POP – Tutorial problems

Instructor: Igor Klep

Problem 1 (SOS Decompositions). *Express the following two polynomials as sums of squares (SOS):*

(a) $2x^4 + 5y^4 - x^2y^2 + 2x^3y + 2x + 2$;

(b) $x^2 - xy^2 + y^4 + 1$.

Problem 2 (Nonnegative but not SOS — *Paper Exercise*). *Consider the polynomial*

$$M(x,y) = x^4y^2 + x^2y^4 - 3x^2y^2 + 1.$$

(a) *Show that* $M \geq 0$ *on* $\mathbb{R}^2$.

(b) *Determine whether* M *is SOS.*

Problem 3 (Sums of Two Squares — *Paper Exercise*). *Show that for univariate polynomials,*

$$p \geq 0 \text{ on } \mathbb{R} \iff p \text{ is a sum of two squares.}$$

Hint: *Use the identity*

$$(P^2 + Q^2)(R^2 + S^2) = (PR + QS)^2 + (PS - QR)^2.$$

Problem 4 (POP: A Polynomial Optimization Problem). *Solve the following polynomial optimization problem (POP):*

$$\min_{x_1, x_2 \in \mathbb{R}} \quad 2x_1^4 + 10x_1^3 + x_1^2x_2 + 19x_1^2 - x_1x_2^2 + 4x_1x_2 + 14x_1 + 2x_2^4 - 10x_2^3 + 19x_2^2 - 14x_2 + 11$$

subject to

$$\begin{aligned} x_1^2 + 3x_1 - x_2^2 + x_2 + 3 &\geq 0, \\ x_1^2 + 2x_1 - x_2^2 + 2x_2 + 1 &\geq 0, \\ x_1^3 + 3x_1^2 + 2x_1 - x_2^3 + 3x_2^2 - 2x_2 &\geq 0. \end{aligned}$$

Problem 5 (Extracting Global Minimizers from Moment Matrices — Coding). *Write code to extract a global minimizer from the moment matrix when the flat extension condition is met. Use a simple POP as an example.*

```
using SumOfSquares
using DynamicPolynomials #Enables symbolic  variables
using MosekTools           #Mosek SDP solver

# Create an SOS optimization model
model = SOSModel(Mosek.Optimizer)

# Define polynomial variables x and y
@polyvar x1 x2

# Define a decision variable t
@variable(model, t)

# Define the constraint set
S = @set x1^2 + x2^2 <= 1

# Add the SOS relaxation constraint:
@constraint(model, x1^4 + x2^4 - 4*x1*x2 + x1^2 + x2^2 >= t,
            domain = S, maxdegree = 6)  # maxdegree controls relaxation order

# Set the objective to maximize t (tightest lower bound)
@objective(model, Max, t)

# Solve the SDP relaxation and Print the optimal solution
optimize!(model)
println("Solution: $(value(t))")

# Extract the moment matrix and check for flat extension
# (Code to compute and analyze the moment matrix goes here)
```

Problem 6 (Max-Cut via Lasserre Relaxation — Coding). *This problem considers Max-Cut from the viewpoint of POP.*

(a) *Given a graph G with uniform edge weights $w_{ij} = 1$ for each edge (i, j), produce code to generate the k-th Lasserre relaxation of the Max-Cut problem for G.*

(b) *Find a graph G where the second Lasserre relaxation performs better than the first.*

(c) *For graphs on up to 10 vertices, analyze the worst performance of the first (second, third) Lasserre relaxation.*

(d) *Generate a random graph on n vertices (with edge probability p). For what sizes of n can you reliably solve the first (second) Lasserre hierarchy for Max-Cut?*

Problem 7 (LP-Based SOS Testing — Coding). *Redo Problem (7) but, instead of using SDP for SOS testing, use the LP method by Ahmadi–Majumdar. Provide code and discuss the differences in performance and certification.*

Problem 8 (PSD vs. sdd Matrices). *Find an example of a matrix that is positive semidefinite (PsD) but not scaled diagonally dominant (sdd). Prove your claim.*

Problem 9 (sdd vs. dd Matrices). *Find an example of a matrix that is sdd but not diagonally dominant (dd). Explain your reasoning.*

Problem 10 (Sparse SOS versus SOS Sparse). *Verify, with examples, that sparse SOS and SOS sparse can yield different results. Use two examples from the lecture notes and explain why the equivalence fails.*

Problem 11 (Chordal Extension and Sparse SDP Relaxation — Coding). *Code a function that, given a sparse POP (with a specified correlative sparsity graph), automatically computes a chordal extension and sets up the corresponding sparse SDP relaxation.*

Hint: *Use available graph libraries and SDP solvers in your favorite language.*

```
# Example pseudocode in Julia
using LightGraphs, JuMP, MosekTools

function chordal_extension_SDP(G::Graph, pop_data)
    # Compute chordal extension (using a heuristic)
    # Set up sparse SDP relaxation for the given POP with sparsity pattern G
    # Return the SDP model
end

# Example usage:
G = erdos_renyi(10, 0.3)
sdp_model = chordal_extension_SDP(G, pop_data)
optimize!(sdp_model)
```

Note: Some of these problems (especially the coding ones) may require the use of external packages such as `SumOfSquares.jl`, `MosekTools.jl`, or graph libraries (e.g., `LightGraphs.jl` in Julia) and a working SDP solver.