

---

Polynomial Optimization – Computer Project 1

---

The aim of this project is to solve the polynomial optimization problem

$$(P) \quad \begin{aligned} & \text{minimize } 2x_1^4 + 10x_1^3 + x_1^2x_2 + 19x_1^2 - x_1x_2^2 + 4x_1x_2 + \\ & \quad 14x_1 + 2x_2^4 - 10x_2^3 + 19x_2^2 - 14x_2 + 11 \\ & \text{over } x_1, x_2 \in \mathbb{R} \\ & \text{subject to } x_1^2 + 3x_1 - x_2^2 + x_2 + 3 \geq 0 \\ & \quad x_1^2 + 2x_1 - x_2^2 + 2x_2 + 1 \geq 0 \\ & \quad x_1^3 + 3x_1^2 + 2x_1 - x_2^3 + 3x_2^2 - 2x_2 \geq 0 \end{aligned}$$

using semidefinite programming. You will need a working version of the commercial system MATLAB<sup>1</sup> including the Symbolic Math Toolbox MuPAD<sup>2</sup> (e.g., a license from the state-wide MATLAB agreement<sup>3</sup> or the version installed in the public PhyMa computer lab<sup>4</sup>). You need to have installed<sup>5</sup> the following MATLAB-packages (cf. Problem Set 1): the modeling language YALMIP<sup>6</sup> and the SDP solver MOSEK<sup>7</sup> or any other SDP solver supported by YALMIP<sup>8</sup> (CSDP, DSDP, LOGDETPPA, PENLAB, SDPA, SDPLR, SDPT3, SDPNAL, SEDUMI, PENSDP).

In this project, you have to construct four files where narendra<sup>9</sup> must be replaced by your given name in lowercase letters:

- (1) a MuPAD notebook pop1narendra.mn
- (2) a MuPAD program pop1narendra.mu
- (3) a MATLAB script pop1narendra\_constraints.m
- (4) a MATLAB script pop1narendra.m

---

<sup>1</sup><http://en.wikipedia.org/wiki/MATLAB>

<sup>2</sup><http://en.wikipedia.org/wiki/MuPAD>

<sup>3</sup><http://www.rz.uni-konstanz.de/en/our-offers/software/state-wide-matlab-agreement/>

<sup>4</sup>[http://www.math.uni-konstanz.de/fb\\_seiten/contrib/studium/phyma/phyma\\_en.php?menu\\_var=Studies](http://www.math.uni-konstanz.de/fb_seiten/contrib/studium/phyma/phyma_en.php?menu_var=Studies)

<sup>5</sup>and added to the MATLAB path

<sup>6</sup><http://users.isy.liu.se/johanl/yalmip/>

<sup>7</sup><https://www.mosek.com/resources/academic-license>

<sup>8</sup><http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Solvers.Solvers>

<sup>9</sup>[http://en.wikipedia.org/wiki/Karmarkar%27s\\_algorithm](http://en.wikipedia.org/wiki/Karmarkar%27s_algorithm)

You will first create (1) following our instructions. This will be your potentially first MuPAD notebook where you can take your first steps with the very nice and elegant MuPAD language. In the end (1) should be a readable MuPAD notebook that creates the constraints of a semidefinite program ( $P_1$ ) relaxing ( $P$ ) and writes them in YALMIP format to the file (3). In other words, file (3) will be created automatically once your MuPAD notebook works correctly and is evaluated. MuPAD notebooks can only be opened but cannot be called from MATLAB scripts. What we finally need is therefore a MuPAD program. So you will create file (2) by just copying all essential commands from (1) into (2). The “master” file (4) will therefore first execute (2) which produces a semidefinite relaxation ( $P_1$ ) of ( $P$ ) and writes it into (3), then (4) will execute (3) to define the constraints in the YALMIP format, and finally (3) will tell YALMIP to solve ( $P_1$ ) using an SDP-solver.

- (a) Start MATLAB and add YALMIP and MOSEK or another SDP solver to the MATLAB path.
- (b) Set the working directory (“Current Folder”) to the folder where you want to store your project in (using `cd`).<sup>10</sup>
- (c) Create a blank MuPAD notebook by typing `mupad` in the MATLAB command window, put a comment with your name in the first line of this notebook. Save the notebook in your working directory under the name of `pop1narendra.mn`.
- (d) Open MATLAB’s or MuPAD’s help window and take a tour through the topics
  - Symbolic Math Toolbox → MuPAD → Getting Started with MuPAD
  - Symbolic Math Toolbox → MuPAD → MuPAD Language Fundamentals
  - Symbolic Math Toolbox → MuPAD → Mathematics → Linear Algebra
 while experimenting in the notebook. Also search on the internet for learning about lists, sequences and functions (in particular the arrow operator) in MuPAD.
- (e) Now search in the help browser for the commands `_assign`, `_equal`, `_seqgen`, `_plus`, `map`, `expand`, `monomials`, `lterm`, `op`, `nops`, `matrix`, `transpose`, `max`, `degree`, `contains`, `subs` and whatever other command you are interested in.
- (f) Define the goal function  $f$  of ( $P$ ) and the three polynomials `p[1]`, `p[2]` and `p[3]` defining the constraints of ( $P$ ).
- (g) Define a function `mon` (using the arrow operator `->`) that yields a sequence of all monomials of degree exactly  $k$  when called with a nonnegative integer  $k$ .
- (h) Define a function `vec` yielding a column vector of all monomials of degree at most  $k$  when called with a nonnegative integer  $k$ .

---

<sup>10</sup>Attention: You should do this before you call MuPAD for the first time during your MATLAB session. Otherwise MuPAD might not notice that you have changed the current directory.

- (i) Define symmetric matrix polynomials  $P_0 \in \text{SIR}[X_1, X_2]^{6 \times 6}$ ,  $P_1 \in \text{SIR}[X_1, X_2]^{3 \times 3}$ ,  $P_2 \in \text{SIR}[X_1, X_2]^{3 \times 3}$  and  $P_3 \in \text{SIR}[X_1, X_2]^{1 \times 1}$  whose positive semidefiniteness expresses the validity of the (mostly redundant) constraints

$$\begin{aligned} (a_1 + a_2x_1 + a_3x_2 + a_4x_1^2 + a_5x_1x_2 + a_6x_2^2)^2 &\geq 0 \\ (a_1 + a_2x_1 + a_3x_2)^2(x_1^2 + 3x_1 - x_2^2 + x_2 + 3) &\geq 0 \\ (a_1 + a_2x_1 + a_3x_2)^2(x_1^2 + 2x_1 - x_2^2 + 2x_2 + 1) &\geq 0 \\ a_1^2(x_1^3 + 3x_1^2 + 2x_1 - x_2^3 + 3x_2^2 - 2x_2) &\geq 0, \end{aligned}$$

( $a_1, \dots, a_6 \in \mathbb{R}$ ) confer the lecture and Exercise Sheet 1.

- (j) Define the list of monomials from  $\mathbb{R}[X_1, X_2]$  appearing somewhere now and another list of the same length  $s$  of variables  $y_i = y[i]$  ( $i \in \{1, \dots, s\}$ ).
- (k) Define a function that *linearizes* polynomials (cf. Exercise Sheet 1 and the lecture) from  $\mathbb{R}[X_1, X_2]$  in the sense that it replaces the  $i$ -th monomial in our list by the corresponding  $y_i$  (except for the constant monomial 1). Use the map function to do this.
- (l) Again using the map function, extend this linearization to matrix polynomials.
- (m) Store the linearized goal function  $f$  in 1 and the linearized matrix polynomials  $P_0, \dots, P_3$  in M0, ..., M3.
- (n) Add and understand the following lines to your notebook:

```
use(generate,MATLAB) // load MATLAB function form library generate
MATLAB(1)
MATLAB(M0)
yalmipspec:="y=sdpvar(".nops(monolist).",1); ":
yalmipspec:=yalmipspec.MATLAB(1).MATLAB(M0).MATLAB(M1).MATLAB(M2).MATLAB(M3):
yalmipspec:=stringlib::subs(yalmipspec,"t0"="1"):
yalmipspec:=stringlib::subs(yalmipspec,"zeros"="sdpvar"):
fprintf(Unquoted,Text,"pop1narendra_constraints.m", yalmipspec)
```

- (o) Create (2) with your favorite editor. Copy and paste everything from (1) which is really needed to fulfill the task of producing (3). Do not copy the things which are nice in a notebook to understand what is going on (like examples). This helps us to see if you have understood what is really necessary. Be careful that you have to slightly adapt your syntax to a program file, e.g. you always have to separate two commands by a semicolon.
- (p) Create (4) with your favorite editor and write the following into this file while trying to understand it using the MATLAB help and the YALMIP wiki.

```

% Narendra Karmarkar
read(symengine, 'pop1narendra.mu')
pop1narendra_constraints
constraints=[M0>=0,M1>=0,M2>=0,M3>=0]
optimize(constraints,1)
value(1)
value(y(2))
value(y(3))
evalin(symengine, 'subs(f,x[1]=-1,x[2]=1)')

```

- (q) Type pop1narendra in the MATLAB command window! Why can you conclude that the SDP-relaxation ( $P_1$ ) has solved (at least up to numerical errors) the POP ( $P$ )?

**Due** by Tuesday, May 6th, 2015, 11:00 am. The four files (1), (2), (3) and (4) must be sent attached to an electronic mail to both Sebastian Gruler<sup>11</sup> and María López Quijorna<sup>12</sup>. Files (1), (2) and (4) must be executable without producing errors. Note that this must work in any directory so please avoid using pathnames when specifying filenames. File (3) must be identical with both the file that is produced by executing (1) and (2), respectively. It is perfectly allowed to collaborate with other students. However, the finalization, annotation and submission of the project has to be done by each participant individually. The only file which must be well documented is (1). The other files need not necessarily contain comments. Comments should be short but pregnant and in English language. In files (1), (2) and (4) you must specify your name by a comment which is of the form // Narandra Karmarkar and % Narandra Karmarkar, respectively.

Where it is little effort, you should keep the code sufficiently general so that it can be easily adapted for future projects, e.g., you should use  $n$  instead of 2 by including a line like

```
n:=2 // number of variables
```

in (1) to allow for later increase of the number of variables. Also you are encouraged to put redundant “pedagogical” code which motivates or explains subsequent more sophisticated code even if it is not necessary for carrying out the project. For example, you could add the line

```
x[i]$i=1..n // the variables
```

which is just there to increase the readability.

Your project submissions will usually not be discussed during the exercise groups but individually in the offices of the tutors. You have to be able to explain your code.

<sup>11</sup><http://www.math.uni-konstanz.de/~gruler/>

<sup>12</sup><http://www.math.uni-konstanz.de/~lopez/>