University of Konstanz
Department of Mathematics and Statistics
Summer Term 2015

Sebastian Gruler
María López Quijorna
Markus Schweighofer

Polynomial Optimization – Computer Project 2

This project will later be continued in Computer Project 3. The aim of the project is to get a good intuition on the accuracy attained by the basic moment relaxation of certain polynomial optimization problems of geometric flavor. This will be done by visualizing the obtained solutions of the relaxation. You will also learn how to communicate between MuPAD and MATLAB without using an auxiliary file. Moreover, we will treat an equality constraint as such instead of writing it as two inequalities.

You will again need MATLAB, its Symbolic Math Toolbox MuPAD and the modeling language YALMIP. As last time, you need to have installed and added to the MATLAB path MOSEK or another SDP solver.

You have to construct four files inside a new directory named `pop2narendra` where `narendra`[1] must be replaced by your given name in lowercase letters:

(1) a MuPAD notebook `robot.mn`

(2) a MuPAD program `robot.mu`

(3) a MATLAB function file `robot.m`

(4) a MATLAB script `robotdemo.m`

All four files have to contain a comment with your name in the first line. Apart from this, the only files which have to be commented are (1) and (4). Indeed, (1) should contain a well-documented version of (2) whose only aim is to explain (2). The MuPAD file (2) will contain (probably amongst others) a procedure `robot:=proc(R)` producing a very basic moment relaxation of polynomial equalities and inequalities associated to the "robot" R (see below). The MATLAB function file (3) contains a MATLAB function `function [constraints,x,y] = robot(R)` calling (2). The MATLAB script (4) contains examples of how to apply the robot procedure with interesting robots that you choose according to our instructions below and also according to your own taste and creativity. You should comment on the outcomes and observations in this script file. Your tutor should have fun while executing (4)!

All files must be executable without producing errors. Note that this must work wherever your directory `pop2narendra` is placed so please avoid using pathnames when specifying filenames. 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. Comments should be concise and in English language. It is preferable to do the MuPAD programming basically in the notebook (1) and to generate (2) from (1) by copy and paste.

---

[1] `http://en.wikipedia.org/wiki/Karmarkar%27s_algorithm`

We define a *robot* to be a strictly (i.e., zero-diagonal) upper triangular matrix $R \in \{0,1,2\}^{n\times n}$ for some $n \geq 3$. One thinks of such a matrix as a "robot" in two-dimensional space with n *joints*, the first two of which are called *base joints*. The base joints are anchored at positions $(0,0)$ and $(1,0)$ in $\mathbb{R}^2$. The position of the other joints can only be constrained by the *links*. A nonzero entry in R means that the joints corresponding to the line and column of the entry are connected by a *rigid* or *flexible* link. Rigid links have unit length and are encoded by 1. Flexible links can vary their length from 1 to $\sqrt{3}$ and are encoded by 2.

To each robot $R \in \{0,1,2\}^{n\times n}$, we associate a finite system of quadratic polynomial equalities and inequalities in the variables x_{ij} ($i \in \{3,\ldots,n\}$, $j \in \{1,2\}$). These variables should be implemented as variables `x(i,j)` of the YALMIP type `sdpvar` in MATLAB and as identifiers `x[i,j]` in MuPAD. We interprete $x_i = (x_{i1}, x_{i2}) \in \mathbb{R}^2$ for $i \in \{1,\ldots,n\}$ as the position of the i-th joint of the robot R. In particular, $x_1 = (0,0)$ and $x_2 = (1,0)$ should always be fixed. However, this information and the corresponding four real variables x_{ij} ($i,j \in \{1,2\}$) do however *not* appear in the system we associate to the robot. Instead, this information should be expressed separately by including at some point `x[1,1]:=0;x[1,2]:=0;x[2,1]:=1;x[2,2]:=0` in your MuPAD code and `x(1,1)=0;x(1,2)=0;x(2,1)=1;x(2,2)=0` in the MATLAB code produced within MuPAD. For example, the system associated to the robot $R_0 := \left(\begin{smallmatrix} 0 & 1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{smallmatrix}\right)$ is

$$x_{3,1}^2 + x_{3,2}^2 = 1,\ 1 \leq (x_{3,1}-1)^2 + x_{3,2}^2 \leq 3$$

where the first equality expresses that the third joint is linked rigidly to the first (anchor) joint and the other two inequalities express that the third joint is linked flexibly to the second (anchor) joint.

(a) In the MuPAD notebook (1), write a MuPAD function `robot:=proc(R)` that takes a robot $R \in \{0,1,2\}^{n\times n}$ as input,

- augments the system of polynomial equalities and inequalities by the family of quadratic redundant inequalities

  $$(a + bx_{3,1} + cx_{3,2} + \ldots)^2 \geq 0 \qquad (a,b,c,\ldots \in \mathbb{R} \text{ parameters}).$$

  Of course, this should be expressed without parameters as a single quadratic polynomial matrix inequality.
- relaxes this system to a system consisting of one linear matrix inequality and possibly other linear inequalities and equalities.
- writes the (data defining) these linear constraints in YALMIP format into a single string `yal` and returns this string. Before returning `yal`, the carriage returns appearing in it should be deleted, e.g. by applying:

  `yal:=yal.stringlib::subs(yal,"\n"="")`

  Use `generate::MATLAB` to convert the linear constraints into a MATLAB readable string. Other MuPAD commands that could potentially be useful are `subs`, `subsex`, `monomials` and `DegreeOrder`. Linear monomials in the variables x_{ij}

($i \in \{3,\dots,n\}$, $j \in \{1,2\}$) should *not* be affected by the linearization. Other monomials will become linear monomials in the new variables $y_1, y_2, \dots$ ($y[i]$ in MuPAD, $y(i)$ in MATLAB). In the example from above, `robot(`R_0`)` should return a string like

```
x=sdpvar(3,2); x(1,1) = 0; x(1,2) = 0; x(2,1) = 1; x(2,2) = 0;
y=sdpvar(3,1);  A0 = sdpvar(3,3);  A0(1,1) = 1.0;  A0(1,2) = x(3,1);
A0(1,3) = x(3,2);  A0(2,1) = x(3,1);  A0(2,2) = y(1);  A0(2,3) = y(2);
A0(3,1) = x(3,2);  A0(3,2) = y(2);  A0(3,3) = y(3);
constraints = [A0>=0];  t0 = 1.0; constraints = [constraints,t0==1];
t0 = y(1)+y(3); constraints = [constraints,t0==1];
t0 = y(1)+y(3)-x(3,1)*2.0+1.0; constraints = [constraints,1<=t0<=3];
```

(b) Comment the file (1) very well, illustrate and test parts of the code with small examples and include this in the MuPAD notebook (1). Once you are convinced that (1) works properly, copy its main content (without the comments) into a MuPAD program (2).

(c) Implement in (3) a MATLAB function `function [constraints,x,y] = robot(R)` that upon input of a robot $R \in \{0,1,2\}^{n\times n}$

- reads (2) by calling `read(symengine,'robot.mu')`,
- calls (2) and executes the command given by the returned string by calling `eval(char(feval(symengine,'robot',R)))`,
- returns the list of linearized constraints in YALMIP format and the families of variables $(x_{ij})_{1\le i\le n, 1\le j\le 2}$ and $(y_i)_i$ of YALMIP type `sdpvar`.

Note that (3) would fit almost in one line.

(d) Write a MATLAB script (4) which should define several example robots and illustrate their behavior by solving SDPs whose constraints are produced by calling (3). For finding a useful objective function, fix a joint of the robot which should serve as *end effector*. One of the robots you consider could be defined by

```
R = zeros(7,7); R(1,2) = 1; R(1,3) = 1; R(2,4) = 1; R(3,4) = 1;
R(2,3) = 2; R(1,4) = 2; R(3,6) = 2; R(3,5) = 1; R(4,6) = 1;
R(5,6) = 1; R(5,7) = 2; R(6,7) = 2;
```

For this robot, it makes sense to consider joint 7 as end effector and one could therefore take a random objective function `randn*x(7,1)+randn*x(7,2)` for solving the SDP. Having solved the SDP one could draw the robot using a command like `gplot(R,value(x),'-*');axis([-4 4 -4 4])`. Make (4) into a pleasant demonstration of your discoveries concerning moment relaxations and robots! In this demo, you should comment on what you think about the quality of the relaxations

Due by Friday, June 5th, 2015, 11:11 am. The four files (1)—(4) must be sent attached to an electronic mail to both Sebastian Gruler[2] and María López Quijorna[3].

[2] `http://www.math.uni-konstanz.de/~gruler/`
[3] `http://www.math.uni-konstanz.de/~lopez/`