do not need to be added). The subfolders “doc” and “examples” contain the documentation and the m-files of the examples reported in this guide.

SMRSOFT requires the Matlab toolbox SeDuMi. The examples reported in this guide have been tested with SeDuMi version 1.3. SMRSOFT can be downloaded from <http://www.eee.hku.hk/~chesi>

Chapter 1

Preliminaries

1.1 Input Formats

This section describes how polynomials, vector polynomials and matrix polynomials can be passed to the functions of SMRSOFT .

Polynomials: Text Input Type

In text input type, the polynomial is provided through text. For instance, the polynomial

$$f(x) = 1/2 - 2.7183x_1 + 3 \cdot 10^{-3}x_2 + 0.5(x_1 - x_2)^2(x_1 + x_2) + 4x_1^2x_2^3$$

can be provided in any of the following ways:

```
>> % quick
>> f='1/2-2.7183x1+3e-3x2+0.5(x1-x2)^2(x1+x2)+4x1^2x2^3';

>> % math standard
>> f='1/2-2.7183*x1+3e-3*x2+0.5*(x1-x2)^2*(x1+x2)+4*x1^2*x2^3';

>> % tex-like
>> f='1/2-2.7183x_1+3e-3x_2+0.5(x_1-x_2)^2(x_1+x_2)+4x_1^2x_2^3';
```

By default, the expected variables are

`x1, x2, x3, ...`

The label `x` of these variables can be changed through the field `var` of the struct `opts` that can be provided when calling a function of SMRSOFT . For instance, the polynomial above can be provided as follows:

```
>> f='1/2-2.7183y1+3e-3y2+0.5(y1-y2)^2(y1+y2)+4y1^2y2^3';
>> opts.var='y';
```

The text can contain constants whose values have to be provided through the field `conval` of the struct `opts`. By default, the expected constants are

`c1, c2, c3, ...`

and the label `x` of these variables can be changed through the field `con` of the struct `opts`. For instance, the polynomial above can be provided as follows:

```
>> % constants with common label: c1,c2,c3
>> opts.conval={1/2;-2.7183;3e-3};
>> f='c1+c2x1+c3x2+0.5(x1-x2)^2(x1+x2)+4x1^2x2^3';

>> % constants with different label: d1,q1,q2
>> opts.con='dq';
>> opts.conval={1/2;[-2.7183;3e-3]};
>> f='d1+q1x1+q2x2+0.5(x1-x2)^2(x1+x2)+4x1^2x2^3';
```

Limitations:

- only the following operators can be used: `+, -, *, /, ^` ;
- labels of variables or constants can contain one char only;
- divisors can be real numbers only;
- exponents can be either nonnegative integer numbers (whenever the base is a polynomial) or real numbers (whenever the base does not contain variables) only.

Polynomials: Struct Input Type

In struct input type, the polynomial is provided through a struct with two main fields: `c` and `B`. `c` is a column vector whose i -th entry is the coefficient of the i -th monomial, while `B` is a matrix whose i -th row contains the powers of the variables in the i -th monomial. For instance, the polynomial

$$f(x) = 1 - 2.3x_1 + \pi x_1^2 x_2^6$$

can be provided as follows:

```
>> f.c=[1;-2.3;pi];
>> f.B=[0 0;1 0;2 6];
```

Vector Polynomials: Text Input Type

In text input type, the vector polynomial is provided through text. For instance, the vector polynomial

$$f(x) = [2^{0.5}, x_1 - x_2, x_1 x_2]$$

can be provided as follows:

```
>> f='[2^0.5,x1-x2,x1x2]';
```

Limitations:

- operations among vector polynomials (such as sum, product, etc) are not admitted in text input type.

Vector Polynomials: Cell Input Type

In cell input type, the vector polynomial is provided through a cell, where each element of the cell is a polynomial (provided either in text input type or struct input type). For instance, the vector polynomial

$$f(x) = [2^{0.5}, x_1 - x_2, x_1 x_2]$$

can be provided in any of the following ways:

```
>> % text on each cell
>> f=[];f{1}='2^0.5';f{2}='x1-x2';f{3}='x1x2';

>> % text on each cell
>> f={'2^0.5','x1-x2','x1x2'};

>> % struct on each cell
>> f=[];
>> f{1}.c=2^0.5;f{1}.B=0;
>> f{2}.c=[1;-1];f{2}.B=[1 0;0 1];
>> f{3}.c=1;f{3}.B=[1 1];

>> % mixed
>> f=[];
>> f{1}='2^0.5';f{2}='x1-x2';
>> f{3}.c=1;f{3}.B=[1 1];
```

Matrix Polynomials: Text Input Type

In text input type, the matrix polynomial is provided through text. For instance, the (symmetric) matrix polynomial

$$f(x) = \begin{bmatrix} 1 & x_1 & 2 - x_2 \\ x_1 & x_1^2 & -2 \\ 2 - x_2 & -2 & x_1^2 - x_2^2 \end{bmatrix}$$

can be provided in any of the following ways:

```
>> % defining all the entries
>> f='[1,x1,2-x2;x1,x1^2,-2;2-x2,-2,x1^2-x2^2]';
```

```
>> % defining only the entries on or above the diagonal
>> f='[1,x1,2-x2;x1^2,-2;x1^2-x2^2]';

>> % defining only the entries on or below the diagonal
>> f='[1;x1,x1^2;2-x2,-2;x1^2-x2^2]';
```

Limitations:

- operations among matrix polynomials (such as sum, product, etc) are not admitted in text input type.

Matrix Polynomials: Cell Input Type

In cell input type, the matrix polynomial is provided through a cell, where each element of the cell is a polynomial (provided either in text input type or struct input type). For instance, the (symmetric) matrix polynomial

$$f(x) = \begin{bmatrix} 1 & x_1 \\ x_1 & x_2 \end{bmatrix}$$

can be provided in any of the following ways:

```
>> % text on each cell
>> f=[];f{1,1}='1';f{1,2}='x1';f{2,1}='x1';f{2,2}='x2';

>> % text on each cell
>> f={'1','x1';'x1','x2'}';

>> % struct on each cell
>> f=[];
>> f{1,1}.c=1;f{1,1}.B=0;
>> f{1,2}.c=1;f{1,2}.B=1;
>> f{2,1}.c=1;f{2,1}.B=1;
>> f{2,2}.c=1;f{2,2}.B=[0 1];

>> % mixed
>> f=[];
>> f{1,1}='1';f{1,2}='x1';f{2,1}='x1';
>> f{2,2}.c=1;f{2,2}.B=[0 1];
```

1.2 Auxiliary Functions

1.2.1 smrplot

The following function can be used to plot a level set of a polynomial in two or three dimensions:

```
smrplot(f,c,xran,xpnt,sty,gxy)
```

Plots the level set $f(x)=c$ in R^2 (if x is in R^2) or R^3 (if x is in R^3).

SMRPLOT INPUTS

f is a polynomial.

c is a real. DEFAULT: $c=0$.

$xran=[x1min\ x1max\ x2min\ x2max\ x3min\ x3max]$ defines the range for x .
DEFAULT: $r=[-3\ 3\ -3\ 3\ -3\ 3]$.

$xpnt=[nx1\ nx3]$ or $n=[nx2\ nx3]$ where nxi is the number of grid points for xi . DEFAULT: $n=[300\ 50]$.

sty is a string defining the style (as in the function `plot`). DEFAULT: $sty='k-'$.

gxy is 1 (2) if the grid in the $x1$ - $x2$ plane is defined for $x1$ ($x2$), consequently $n=[nx1\ nx3]$ ($n=[nx2\ nx3]$). If not provided, `smrplot` attempts to select the most appropriate value for gxy .

1.2.2 smrprint

The following function can be used to print a polynomial, vector polynomial or matrix polynomial:

```
s=smrprint(f,sn,lab)
```

Prints a polynomial, vector polynomial or matrix polynomial.

SMRPRINT INPUTS

f is a polynomial, vector polynomial or matrix polynomial.

sn is either the maximum # of digits or the format string for displaying reals. DEFAULT: $sn=4$.

lab is the label to print for f . DEFAULT: $lab=[]$.

SMRPRINT OUTPUTS

s is a string containing the result.

Part I: Basic Optimization Problems

Chapter 2

Positivity and Non-Positivity

2.1 smrsos

The function `smrsos` provides the *generalized SOS index*, which quantifies how SOS or ratio of two SOS polynomials is a polynomial or a matrix polynomial with unconstrained variables or variables constrained over a semialgebraic set. This allows one to investigate positivity and non-positivity of a polynomial or a matrix polynomial with unconstrained variables or variables constrained over a semialgebraic set. The syntax of `smrsos` is as follows:

```
[ind,M,Dat] = smrsos(f,k,a,b,typ,opts)
```

Computes the generalized SOS index of the polynomial or matrix polynomial $f(x)$ over the semialgebraic set $\Theta = \{x: a(x) \geq 0, b(x) = 0\}$, and checks for non-positivity. $f(x)$ is nonnegative (positive) over Θ if $\text{ind} \geq 0$ ($\text{ind} > 0$). $f(x)$ is not nonnegative (not positive) over Θ if $f(x) < 0$ ($f(x) \leq 0$) at any minimizer lying on Θ .

SMRSOS INPUTS

$f(x)$ (i.e., f) is a polynomial or a symmetric matrix polynomial. With $k=0$ and $\Theta = \mathbb{R}^n$, ind is the SOS index of $f(x)$, which measures how SOS is $f(x)$: $f(x)$ is SOS (and positive) iff $\text{ind} \geq 0$ (and $\text{ind} > 0$). ind is normalized so that it is invariant to positive scaling and $\text{ind}=1$ if $f(x)$ is the sum of squares of all monomials with unit coefficient and degree less than a given integer.

k is a nonnegative integer. With $k>0$ and $\Theta = \mathbb{R}^n$, ind is the generalized SOS index of $f(x)$, which measures how ratio of two SOS polynomials is $f(x)$: there exists a SOS polynomial $q(x)$ of degree not greater than $2k$ such that $q(x)f(x)$ is SOS (and positive) iff $\text{ind} \geq 0$ (and $\text{ind} > 0$). DEFAULT: $k=0$.

$a(x)$ and $b(x)$ (i.e., a and b) are vector polynomials. $\Theta = \mathbb{R}^n$ if $a(x)$

and $b(x)$ are empty. When at least one of them is nonempty, ind is the generalized SOS index of $f(x)$ over the semialgebraic set $\Theta = \{x: a(x) \geq 0, b(x) = 0\}$: $f(x)$ is nonnegative (positive) over Θ if $\text{ind} \geq 0$ ($\text{ind} > 0$). DEFAULT: $a(x) = b(x) = []$.

typ specifies the type of polynomial for $f(x)$: 'g' (generic), 'l' (locally quadratic), 'h' (homogeneous). For locally quadratic and homogeneous polynomials, $\text{ind} > 0$ implies that $f(x)$ is positive definite. DEFAULT: $\text{typ} = \text{'g'}$.

opts specifies the options. In particular:

opts.con : labels of the constants. DEFAULT: $\text{opts.con} = \text{'c'}$;
 opts.conval : values of the constants. DEFAULT: $\text{opts.conval} = []$;
 opts.disp : 0 (1) if display messages is off (on). DEFAULT: $\text{opts.disp} = 1$;
 opts.norm : 0 (1) if data normalization is off (on). DEFAULT: $\text{opts.norm} = 1$;
 opts.pars : SeDuMi parameter pars. DEFAULT: $\text{pars.eps} = 1\text{e-}9$, $\text{pars.fid} = 0$;
 opts.sn : either maximum # of digits or format string for displaying reals. DEFAULT: $\text{opts.sn} = 4$;
 opts.var : label of the variable label. DEFAULT: $\text{opts.var} = \text{'x'}$.

SMRSOS OUTPUTS

ind is the generalized SOS index of $f(x)$ over Θ .

M provides information about the candidate minimizers. In particular:

$M.x$ (each column of): candidate minimizer;
 $M.\text{cost}$ (each column of): $\min(\text{eig}(f(x)))$ for x in $M.x$;
 $M.a$ and $M.b$ (each column of): $a(x)$ and $b(x)$ for x in $M.x$;
 $M.x\text{accu}$ (each entry of): accuracy of x in $M.x$ (0 if best accuracy).

Dat provides data about the problem. In particular:

Dat.err (Dat.errmsg): code (message) of the errors encountered;
 Dat.dec : SOS decomposition of $q(x)f(x)$ over Θ ;
 Dat.lmis.feas : 0 (1) if the LMI problem is infeasible (feasible);
 Dat.lmis.find : LMI feasibility index;
 Dat.lmis.info : SeDuMi info;
 Dat.lmis.nvar : # of LMI scalar variables;
 Dat.n : $[\text{nr } \text{nx}]$ where nr (nx) is the length of $f(x)$ (x);
 Dat.q : $q(x)$.

2.2 Examples: Positivity

Example POS1. We want to establish whether the polynomial

$$f(x) = 4 - 3x + x^4$$

is positive (i.e., $f(x) > 0$ for all x). Hence:

```
>> f='4-3x1+x1^4';
>> [ind,M,Dat]=smrsos(f);
```

```
SOS index (k=0):
ind=0.1808
```

The found SOS index is positive, hence implying that $f(x)$ is positive.

Next, we want to find a SOS decomposition of $f(x)$. Hence:

```
>> smrprint(Dat.dec,[],'g')
```

```
g_1=2-0.75*x1-0.4147*x1^2
g_2=1.047*x1-0.297*x1^2
g_3=0.8601*x1^2
```

We find the polynomials $g_i(x)$ such that $f(x) = \sum_{i=1}^3 g_i(x)^2$.

Example POS2. We want to establish whether the (symmetric) matrix polynomial

$$F(x) = \begin{pmatrix} 2 - x_1 + x_1^2 + 3x_2^2 & -1 + x_2 + 2x_1x_2 \\ \star & 1 + 3x_1^2 + x_1x_2 + x_2^2 \end{pmatrix}.$$

is positive (i.e., $F(x) > 0$ for all x). Hence:

```
>> F='[2-x1+x1^2+3x2^2,-1+x2+2x1x2;1+3x1^2+x1x2+x2^2]';
>> [ind,M,Dat]=smrsos(F);
```

```
SOS index (k=0):
ind=0.01196
```

The found SOS index is positive, hence implying that $F(x)$ is positive.

Next, we want to find a SOS decomposition of $F(x)$. Hence:

```
>> smrprint(Dat.dec,[],'G')
```

```
G_1=[1.414-0.3536*x1,-0.7071-0.08709*x1+0.3198*x2;-0.1794*x1+0.7745*x2,0.7071-0.08709*x1
G_2=[0.9181*x1+0.1513*x2,-0.05055*x1+0.2905*x2;1.146*x2,1.727*x1+0.3303*x2]
G_3=[1.032*x2,-0.6497*x2;0,0.4241*x2]
```

We find the matrix polynomials $G_i(x)$ such that $F(x) = \sum_{i=1}^3 G_i(x)'G_i(x)$.

2.3 Examples: Non-Positivity

Example POS3. We want to establish whether the polynomial

$$f(x) = 3.3 - 2x_1 + 3x_2 - x_1x_2 + x_1^2x_2 - x_1x_2^3 + x_1^3x_2^2 + x_1^6 + x_2^6$$

is positive. Hence:

```
>> f='3.3-2x1+3x2-x1x2+x1^2x2-x1x2^3+x1^3x2^2+x1^6+x2^6';
>> [ind,M,Dat]=smrsos(f);
```

SOS index (k=0):

```
ind=-5.787e-005
```

Points in set M:

```
x=[-1.039,-1.144] -> f(x)=-0.001455
```

The found SOS index is negative, i.e. $f(x)$ is not SOS. Moreover, we have found that $f(x) < 0$ for some x (point in set M), and hence $f(x)$ is non-positive.

Example POS4. We want to establish whether the (symmetric) matrix polynomial

$$F(x) = \begin{pmatrix} 1 + 2x_1^4 + 3x_2^4 & x_1 - x_2 - 3x_1x_2 \\ \star & 3 + x_1^4 + x_2^4 \end{pmatrix}$$

is positive. Hence:

```
>> F='[1+2x1^4+3x2^4,x1-x2-3x1x2;3+x1^4+x2^4]';
>> [ind,M,Dat]=smrsos(F);
```

SOS index (k=0):

```
ind=-0.02225
```

Points in set M:

```
x=[0.7677,-0.7067] -> min(eig(F(x)))=-0.1353
```

The found SOS index is negative, i.e. $F(x)$ is not SOS. Moreover, we have found that $F(x) \not\geq 0$ for some x (point in set M), and hence $F(x)$ is non-positive.

2.4 Examples: Positivity of PNS Polynomials

Example POS5 (Motzkin polynomial). Here we consider the Motzkin polynomial

$$f(x) = 1 - 3x_1^2x_2^2 + x_1^4x_2^2 + x_1^2x_2^4.$$

Motzkin polynomial is a PNS polynomial (i.e., nonnegative but not SOS). Clearly, the SOS index has to be negative. In fact:

```
>> f='1-3x1^2x2^2+x1^4x2^2+x1^2x2^4';
>> [ind,M,Dat]=smrsos(f);
```

```
SOS index (k=0):
    ind=-0.00638
Points in set M:
    No point has been found
```

Hence, we compute the generalized SOS index (case $k = 1$):

```
>> k=1;
>> [ind,M,Dat]=smrsos(f,k);
```

```
SOS index (k=1):
    ind=-5.593e-010
Points in set M:
    No point has been found
```

The found generalized SOS index is nonnegative (+/- solver precision).

Example POS6. Here we consider the PNS polynomial

$$f(x) = 4 - 19x_1^2 + 29x_2^2 + 29x_1^4 - 30x_1^2x_2^2 - 19x_2^4 + 4x_1^6 - 19x_1^4x_2^2 + 29x_1^2x_2^4 + 4x_2^6 - 0.4(1 + x_1^2 + x_2^2)^3.$$

This PNS polynomial is positive, but clearly the SOS index is negative:

```
>> f='4-19x1^2+29x2^2+29x1^4-30x1^2x2^2-19x2^4';
>> f=[f,'+4x1^6-19x1^4x2^2+29x1^2x2^4+4x2^6-0.4(1+x1^2+x2^2)^3'];
>> [ind,M,Dat]=smrsos(f);
```

```
SOS index (k=0):
    ind=-0.03663
Points in set M:
    No point has been found
```

Hence, we compute the generalized SOS index (case $k = 1$):

```
>> k=1;
>> [ind,M,Dat]=smrsos(f,k);
```

```
SOS index (k=1):
    ind=0.001198
```

The found generalized SOS index is positive, hence implying that $f(x)$ is positive.

2.5 Examples: Positivity and Non-Positivity over Semialgebraic Sets

Example POS7. We want to establish whether the polynomial

$$f(x) = 3 + x_1 - x_1x_2 + 3x_1^2x_2 + x_2^3$$

is positive over the semialgebraic set

$$\Theta = \{x : 1 - x_1^2 - x_2^2 \geq 0\}.$$

Hence:

```
>> f='3+x1-x1x2+3x1^2x2+x2^3';
>> a='1-x1^2-x2^2';
>> [ind,M,Dat]=smrsos(f,[],a);
```

SOS index (k=0):

ind=0.06664

The found generalized SOS index is positive, hence implying that $f(x)$ is positive over Θ .

Example POS8. We want to establish whether the (symmetric) matrix polynomial

$$F(x) = \begin{pmatrix} 3 + 3x_1^2 - x_1x_2^2 & -1 + x_1 - 2x_2 \\ \star & 4 + x_1^4 + x_2^4 \end{pmatrix}$$

is positive over the semialgebraic set

$$\Theta = \{x : 4 - x_1^2 \geq 0, 9 - x_2^2 \geq 0, 1 - x_1x_2 + x_1^3 - 2x_2^2 = 0\}.$$

Hence:

```
>> F='[3+3x1^2-x1x2^2,-1+x1-2x2;4+x1^4+x2^4]';
>> a={'4-x1^2','9-x2^2'};
>> b='1-x1x2+x1^3-2x2^2';
>> [ind,M,Dat]=smrsos(F,[],a,b);
```

SOS index (k=0):

ind=0.0003031

The found generalized SOS index is positive, hence implying that $F(x)$ is positive over Θ .

Example POS9. We want to establish whether the (symmetric) matrix polynomial

$$F(x) = \begin{pmatrix} 3 - x_2x_3 + x_3^2 & x_2 - x_1x_3 & 3 + x_2 \\ \star & 5 + x_1x_3 - x_2^2 & 1 + 2x_3 \\ \star & \star & 3 + x_1 + x_2^2 \end{pmatrix}$$

is positive over the semialgebraic set

$$\Theta = \{x : 1 - x_1^2 \geq 0, 1 - x_2^2 \geq 0, 1 - x_3^2 \geq 0\}.$$

Hence:

```
>> F='[3-x2*x3+x3^2,x2-x1*x3,3+x2;5+x1*x3-x2^2,1+2*x3;3+x1+x2^2]';
>> a={'1-x1^2','1-x2^2','1-x3^2'};
>> [ind,M,Dat]=smrsos(F,[],a);

SOS index (k=0):
    ind=-0.1376
Points in set M:
    x=[-0.5173,0.4622,0.2878] -> min(eig(F(x)))=-0.7415, min(a(x))=0.7324
```

The found generalized SOS index is negative. Moreover, we have found that $F(x) \not\geq 0$ for some $x \in \Theta$ (point in set $\mathbb{M}$), and hence $F(x)$ is non-positive over Θ .

2.6 Examples: Positivity of Homogeneous Polynomials

Example POS10. We want to establish whether the homogeneous polynomial

$$f(x) = x_1^4 + x_2^4 + x_3^4 - x_1^2 x_2 x_3 - x_2^2 x_3^2$$

is positive definite (i.e., $f(x) > 0$ for all $x \neq 0$). Hence:

```
>> f='x1^4+x2^4+x3^4-x1^2x2x3-x2^2x3^2';
>> [ind,M,Dat]=smrsos(f,[],[],[],'h');
```

```
SOS index (k=0):
    ind=0.1756
```

The found SOS index is positive, hence implying that $f(x)$ is positive definite.

Chapter 3

Minimum of Rational Functions

3.1 smropt

The function `smropt` computes a lower bound of the minimum of a rational function with unconstrained variables or variables constrained over a semialgebraic set, and verifies the tightness of the found lower bound. The syntax of `smropt` is as follows:

```
[bnd,M,Dat] = smropt(f,k,g,a,b,opts)
```

Computes a lower bound of the minimum of the rational function $f(x)/g(x)$ over the semialgebraic set $\text{Theta}=\{x: a(x)\geq 0, b(x)=0\}$. The lower bound is tight iff it is achieved for any minimizer in the set M lying on Theta .

SMROPT INPUTS

$f(x)$ and $g(x)$ (i.e., f and g) are polynomials. $g(x)$ has to be positive. DEFAULT: $g(x)=1$.

k is the semi-degree of the SOS multiplier for $f(x)$ and $g(x)$. The conservatism of the lower bound decreases as k increases. DEFAULT: $k=0$.

$a(x)$ and $b(x)$ (i.e., a and b) are vector polynomials. $\text{Theta}=\mathbb{R}^n$ if $a(x)$ and $b(x)$ are empty. DEFAULT: $a(x)=b(x)=[]$.

`opts` specifies the options. In particular:

`opts.con`: labels of the constants. DEFAULT: `opts.con='c'`;

`opts.conval`: values of the constants. DEFAULT: `opts.conval=[ ]`;

`opts.bise`: lower limit, initial lower limit, initial upper limit, upper limit, and relative tolerance for bisection (used when $k>0$). DEFAULT: `opts.bise=[-1e6 -20 20 1e6 1e-5]`;

`opts.disp`: 0 (1) if display messages is off (on). DEFAULT: `opts.disp=1`;

`opts.norm`: 0 (1) if data normalization is off (on). DEFAULT: `opts.norm=1`;

`opts.pars`: SeDuMi parameter `pars`. DEFAULT: `pars.eps=1e-9, pars.fid=0`;

opts.sn: either maximum # of digits or format string for displaying reals. DEFAULT: opts.sn=4;
 opts.var: label of the variable label. DEFAULT: opts.var='x'.

SMROPT OUTPUTS

bnd is the lower bound of $f(x)/g(x)$ over Theta.

M provides information about the minimizers. In particular:
 M.x (each column of): candidate minimizer;
 M.cost (each column of): $f(x)/g(x)$ for x in M.x;
 M.a and M.b (each column of): $a(x)$ and $b(x)$ for x in M.x;
 M.xaccu (each entry of): accuracy of x in M.x (0 if best accuracy).

Dat provides data of the problem. In particular:
 Dat.err (Dat.errmsg): code (message) of the errors encountered;
 Dat.lmis.feas: 0 (1) if the LMI problem is infeasible (feasible);
 Dat.lmis.find: LMI feasibility index;
 Dat.lmis.info: SeDuMi info;
 Dat.lmis.nvar: # of LMI scalar variables;
 Dat.n: length of x .

3.2 Examples: Unconstrained Variable

Example OPT1. We want to determine the minimum of the polynomial

$$f(x) = 2x_1 + 3x_2 + 4x_3 + x_1x_2x_3 + x_1^4 + x_2^4 + x_3^4.$$

Hence:

```
>> f='2x1+3x2+4x3+x1x2x3+x1^4+x2^4+x3^4';
>> [bnd,M,Dat]=smropt(f);
```

Lower bound (k=0):

```
bnd=-7.077
```

Points in set M:

```
x=[-0.9155,-0.9984,-1.071] -> f(x)/g(x)=-7.077
```

The found lower bound is achieved for some x (point in set M), and hence the lower bound is tight.

Example OPT2. We want to determine the minimum of the rational function

$$\frac{f(x)}{g(x)} = \frac{-2x_2x_3 - 3x_1^2x_2 + x_1^4 + x_2^4 + x_3^4}{1 + x_1 - x_3 + 2x_1^2 + 2x_2^2 + 2x_3^2}.$$

Hence:

```
>> f='-2x2x3-3x1^2x2+x1^4+x2^4+x3^4';
>> g='1+x1-x3+2x1^2+2x2^2+2x3^2';
>> [bnd,M,Dat]=smropt(f,[],g);
```

Lower bound (k=0):

bnd=-0.549

Points in set M:

x=[-0.8636,0.7573,0.5815] -> f(x)/g(x)=-0.549

The found lower bound is achieved for some x (point in set M), and hence the lower bound is tight.

3.3 Examples: Variable Constrained over Semialgebraic Sets

Example OPT3. We want to determine the minimum of the rational function

$$\frac{f(x)}{g(x)} = \frac{7 + x_1x_2^2 + x_1^4 - x_2^3}{1 + x_2 + x_2^2}$$

over the semialgebraic set

$$\Theta = \{x : 4 - x_1^2 - 2x_2^2 \geq 0, -1 + x_1^2 + x_1 + x_1x_2 + x_2 + x_2^3 = 0\}.$$

Hence:

```
>> f='7+x1x2^2+x1^4-x2^3';
>> g='1+x2+x2^2';
>> a='4-x1^2-2x2^2';
>> b='-1+x1^2+x1+x1x2+x2+x2^3';
>> [bnd,M,Dat]=smropt(f,[],g,a,b);
```

Lower bound (k=0):

bnd=1.913

Points in set M:

x=[-0.8144,0.9891] -> f(x)/g(x)=1.913, min(a(x))=1.38, max(|b(x)|)=3.326e-005

The found lower bound is achieved for some $x \in \Theta$ (point in set M), and hence the lower bound is tight.

Example OPT4. We want to determine the minimum of the rational function

$$\frac{f(x)}{g(x)} = \frac{x_2 + 2x_3 + x_1^4 + x_2^4 + x_3^4}{1 + x_1^2 + x_2^2 + x_3^2}$$

over the semialgebraic set

$$\Theta = \{x : x_1^2 - 1 \geq 0, x_2^2 - 1 \geq 0, x_3^2 - 1 \geq 0, x_1x_2x_3 - 3 = 0\}.$$

Hence:

```
>> f='x2+2x3+x1^4+x2^4+x3^4';
>> g='1+x1^2+x2^2+x3^2';
>> a={'x1^2-1','x2^2-1','x3^2-1'};
>> b='x1x2x3-3';
>> [bnd,M,Dat]=smropt(f,[],g,a,b);
```

Lower bound (k=0):

bnd=1.19

Points in set M:

x=[1.406,-1.443,-1.478] -> f(x)/g(x)=1.19, min(a(x))=0.9781, max(|b(x)|)=2.391e-005

The found lower bound is achieved for some $x \in \Theta$ (point in set M), and hence the lower bound is tight.

Chapter 4

Solving Systems of Polynomial Equations and Inequalities

4.1 smrsps

The function `smrsps` determines the solutions of a systems of polynomial equations and inequalities. The syntax of `smrsps` is as follows:

```
[snd,M,Dat] = smrsps(f,k,a,opts)
```

Solves the system of polynomial equations $f(x)=0$ over the semialgebraic set $\Theta=\{x: a(x)\geq 0\}$. There is no solution if $snd>0$.

SMRSPS INPUTS

$f(x)$ (i.e., f) is a vector polynomial.

k is a nonnegative integer. `smrsps` looks for the zeros of $q(x)f(x)'f(x)$ where $q(x)$ is a positive polynomial of degree $2k$. DEFAULT: $k=0$.

$a(x)$ (i.e., a) is a vector polynomial. $\Theta=\mathbb{R}^n$ if $a(x)$ is empty. DEFAULT: $a(x)=[]$.

`opts` specifies the options. In particular:

`opts.con`: labels of the constants. DEFAULT: `opts.con='c'`;

`opts.conval`: values of the constants. DEFAULT: `opts.conval=[]`;

`opts.disp`: 0 (1) if display messages is off (on). DEFAULT: `opts.disp=1`;

`opts.norm`: 0 (1) if data normalization is off (on). DEFAULT: `opts.norm=1`;

`opts.pars`: SeDuMi parameter `pars`. DEFAULT: `pars.eps=1e-9`, `pars.fid=0`;

`opts.sn`: either maximum # of digits or format string for displaying reals. DEFAULT: `opts.sn=4`;

`opts.var`: label of the variable label. DEFAULT: `opts.var='x'`.

SMRSPS OUTPUTS

snd is a scaled generalized SOS index of $f(x)'f(x)$ over Theta.

M provides information about the minimizers. In particular:

M.x (each column of): candidate solution;

M.cost (each column of): $\max(|f|)$ for x in M.x;

M.a (each column of): $a(x)$ for x in M.x;

M.xaccu (each entry of): accuracy of x in M.x (0 if best accuracy).

Dat provides data about the problem. In particular:

Dat.err (Dat.errmsg): code (message) of the errors encountered;

Dat.lmis.feas: 0 (1) if the LMI problem is infeasible (feasible);

Dat.lmis.find: LMI feasibility index;

Dat.lmis.info: SeDuMi info;

Dat.lmis.nvar: # of LMI scalar variables;

Dat.n: length of x.

4.2 Examples: Systems of Polynomial Equations

Example SPS1. We want to determine the solutions of the system of polynomial equations

$$\begin{cases} 1 - x_1 - x_1^3 + x_1x_2^2 &= 0 \\ 1 + x_2 - x_2^3 + 3x_1^2 &= 0. \end{cases}$$

Hence:

```
>> f={'1-x1-x1^3+x1x2^2','1+x2-x2^3+3x1^2'};
>> [bnd,M,Dat]=smrsp(f);
```

scaled SOS index (k=0):

snd=-1.19e-009

Points in set M:

x=[3.103,3.211] -> $\max(|f(x)|)=0.01158$

x=[-2.751,2.99] -> $\max(|f(x)|)=0.02589$

x=[-0.8646,1.704] -> $\max(|f(x)|)=0.000911$

We have found three solutions (points in set M), whose accuracy is measured by $\max(|f(x)|)$.

Example SPS2. We want to determine the solutions of the system of polynomial equations

$$\begin{cases} 0.1x_1^4 + 0.2x_2^4 - 0.2x_1x_2^2 + 0.2x_2^3 - 2.5x_1^2 - 0.5x_2^2 + 3x_1 - 0.5x_2 + 2 &= 0 \\ 0.2x_1^4 + 0.3x_2^4 + 0.5x_2^3 - 2.5x_2^2 - 3x_2 + 0.5 &= 0 \end{cases}$$

Hence:

```
>> f={'0.1x1^4+0.2x2^4-0.2x1x2^2+0.2x2^3-2.5x1^2-0.5x2^2+3x1-0.5x2+2';...
>>      '0.2x1^4+0.3x2^4+0.5x2^3-2.5x2^2-3x2+0.5'};
>> [bnd,M,Dat]=smrsps(f);
```

scaled SOS index (k=0):

```
snd=-2.34e-010
```

Points in set M:

```
x=[2.287,2.256] -> max(|f(x)|)=0.008934
x=[2.071,-2.586] -> max(|f(x)|)=0.01291
x=[1.757,0.571] -> max(|f(x)|)=0.003159
x=[1.657,-1.683] -> max(|f(x)|)=0.000357
x=[-1.843,-2.984] -> max(|f(x)|)=0.002716
x=[-1.764,2.584] -> max(|f(x)|)=0.002959
x=[-0.4968,-1.284] -> max(|f(x)|)=0.000306
x=[-0.4621,0.1513] -> max(|f(x)|)=2.449e-005
```

We have found eight solutions (points in set M), whose accuracy is measured by $\max(|f(x)|)$.

4.3 Examples: Systems of Polynomial Equations and Inequalities

Example SPS3. We want to determine the solutions of the system of polynomial equations

$$\begin{cases} x_1^4 + x_2^4 - 4x_1x_2 + 1 = 0 \\ x_1 - x_2^2 = 0 \end{cases}$$

over the semialgebraic set

$$\Theta = \{x : 1 - x_1^2 \geq 0, 1 - x_2^2 \geq 0\}.$$

Hence:

```
>> f={'x1^4+x2^4-4x1x2+1';'x1-x2^2'};
>> a={'1-x1^2';'1-x2^2'};
>> [bnd,M,Dat]=smrsps(f,[],a);
```

scaled SOS index (k=0):

```
snd=5.175e-009
```

Points in set M:

```
x=[0.4631,0.6805] -> max(|f(x)|)=4.964e-005, min(a(x))=0.5369
```

We have found one solution (point in set M), whose accuracy is measured by $\max(|f(x)|)$ and whose feasibility is measured by $\min(|a(x)|)$.

Example SPS4. We want to determine the solutions of the system of polynomial equations

$$\begin{cases} x_1^4 + x_2^4 + x_3^4 - 1 &= 0 \\ x_1 x_2 x_3 + x_1^2 + x_2 &= 0 \\ x_1^2 - x_2^2 - x_3^2 + x_2 x_3 - x_1 &= 0 \end{cases}$$

over the semialgebraic set

$$\Theta = \{x : x_3 \geq 0\}.$$

Hence:

```
>> f={'x1^4+x2^4+x3^4-1','x1x2x3+x1^2+x2','x1^2-x2^2-x3^2+x2x3-x1'};
>> a='x3';
>> [bnd,M,Dat]=smrsps(f,[],a);

scaled SOS index (k=0):
    snd=3.922e-009
Points in set M:
    x=[-0.7683,-0.886,0.4344] -> max(|f(x)|)=8.972e-005, min(a(x))=0.4344
```

We have found one solution (point in set M), whose accuracy is measured by $\max(|f(x)|)$ and whose feasibility is measured by $\min(|a(x)|)$.

Part II: Dynamical Systems

Chapter 5

Analysis and Control of the Domain of Attraction

5.1 smrda

`smrda` computes a lower bound of the largest estimate of the DA (LEDA) of an equilibrium point of a polynomial system for a given polynomial Lyapunov function (LF), verifies the tightness of the found lower bound, searches for optimal polynomial LFs maximizing the size of the LEDA, and computes a polynomial output controller for enlarging the LEDA. The function `smrda` also plots the found results in the case of second or third order. The syntax of `smrda` is as follows:

```
[bnd,M,Dat]=smrda(sys,h,v,c,e,opts)
```

Estimates the domain of attraction (DA) of the equilibrium point `e`, and computes a controller to enlarge such an estimate. The system is

```
dot(x)=f(x,u)
y=g(x)
u=k(y)
```

where `x` is the state, `u` is the input, `y` is the output. The vectors `f(x,u)`, `g(x)` and `k(y)` are polynomial in `x` and `y`, and linear in `u`.

If `e` is the origin, the estimate of the DA is the sublevel set

$$V(c)=\{x: v(x)\leq c\}$$

where `v(x)` is a candidate Lyapunov function (LF) for the origin. If `e` is not the origin, the estimate of the DA is the shifted sublevel set

$$V(c)=\{x+e: v(x)\leq c\}$$

where $v(x+e)$ is a candidate LF for e .

If c is provided with $c>0$, `smrda` is in "feasibility mode" and tests if $V(c)$ is included in the DA: $bnd>0$ if yes, while $bnd\leq 0$ if nothing can be said.

If c is empty/not provided, `smrda` is in "LEDA mode" and estimates the LEDA provided by $v(x)$, i.e. $V(\gamma)$ where

$$\gamma = \sup c \text{ s.t. } \text{dot}(v)(x) < 0 \text{ for all } x \text{ in } V(c) \setminus \{0\}.$$

In this case, bnd is a lower bound of γ . The lower bound is tight iff for any point in the set M one has $\text{dot}(v)(x)=0$ and $v(x)=bnd$.

If the controller $k(y)$ is provided, the system is considered closed-loop, otherwise open-loop ($u=0$).

If the degree of the controller $k(y)$ is provided, `smrda` is in "synthesis mode" and looks for the controller $k(y)$ of such a degree maximizing the LEDA.

If `opts.optlf=1` or `opts.optlf=2`, `smrda` is in "LF search mode" and searches for a LF maximizing the LEDA. Specifically, with `opts.optlf=1` `smrda` searches for a LF $v(x)$ minizing the trace of the SMR matrix of $v(x)$ under the constraint $V(1)$ is included in the DA. With `opts.optlf=2`, `smrda` searches for a LF $v(x)$ maximizing the size of the sublevel set $W(d)=\{x: w(x)\leq d\}$ under the constraints $W(d)$ is included in $V(1)$ and $V(1)$ is included in the DA, where $w(x)$ is a chosen positive definite polynomial.

SMRDA INPUTS

`sys` describes the system. In particular:

`sys.f`: $f(x,u)$ (text or cell input type);

`sys.g`: $g(x)$ (text or cell input type). DEFAULT: `sys.g=[]`;

`sys.k`: $k(y)$ (text or cell input type). DEFAULT: `sys.k=[]`;

`sys.kdeg`: degree of $k(y)$. DEFAULT: `sys.kdeg=[]`;

`sys.kbnd`: bounds for the coefficients of $k(y)$. If `sys.kbnd=[b1,b2]`, all the coefficients of $k(y)$ are bounded in $[b1,b2]$. If `sys.kbnd` is a matrix where each row has the form $[b1,b2,z1,z2,...]$, the coefficient of the monomial $y1^{z1}y2^{z2}...$ of $k(y)$ is bounded in $[b1,b2]$. DEFAULT: `sys.kbnd=[0 0]`;

`sys.w`: $w(x)$ (text or struct input type).

if `sys` is a vector polynomial (text or cell input type), it is assumed

that $f(x,u)=\text{sys}$.

h is a nonnegative integer. The conservatism of the lower bound decreases as h increases. DEFAULT: $h=0$.

$v(x)$ (i.e., v) is a polynomial (text or struct input type). $v(x)$ has to be positive definite, with positive definite lowest and highest degree forms.

c is a real. In particular:

$c>0$: smrda checks whether $V(c)$ is included in the DA;

c empty/not provided: smrda computes a lower bound of γ .

DEFAULT: $c=[]$.

e is the equilibrium point of interest. DEFAULT: $e=0$.

opts specifies the options:

opts.con: constant label. DEFAULT: $\text{opts.con}='c'$;

opts.conval: constant value. DEFAULT: $\text{opts.conval}=[]$;

opts.bise: lower limit, initial lower limit, initial upper limit, upper limit, and relative tolerance for bisection. DEFAULT:

$\text{opts.bise}=[-1e6 \ -20 \ 20 \ 1e6 \ 1e-5]$;

opts.disp: 0/1 if display messages is off/on. DEFAULT: $\text{opts.disp}=1$;

opts.optlf: optimization criterion and relative tolerance for LF search.

Admissible values for the criterion are 0 (no search), 1 (search by minimizing the trace of the SMR matrix of $v(x)$), and 2 (search by maximizing the size of $W(d)$). DEFAULT: $\text{opts.optlf}=[0 \ 1e-4]$;

opts.pars: SeDuMi parameter pars. DEFAULT: $\text{pars.eps}=1e-9$, $\text{pars.fid}=0$;

opts.var: variable labels. DEFAULT: $\text{opts.var}='xuy'$;

SMRDA OUTPUTS

("feasibility mode") $\text{bnd}>0$ if feasible, $\text{bnd}\leq 0$ if nothing can be said;

("LEDA mode") bnd is a lower bound of γ .

M provides information about the minimizers. In particular:

$M.x$ (each column of): candidate minimizer;

$M.\text{cost}$ (each column of): $[\text{dot}(v(x));v(x)]$ for x in $M.x$;

$M.x\text{accu}$ (each entry of): accuracy of x in $M.x$ (0 if best accuracy).

Dat provides data of the problem. In particular:

$Dat.A$: state-state matrix of the linearized system;

$Dat.c$: c ;

$Dat.\text{cost}$: cost for LF search;

$Dat.d$: d ;

```
Dat.dotv: temporal derivative of v(x);
Dat.err (Dat.errmsg): code (message) of the errors encountered;
Dat.k: k(y);
Dat.lmis.info: SeDuMi info;
Dat.lmis.nvar: # of LMI scalar variables.
```

5.2 Examples

Example DA1 (Van der Pol system). We want to establish whether the sublevel set $\mathcal{V}(c)$ given by

$$\begin{aligned}\mathcal{V}(c) &= \{x : v(x) \leq c\} \\ v(x) &= x_1^2 + x_1x_2 + x_2^2 \\ c &= 0.5\end{aligned}$$

is included in the DA of the origin of the Van der Pol system

$$\dot{x} = \begin{pmatrix} x_2 \\ -x_1 - x_2(1 - x_1^2) \end{pmatrix}.$$

Hence:

```
>> f={'x2','-x1-x2(1-x1^2)'};
>> v='x1^2+x1x2+x2^2';
>> c=0.5;
>> [bnd,M,Dat]=smrda(f,[],v,c);
```

Sublevel set included in DA (c=0.5, h=0):

Yes (LMI problem is feasible, feasibility index=0.1317)

We find that the LMI problem is feasible, and hence $\mathcal{V}(c)$ is included in the DA.

Next, we want to compute the LEDA provided by the LF $v(x)$, i.e. $\mathcal{V}(\gamma)$ where

$$\gamma = \sup c \text{ s.t. } \dot{v}(x) < 0 \text{ for all } x \in \mathcal{V}(c) \setminus \{0\}.$$

Hence:

```
>> [bnd,M,Dat]=smrda(f,[],v);
>> clf
>> smrda('plot',Dat)
```

LEDA lower bound (h=0):

bnd=0.9152

Points in set M:

```
x=[-0.8586,1.031] -> dot(v)(x)=-5.386e-006, v(x)=0.9152
x=[0.8586,-1.031] -> dot(v)(x)=-5.386e-006, v(x)=0.9152
```

We find the lower bound `bnd` of γ . We have that, for some x (points in set M), $\dot{v}(x) = 0$ and $v(x) = \text{bnd}$, and hence the lower bound is tight.

Next, we want to compute the LEDA provided by the LF

$$v(x) = x_1^2 + x_1x_2 + x_2^2 + x_1^4 + x_2^4$$

Hence:

```
>> v='x1^2+x1x2+x2^2+x1^4+x2^4';
>> [bnd,M,Dat]=smrda(f,[],v);
```

LEDA lower bound (h=0):

bnd=0.4594

Points in set M:

No point has been found

We find the lower bound `bnd` of γ . The set M is empty, which suggests that the lower bound is not tight.

Next, we repeat the computation with $h = 1$ in order to improve the lower bound. Hence:

```
>> h=1;
>> [bnd,M,Dat]=smrda(f,h,v);
>> clf
>> smrda('plot',Dat)
```

LEDA lower bound (h=1):

bnd=1.777

Points in set M:

x=[0.8605,0.3778] -> dot(v)(x)=-3.199e-006, v(x)=1.777

x=[-0.8605,-0.3778] -> dot(v)(x)=-3.199e-006, v(x)=1.777

We find the new lower bound `bnd` of γ . We have that, for some x (points in set M), $\dot{v}(x) = 0$ and $v(x) = \text{bnd}$, and hence the lower bound is tight.

Example DA2 (predator-prey system). Consider the predator-prey system

$$\dot{x} = \begin{pmatrix} -3x_1 + 4x_1^2 - 0.5x_1x_2 - x_1^3 \\ -2.1x_2 + x_1x_2 \end{pmatrix}.$$

and its equilibrium point $e = (2.1, 1.98)'$. We want to establish whether the sublevel set $\mathcal{V}(c)$ given by

$$\begin{aligned} \mathcal{V}(c) &= \{x + e : v(x) \leq c\} \\ v(x) &= x_1^2 + x_1x_2 + x_2^2 \\ c &= 1 \end{aligned}$$

is included in the DA of e . Hence:

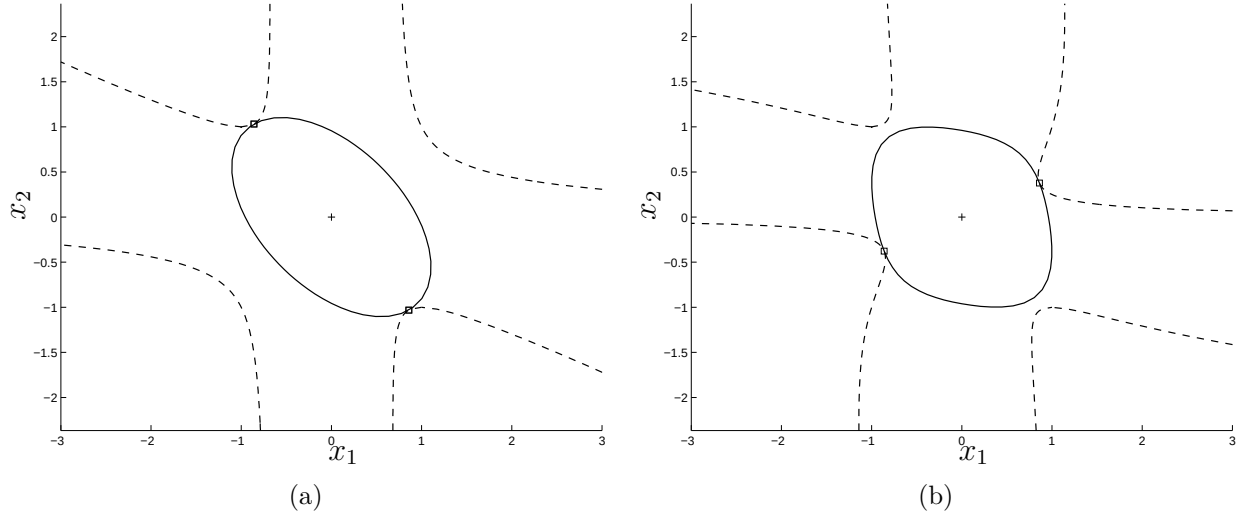


Figure 5.1: Example DA1. (a) LEDA for the quadratic LF. (b) LEDA for the quartic LF.

```
>> f={'-3x1+4x1^2-0.5x1x2-x1^3','-2.1x2+x1x2'};
>> v='4x1^2+x1x2+2x2^2';
>> c=1;
>> e=[2.1;1.98];
>> [bnd,M,Dat]=smrda(f,[],v,c,e);
```

Sublevel set included in DA ($c=1$, $h=0$):

Unknown (LMI problem is infeasible, index=-4.683)

We find that the LMI problem is infeasible, and hence nothing can be said.

Next, we want to compute the LEDA provided by the LF $v(x)$, i.e. $\mathcal{V}(\gamma)$ where

$$\gamma = \sup c \text{ s.t. } \dot{v}(x) < 0 \text{ for all } x \in \mathcal{V}(c) \setminus \{e\}.$$

Hence:

```
>> [bnd,M,Dat]=smrda(f,[],v,[],e);
>> clf
>> pop=[];
>> pop.xran=[-1 3 -1 3];
>> smrda('plot',Dat,pop)
```

LEDA lower bound ($h=0$):

bnd=0.02167

Points in set M:

x=[-0.07475,0.01531] -> dot(v)(x)=-1.592e-008, v(x)=0.02167

We find the lower bound bnd of γ . We have that, for some x (point in set M), $\dot{v}(x) = 0$ and $v(x) = \text{bnd}$, and hence the lower bound is tight.

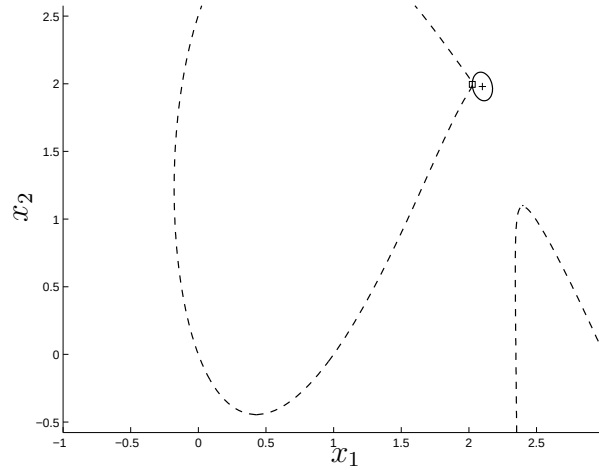


Figure 5.2: Example DA2. LEDA.

Example DA3 (Controller Design: SISO Plant). Consider the system

$$\begin{cases} \dot{x} &= \begin{pmatrix} -x_1 + x_2 - x_1^2 - x_2^3 + x_1 u \\ -2x_2 - x_1^2 + u \end{pmatrix} \\ y &= x_1. \end{cases}$$

We want to design a polynomial static output controller of fixed degree of the form

$$u = k(y) = k_0 + k_1 y + k_2 y^2 + \dots$$

subject to the constraints

$$k_i \in [-1, 1] \quad \forall i = 0, 1, 2, \dots$$

in order to maximize the LEDA of the origin provided by the LF

$$v(x) = x_1^2 + x_2^2$$

i.e. solving

$$\gamma_{con} = \sup_{c, k} c \quad \text{s.t.} \quad \dot{v}(x) < 0 \text{ for all } x \in \mathcal{V}(c) \setminus \{0\} \text{ with } u = k(y).$$

With $k(y)$ of degree 2 we have:

```
>> sys.f={'-x1+x2-x1^2-x2^3+x1u','-2x2-x1^2+u'};
>> sys.g='x1';
>> sys.kbnd=[-1 1];
>> sys.kdeg=0;
>> v='x1^2+x2^2';
```

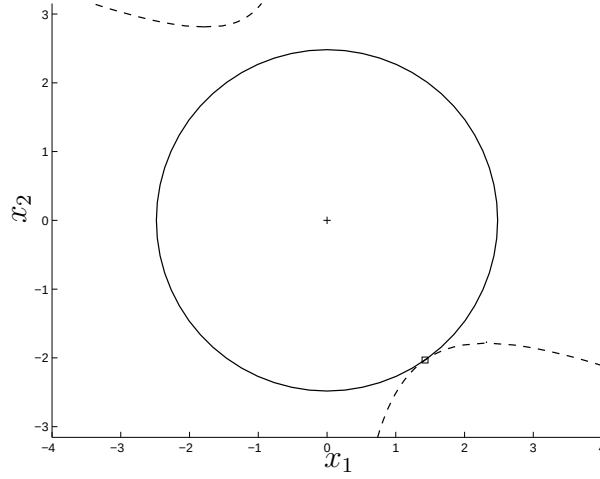


Figure 5.3: Example DA3. LEDA for the found controller.

```
>> [bnd,M,Dat]=smrda(sys,[],v);
>> clf
>> pop=[];
>> pop.xran=[-4 4 -4 4];
>> smrda('plot',Dat,pop)
```

Controller:

$$k(y)=0.9401*y_1-0.1182*y_1^2$$

LEDA lower bound (h=0):

$$bnd=6.158$$

Points in set M:

$$x=[1.425,-2.031] \rightarrow \dot{v}(x)=-4.567e-006, \quad v(x)=6.158$$

We find the lower bound **bnd** of γ_{con} and the corresponding controller $k(y)$. We have that, for some x (point in set M), $\dot{v}(x) = 0$ and $v(x) = \mathbf{bnd}$, and hence the lower bound is tight for the found controller.

Example DA4 (Controller Design: MIMO Plant). Consider the system

$$\begin{cases} \dot{x} &= \begin{pmatrix} x_2 - x_3^2 \\ x_3 - x_1^2 + u_1 \\ -x_1 - 2x_2 - x_3 + x_2^3 + u_2 \end{pmatrix} \\ y_1 &= x_1 - x_2 \\ y_2 &= x_2 - x_3. \end{cases}$$

We want to design a polynomial static output controller of fixed degree of the form

$$u = k(y) = k_0 + k_1 y_1 + k_2 y_2 + k_3 y_1^2 + \dots$$

subject to the constraints

$$k_i \in [-1, 1] \quad \forall i = 0, 1, 2, 3, \dots$$

in order to maximize the LEDA of the origin provided by the LF

$$v(x) = 5x_1^2 + 10x_1x_2 + 2x_1x_3 + 10x_2^2 + 6x_2x_3 + 4x_3^2$$

i.e. solving

$$\gamma_{con} = \sup_{c,k} c \quad \text{s.t.} \quad \dot{v}(x) < 0 \text{ for all } x \in \mathcal{V}(c) \setminus \{0\} \text{ with } u=k(y).$$

With $k(y)$ of degree 2 we have:

```
>> sys.f={'x2-x3^2'; 'x3-x1^2+u1'; '-x1-2x2-x3+x2^3+u2'};
>> sys.g={'x1-x2'; 'x2-x3'};
>> sys.kbnd=[-1 1];
>> sys.kdeg=2;
>> v='5x1^2+10x1x2+2x1x3+10x2^2+6x2x3+4x3^2';
>> [bnd,M,Dat]=smrda(sys,[],v);
>> clf
>> pop=[];
>> pop.xran=[-1 1 -1 1];
>> smrda('plot',Dat,pop)
```

Controller:

$$k(y)_1 = -0.09232y_1 - 0.1495y_2 + 0.2185y_1^2 + 0.3858y_1y_2 + 0.205y_2^2$$

$$k(y)_2 = 0.01455y_1 + 0.3286y_2 + 0.08405y_1^2 + 0.02679y_1y_2 - 0.1478y_2^2$$

LEDA lower bound (h=0):

$$\text{bnd} = 0.3602$$

Points in set M:

$$x = [-0.04786, 0.183, 0.07722] \rightarrow \text{dot}(v)(x) = -1.114e-007, \quad v(x) = 0.3602$$

$$x = [0.1609, -0.2319, -0.05185] \rightarrow \text{dot}(v)(x) = -2.256e-007, \quad v(x) = 0.3602$$

$$x = [-0.1369, -0.07266, -0.1021] \rightarrow \text{dot}(v)(x) = -9.093e-008, \quad v(x) = 0.3602$$

We find the lower bound **bnd** of γ_{con} and the corresponding controller $k(y)$. We have that, for some x (points in set M), $\dot{v}(x) = 0$ and $v(x) = \text{bnd}$, and hence the lower bound is tight for the found controller.

Example DA5 (LF search for maximizing the LEDA in Van der Pol system). We want to search for a LF of fixed degree maximizing the size of the LEDA of the origin of the Van der Pol system

$$\dot{x} = \begin{pmatrix} x_2 \\ -x_1 - x_2(1 - x_1^2) \end{pmatrix}.$$

We express the LEDA as

$$\mathcal{V}(1) = \{x : v(x) \leq 1\}$$

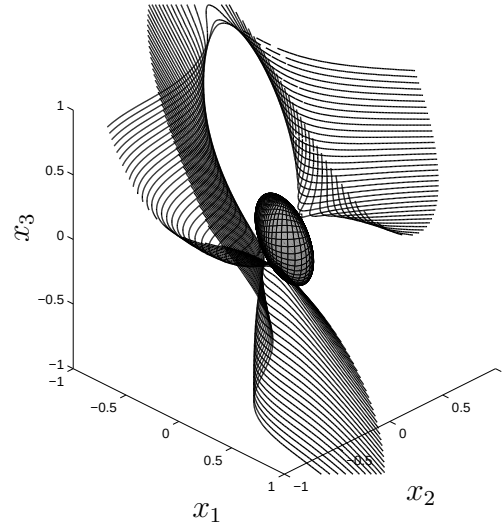


Figure 5.4: Example DA4. LEDA for the found controller.

where $v(x)$ is the LF to determine.

First, we proceed by minimizing the trace of the SMR matrix of $v(x)$. We choose $\deg(v) = 2$ and the initialization $v(x) = x_1^2 + x_1x_2 + x_2^2$:

```
>> f={'x2','-x1-x2(1-x1^2)'};
>> v='x1^2+x1x2+x2^2';
>> opts.optlbf=1;
>> [bnd,M,Dat]=smrda(f,[],v,[],[],opts);
>> clf
>> smrda('plot',Dat)
```

Optimal cost, LF and LEDA lower bound ($\deg(v)=2$, $h=0$):

```
cost=1.035
v(x)=0.5969*x1^2+0.364*x1*x2+0.4383*x2^2
bnd=1
```

We find the final cost `cost` (i.e., trace of the SMR matrix of $v(x)$) and the final LF `v(x)`.

Second, we proceed by maximizing the size of the sublevel set

$$\mathcal{W}(d) = \{x : w(x) \leq d\}$$

included in the LEDA. We choose $\deg(v) = 2$, the initialization $v(x) = x_1^2 + x_1x_2 + x_2^2$, and $w(x) = x_1^2 + x_2^2$:

```
>> w='x1^2+x2^2';
>> opts.optlbf=2;
```

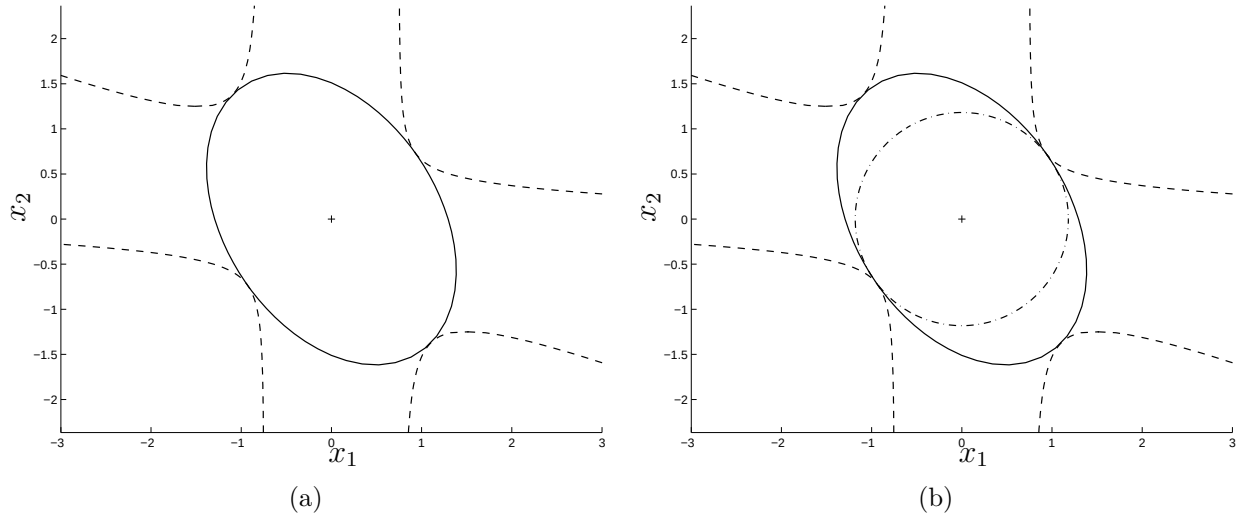


Figure 5.5: Example DA5. (a) LEDA for the LF found by minimizing the trace of the SMR matrix of $v(x)$. (b) LEDA for the LF found by maximizing the size of the sublevel set $W(d)$ included in the LEDA.

```
>> [bnd,M,Dat]=smrda(f,[],v,[],[],opts);
>> clf
>> smrda('plot',Dat)
```

Optimal cost, LF and LEDA lower bound ($\deg(v)=2$, $h=0$):

```
cost=1.396
v(x)=0.5969*x1^2+0.364*x1*x2+0.4383*x2^2
bnd=1
```

We find the final cost `cost` (i.e., d) and the final LF $v(x)$. Observe that, in this case, both methods return the same LF.