JuMP-dev workshop 2018

A Julia JuMP-based module for polynomial optimization with complex variables applied to ACOPF

Gilles Bareilles, Manuel Ruiz, Julie Sliwak

Outline

1. MathProgComplex.jl: A toolbox for Polynomial Optimization Problems with Complex variables $(\boldsymbol{POP} - \mathbb{C})$
2. The Lasserre hierarchy for $(\boldsymbol{POP} - \mathbb{C})$
3. Application to Optimal Power Flow in Alternating Current (ACOPF)
4. Conclusion and future work

https://github.com/JulieSliwak/MathProgComplex.jl (MIT license)

1

A tool for Polynomial Optimization Problems with Complex variables $(POP - \mathbb{C})$

Polynomial Optimization Problems with Complex Variables $(POP - \mathbb{C})$

$$\min \quad \sum_{\alpha,\beta} p^0_{\alpha,\beta}\, \bar{z}^{\alpha} z^{\beta}$$

$$s.t. \quad \sum_{\alpha,\beta} p^i_{\alpha,\beta}\, \bar{z}^{\alpha} z^{\beta} \geq 0 \quad \forall i = 1..p$$

$$z \in \mathbb{C}^n$$

- Optimize a generic complex multivariate **polynomial** function, subject to some complex polynomial equality and inequality constraints.
- A complex multivariate polynomial is a polynomial whose variables and coefficients are **complex numbers**.

A modeler for Polynomial Optimization Problems with Complex variables $(POP-\mathbb{C})$

- Our modeler provides a structure and methods for working with $(POP-\mathbb{C})$.
- The algebraic operations (**+**, **-**, *****, **/**, **conj**, **|.|**) are implemented.
- The base type is **Variable**, from which **Exponents** and **Polynomial** can be constructed by calling the respective constructors or with algebraic operations.
- The **Point** type holds the variables at which polynomials can be evaluated.

Basic structures

Structure	Definition	Notation	Examples
Variable	A pair (String, Type) where Type can be Complex, Real or Bool	z	$\mathrm{x} = Variable(\text{"x"}, Complex)$ $y = Variable(\text{"y"}, Complex)$ $\mathrm{w} = Variable(\text{"w"}, Real)$ $u = Variable(\text{"binary"}, Bool)$
Exponent	A product of Variables	$\prod_i \overline{z_i}^{\alpha_i} z_i^{\beta_i}$	$expo1 = x^2 conj(y)^3,$ $expo2 = xy$
Polynomial	A sum of Exponents times complex coefficient	$\sum c_k \prod_{k_i} \overline{z_{k_i}}^{\alpha_{k_i}} z_{k_i}^{\beta_{k_i}}$	$p(x, y) = (1 + 4im)expo1 + 3expo2$
Point	A dictionary (variable => value) to evaluate a polynomial	$\begin{pmatrix} z_1 \\ \vdots \\ z_k \end{pmatrix} = \begin{pmatrix} value_1 \\ \vdots \\ value_k \end{pmatrix}$	$pt = Dict(x \Rightarrow 1 + 2im, y \Rightarrow 3im)$

$$evaluate(p, pt) = -145 + 28im$$

Polynomial Optimization Problems

Structure	Definition	Examples
Constraint	A **Polynomial** with complex bounds	$3x + y + 2 \leq 3 + 5im$
Problem	$(POP - \mathbb{C}) \begin{cases} \text{several **Variables**} \\ \text{a **Polynomial** objective} \\ \text{several named **Constraints**} \end{cases}$	$\begin{array}{ll} min & x\bar{x} + y^2 + 2 \\ s.t. & 3x + y + 2 \leq 3 + 5im \\ & 2 - im \leq y^2 + 5xy + 2 \leq 3 + 7im \\ & \bar{x}y = 0 \\ & x \in \mathbb{C}, y \in \mathbb{R} \end{array}$

```
x = Variable("x", Complex)
y = Variable("y", Real)
p_obj = abs2(x) + abs2(y) + 2
p_cstr1 = 3*x + y + 2
p_cstr2 = abs2(y) + 5*x*y + 2
p_cstr3 = conj(x)*y
```

```
pb = Problem()
set_objective!(pb, p_obj)
add_constraint!(pb, "Cstr 1", p_cstr1 << 3+5im)
add_constraint!(pb, "Cstr 2", 2-im << p_cstr2 << 3+7im)
add_constraint!(pb, "Cstr 3", p_cstr3 == 0)
```

Conversion to real numbers

Method to convert $(\mathrm{POP}-\mathbb{C})$ to $(\mathrm{POP}-\mathbb{R})$ using rectangular form:

$$
\begin{array}{ll}
min & \dfrac{(1-i)}{2}v_1 + \dfrac{(1+i)}{2}\overline{v_1} \\
s.t. & 0.95 \le v_1\overline{v_1} \le 1.05 \\
 & v_1 \in \mathbb{C}
\end{array}
\quad \xRightarrow{\text{pb\_cplx2real}} \quad
\begin{array}{ll}
min & v_{1Re} + v_{1Im} \\
s.t. & 0.95 \le v_{1Re}^2 + v_{1Im}^2 \le 1.05 \\
 & v_{1Re}, v_{1Im} \in \mathbb{R}
\end{array}
$$

```
V1 = Variable("VOLT_1",Complex)
p_obj = 0.5*((1-im)*V1+(1+im)*conj(V1))
p_ctr1 = abs2(V1)
problem_poly=Problem()
add_variable!(problem_poly,V1)
add_constraint!(problem_poly, "ctr1", 1 << p_ctr1 << 1 )
set_objective!(problem_poly, p_obj)
```

```
pb_poly_real = pb_cplx2real(problem_poly)
```

Conversion to real numbers

Method to convert $(\mathrm{POP}-\mathbb{C})$ to $(\mathrm{POP}-\mathbb{R})$ using rectangular form:

$$\begin{array}{ll} min & \dfrac{(1-i)}{2}v_1 + \dfrac{(1+i)}{2}\overline{v_1} \\ s.t. & 0.95 \leq v_1\overline{v_1} \leq 1.05 \\ & v_1 \in \mathbb{C} \end{array} \xrightarrow{\text{pb\_cplx2real}} \begin{array}{ll} min & v_{1Re} + v_{1Im} \\ s.t. & 0.95 \leq v_{1Re}^2 + v_{1Im}^2 \leq 1.05 \\ & v_{1Re}, v_{1Im} \in \mathbb{R} \end{array}$$

Future work: conversion using polar form

$$\begin{array}{ll} min & \dfrac{(1-i)}{2}v_1 + \dfrac{(1+i)}{2}\overline{v_1} \\ s.t. & 0.95 \leq v_1\overline{v_1} \leq 1.05 \\ & v_1 \in \mathbb{C} \end{array} \Longrightarrow \begin{array}{ll} min & r_1(\cos(\theta_1) + \sin(\theta_1)) \\ s.t. & 0.95 \leq r_1^2 \leq 1.05 \\ & r_1, \theta_1 \in \mathbb{R} \end{array}$$

Resolution

JuMP

```
m, jumpvar = get_JuMP_cartesian_model(pb, solver)

solve(m)
```

```
Final objective value               =  1.45883471040128e+003
Final feasibility error (abs / rel) =   1.44e-007 / 1.15e-009
Final optimality error  (abs / rel) =   3.04e-007 / 3.21e-011
# of iterations                     =        15
# of CG iterations                  =         7
# of function evaluations           =        24
# of gradient evaluations           =        16
# of Hessian evaluations            =        15
Total program time (secs)           =       0.198 (     0.203 CPU time)
Time spent in evaluations (secs)    =       0.163
```

```
Problem Characteristics
-----------------------
Objective goal:  Minimize
Number of variables:                            18
    bounded below only:                          0
    bounded above only:                          0
    bounded below and above:                     0
    fixed:                                       0
    free:                                       18
Number of constraints:                          27
    linear equalities:                           0
    nonlinear equalities:                       12
    linear one-sided inequalities:               0
    nonlinear one-sided inequalities:            0
    linear two-sided inequalities:               0
    nonlinear two-sided inequalities:           15
Number of nonzeros in Jacobian:
126
Number of nonzeros in Hessian:                  54
```

Resolution

JuMP	AMPL

`m, jumpvar = get_JuMP_cartesian_model(pb, solver)`
 	
`solve(m)`
	`export_to_dat(pb, amplexportpath, point)`
	
	`run_knitro(amplexportpath, amplscriptpath)`
	
	`pt_knitro = read_Knitro_output(amplexportpath, pb)`
	
	`feas,ctr = get_minslack(pb, pt_knitro)`
	`objective = get_objective(pb, pt_knitro)`

Final objective value = 1.45883471040128e+003 Final feasibility error (abs / rel) = 1.44e-007 / 1.15e-009 Final optimality error (abs / rel) = 3.04e-007 / 3.21e-011 # of iterations = 15 # of CG iterations = 7 # of function evaluations = 24 # of gradient evaluations = 16 # of Hessian evaluations = 15 Total program time (secs) = 0.198 (0.203 CPU time) Time spent in evaluations (secs) = **0.163**	Final objective value = 1.45883471040144e+003 Final feasibility error (abs / rel) = 1.43e-007 / 1.15e-009 Final optimality error (abs / rel) = 3.03e-007 / 3.20e-011 # of iterations = 15 # of CG iterations = 7 # of function evaluations = 24 # of gradient evaluations = 16 # of Hessian evaluations = 15 Total program time (secs) = 0.005 (0.000 CPU time) Time spent in evaluations (secs) = **0.001**

18 variables, 27 non linear constraints

Resolution

JuMP	AMPL

`m, jumpvar = get_JuMP_cartesian_model(pb, solver)`
 	
`solve(m)`
	`export_to_dat(pb, amplexportpath, point)`
	
	`run_knitro(amplexportpath, amplscriptpath)`
	
	`pt_knitro = read_Knitro_output(amplexportpath, pb)`
	
	`feas,ctr = get_minslack(pb, pt_knitro)`
	`objective = get_objective(pb, pt_knitro)`

Final objective value = 1.33980721247613e+005 Final feasibility error (abs / rel) = 1.58e-008 / 4.09e-012 Final optimality error (abs / rel) = 2.14e-006 / 2.14e-012 # of iterations = 48 # of CG iterations = 24 # of function evaluations = 49 # of gradient evaluations = 49 # of Hessian evaluations = 48 Total program time (secs) = 26.224 (26.000 CPU time) Time spent in evaluations (secs) = **24.457**	Final objective value = 1.33980721261059e+005 Final feasibility error (abs / rel) = 4.21e-007 / 1.09e-010 Final optimality error (abs / rel) = 5.41e-004 / 5.99e-010 # of iterations = 47 # of CG iterations = 24 # of function evaluations = 48 # of gradient evaluations = 48 # of Hessian evaluations = 47 Total program time (secs) = 2.548 (2.531 CPU time) Time spent in evaluations (secs) = **1.093**

5378 variables, 8607 non linear constraints

2

Lasserre hierarchy for $(POP - \mathbb{C})$

•

SemiDefinite Programming (SDP) relaxations of $(POP - \mathbb{C})$

$$(POP-\mathbb{C})\begin{cases} \min & f(z) = \sum_{\alpha,\beta} f^0_{\alpha,\beta}\, \overline{z}^{\alpha} z^{\beta} \\ s.t. & g_i(z) = \sum_{\alpha,\beta} g^i_{\alpha,\beta}\, \overline{z}^{\alpha} z^{\beta} \geq 0 \quad \forall i = 1..m \\ & z \in \mathbb{C}^n \end{cases}$$

$$\Downarrow$$

Several SDP relaxations tighter and tighter (convergent hierarchy)

$$(SDP)\begin{cases} min & C \cdot X \\ & A_i \cdot X \leq b_i \quad \forall i = 1..m \\ & X \succcurlyeq 0 \end{cases} \qquad (dSDP)\begin{cases} max & b^T y \\ & \sum_{i}^{m} A_i y_i + S = C \\ & S \succcurlyeq 0 \end{cases}$$

Moment matrices

$$z_d = \begin{pmatrix} 1 & z_1 & z_2 & \dots & z_{n-1}z_n^{d-1} & z_n^d \end{pmatrix}^T$$

$$\mathcal{M}_d(z) = z_d z_d^H$$

Order d	0	1	2
z_d	(1)	$\begin{pmatrix} 1 & z_1 & z_2 \end{pmatrix}$	$\begin{pmatrix} 1 & z_1 & z_2 & z_1 z_2 & z_1^2 & z_2^2 \end{pmatrix}$
$\mathcal{M}_d(z)$	(1)	$\begin{pmatrix} 1 & z_1 & z_2 \\ \bar{z}_1 & \lvert z_1\rvert^2 & \bar{z}_1 z_2 \\ \bar{z}_2 & \bar{z}_2 z_1 & \lvert z_2\rvert^2 \end{pmatrix}$	$\begin{pmatrix} 1 & z_1 & z_2 & z_1 z_2 & z_1^2 & z_2^2 \\ \bar{z}_1 & \lvert z_1\rvert^2 & \bar{z}_1 z_2 & \lvert z_1\rvert^2 z_2 & \lvert z_1\rvert^2 z_1 & \bar{z}_1 z_2^2 \\ \bar{z}_2 & \bar{z}_2 z_1 & \lvert z_2\rvert^2 & \lvert z_2\rvert^2 z_1 & \bar{z}_2 z_1^2 & \lvert z_2\rvert^2 z_2 \\ \bar{z}_1 \bar{z}_2 & \lvert z_1\rvert^2 \bar{z}_2 & \lvert z_2\rvert^2 \bar{z}_1 & \lvert z_1\rvert^2 \lvert z_2\rvert^2 & \lvert z_1\rvert^2 z_1 \bar{z}_2 & \lvert z_2\rvert^2 \bar{z}_1 z_2 \\ \bar{z}_1^2 & \lvert z_1\rvert^2 \bar{z}_1 & \bar{z}_1^2 z_2 & \lvert z_1\rvert^2 \bar{z}_1 z_2 & \lvert z_1\rvert^4 & \bar{z}_1^2 z_2^2 \\ \bar{z}_2^2 & \bar{z}_2^2 z_1 & \lvert z_2\rvert^2 \bar{z}_2 & \lvert z_2\rvert^2 z_1 \bar{z}_2 & \bar{z}_2^2 z_1^2 & \lvert z_2\rvert^4 \end{pmatrix}$

Increasing the order improves the quality of the relaxation but **increases significantly the size of the problem**.

SDP relaxation

$$\begin{cases} \min & f(z) \\ s.t. & g_i(z) \geq 0 \quad \forall i = 1..m \\ & z \in \mathbb{C}^n \end{cases} \Leftrightarrow \begin{cases} \min & f(z) \\ s.t. & g_i(z)\mathcal{M}_{d-k_i}(z) \succcurlyeq 0 \quad \forall i = 1..m \\ & \mathcal{M}_d(y) = z_d z_d^H \end{cases}$$

Let us denote $y_{\alpha\beta} = \mathcal{M}_d(z)[\alpha,\beta]$

$$\Leftrightarrow \begin{cases} \min & f(y) \\ s.t. & \mathcal{M}_{d-k_i}(g_i y) \succcurlyeq 0 \quad \forall i = 1..m \\ & \mathcal{M}_d(y) = z_d z_d^H \end{cases}$$

SDP relaxation

$$\begin{cases} \min & f(z) \\ s.t. & g_i(z) \geq 0 \quad \forall i = 1..m \\ & z \in \mathbb{C}^n \end{cases} \Leftrightarrow \begin{cases} \min & f(z) \\ s.t. & g_i(z)\mathcal{M}_{d-k_i}(z) \succcurlyeq 0 \quad \forall i = 1..m \\ & \mathcal{M}_d(y) = z_d z_d^H \end{cases}$$

Let us denote $y_{\alpha\beta} = \mathcal{M}_d(z)[\alpha,\beta]$

$$\Leftrightarrow \begin{cases} \min & f(y) \\ s.t. & \mathcal{M}_{d-k_i}(g_i y) \succcurlyeq 0 \quad \forall i = 1..m \\ & \begin{cases} \mathcal{M}_d(y) \succcurlyeq 0 \\ rank(\mathcal{M}_d(y)) = 1 \end{cases} \end{cases}$$

SDP relaxation

$$\begin{cases} \min & f(z) \\ s.t. & g_i(z) \geq 0 \quad \forall i = 1..m \\ & z \in \mathbb{C}^n \end{cases} \Leftrightarrow \begin{cases} \min & f(z) \\ s.t. & g_i(z)\mathcal{M}_{d-k_i}(z) \succcurlyeq 0 \quad \forall i = 1..m \\ & \mathcal{M}_d(y) = z_d z_d^H \end{cases}$$

Let us denote $y_{\alpha\beta} = \mathcal{M}_d(z)[\alpha, \beta]$

$$\Rightarrow \begin{cases} \min & f(y) \\ s.t. & \mathcal{M}_{d-k_i}(g_i y) \succcurlyeq 0 \qquad \forall i = 1..m \\ & \begin{cases} \mathcal{M}_d(y) \succcurlyeq 0 \\ \cancel{rank(\mathcal{M}_d(y)) = 1} \end{cases} \end{cases}$$

SDP relaxation

$$\begin{cases} \min & f(z) \\ s.t. & g_i(z) \geq 0 \quad \forall i = 1..m \\ & z \in \mathbb{C}^n \end{cases} \Leftrightarrow \begin{cases} \min & f(z) \\ s.t. & g_i(z)\mathcal{M}_{d-k_i}(z) \succcurlyeq 0 \quad \forall i = 1..m \\ & \mathcal{M}_d(y) = z_d z_d^H \end{cases}$$

Let us denote $y_{\alpha\beta} = \mathcal{M}_d(z)[\alpha, \beta]$

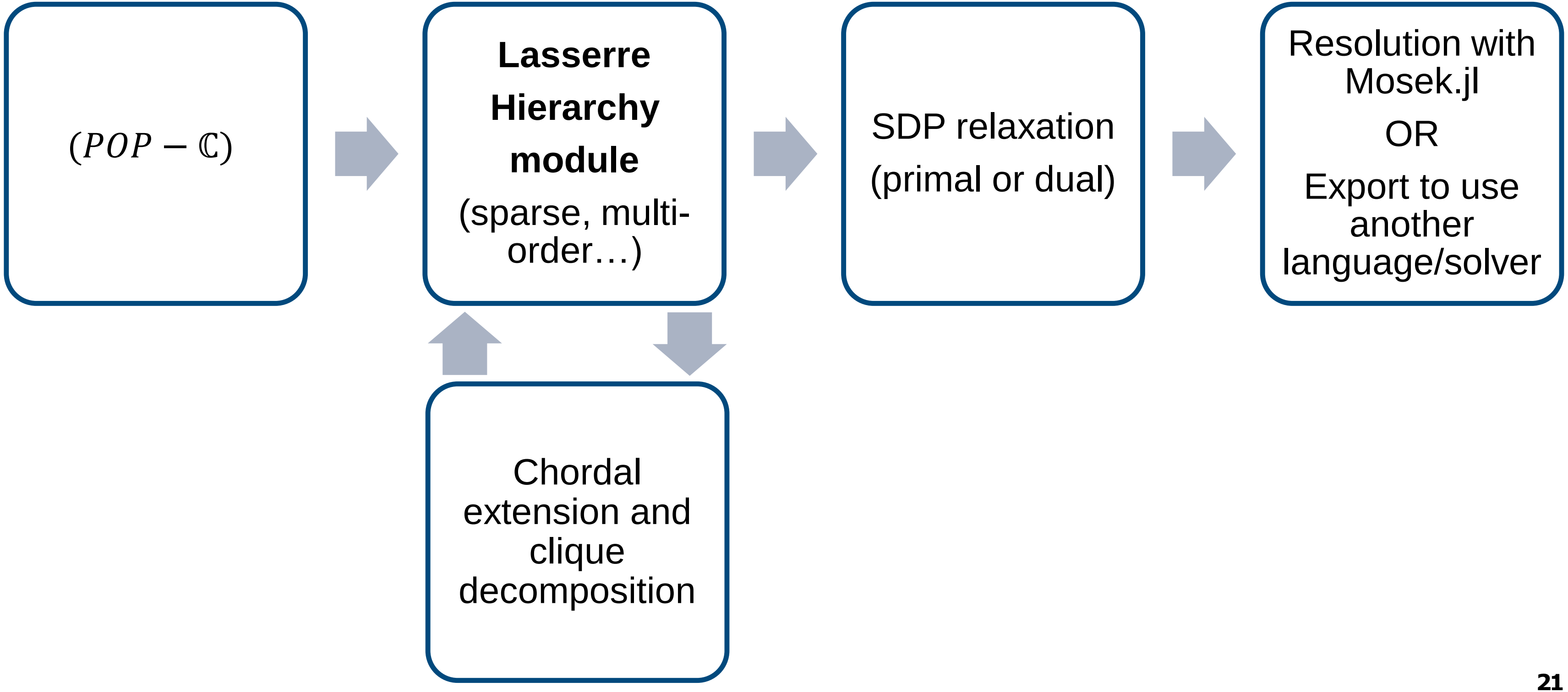
$$\Leftrightarrow \begin{cases} \min & f(y) \\ s.t. & \mathcal{M}_{d-k_i}(g_i y) \succcurlyeq 0 \quad \forall i = 1..m \\ & \begin{cases} \mathcal{M}_d(y) \succcurlyeq 0 \\ rank\left(\mathcal{M}_d(y)\right) = 1 \end{cases} \end{cases} \Rightarrow \boxed{\begin{array}{ll} \min & f(y) \\ s.t. & \mathcal{M}_{d-k_i}(g_i y) \succcurlyeq 0 \quad \forall i = 1..m \\ & \mathcal{M}_d(y) \succcurlyeq 0 \end{array}}$$

Order d relaxation

Available options

- Lasserre hierarchy is workable on **complex or real problems**.
- **Sparsity** is exploited: the set of exponents can be split into smaller cliques.
- **Multi-ordered hierarchy** is possible: different orders can be applied on different constraints.
- Some **symmetries** can be speficied to simplify the problems (for example if $x\ solution \Leftrightarrow -x\ solution$).

Workflow process



3

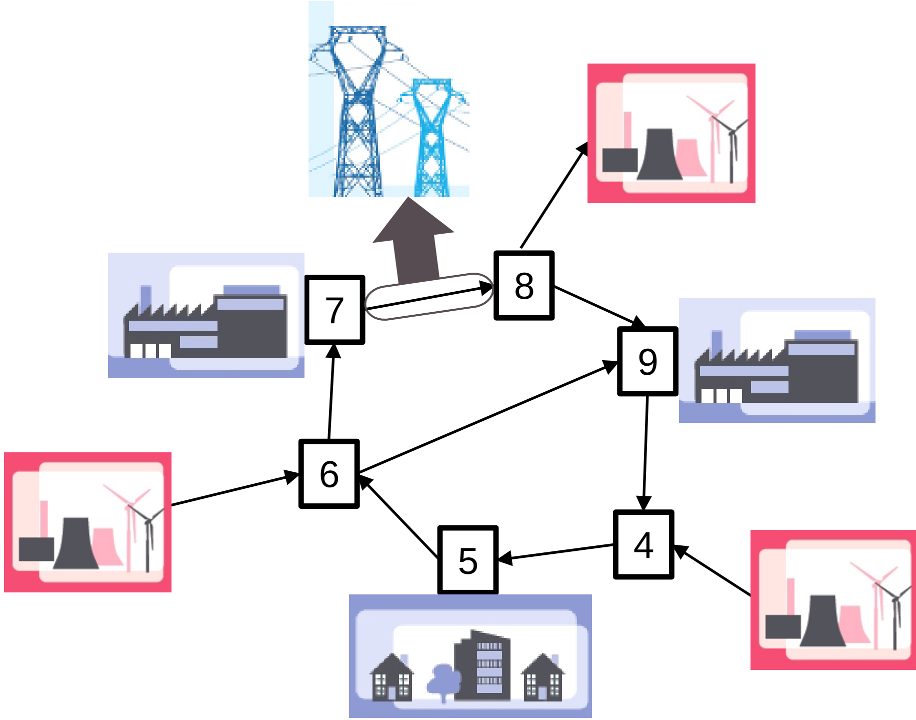
Application to Optimal Power Flow in Alternating Current

Context and motivations

- RTE is the **French transmission system operator** which provides economical, reliable and clean access to electrical power.

- Power transmission networks in Alternating Current involve **complex quantities** (voltage, current, power flows, etc).

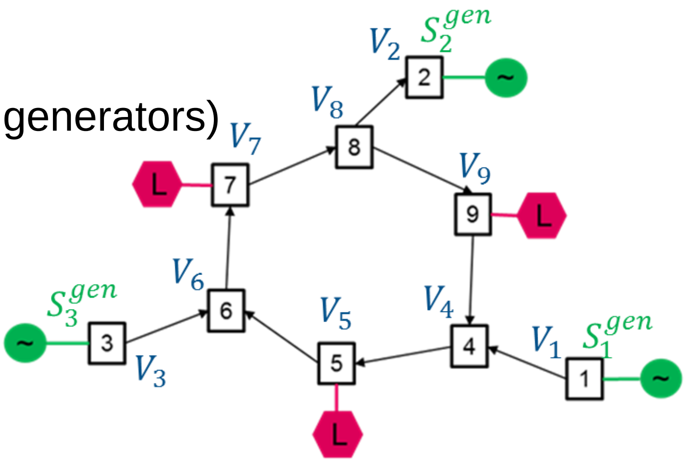
- RTE needs tools for $(POP - \mathbb{C})$ to reduce the time spent in testing methods



Optimal Power Flow in Alternating Current

Variables:

- $V_n \in \mathbb{C}, \forall n \in N$: voltage at bus n
- $S_n^{gen} \in \mathbb{C}, \forall n \in G \subset N$: power at generator bus n (G: set of generators)



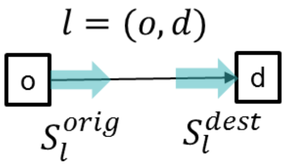
Constraints:

- Power flow equations $\forall n \in N$:

$$S_n^{load} + \sum_{l=(n,d)} S_l^{orig}(V) - \sum_{l=(o,n)} S_l^{dest}(V) = S_n^{gen}$$

Safety constraints

- Generator constraints: $S_n^{min} \leq S_n^{gen} \leq S_n^{max}\ \forall n \in G$
- Voltage magnitude constraints: $(V_n^{min})^2 \leq |V_n|^2 \leq (V_n^{max})^2\ \forall n \in N$
- Thermal limits on branches: $\left|S_l^{orig}(V)\right|^2, \left|S_l^{dest}(V)\right|^2 \leq (S_l^{max})^2\ \forall l \in L$



Minimization of active power generation cost: $\min \sum_{g \in G} c_g(Real(S_g^{gen}))$

Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

…

ACOPF

$(\boldsymbol{POP} - \mathbb{C})$

Conversion to

$(\boldsymbol{POP} - \mathbb{R})$

using **rectangular form**

Computation of **feasible solutions** with AMPL and Knitro

Computation of **lower bounds** using Lasserre hierarchy

Results for the Grid Optimization Competition

- Challenge launched by ARPA-E (Advanced Research Projects Agency-Energy)
- The problem to solve is an ACOPF in which some contigencies are anticipated.
- It can be formulated as a Mixed-Integer Polynomial Optimization Problem with Complex numbers $(MIPOP - \mathbb{C})$.

Dataset	# of buses	# of contingencies	# of real variables	# of constraints	# of nonzeros in Jacobian	# of nonzeros in Hessian	# of solved scenarios
IEEE14	14	1	92	207	937	245	90/100
Modified_IEEE14	14	1	92	203	905	237	84/100
RTS96	73	10	4784	12157	49838	7199	90/100

More information: https://gocompetition.energy.gov/

4

Future works

Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

…

ACOPF

$(\boldsymbol{POP} - \mathbb{C})$

Conversion to

$(\boldsymbol{POP} - \mathbb{R})$

using **rectangular form**

Computation of **feasible solutions** with AMPL and Knitro

Computation of **lower bounds** using Lasserre hiearchy

Collaboration with Carleton Coffrin to integrate a power model in complex variables in **PowerModels.jl** (a Julia/JuMP package for Steady-State Power Network Optimization)

Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

…

ACOPF

$(\boldsymbol{POP} - \mathbb{C})$

Conversion to

$(\boldsymbol{POP} - \mathbb{R})$

using **rectangular form**

Polar form

Computation of **feasible solutions** with AMPL and Knitro

Computation of **lower bounds** using Lasserre hiearchy

Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

…

ACOPF $(\boldsymbol{POP}-\mathbb{C})$

Conversion to $(\boldsymbol{POP}-\mathbb{R})$ using **rectangular form**

Resolution with JuMP

Computation of **feasible solutions** with AMPL and Knitro

Computation of **lower bounds** using Lasserre hiearchy

Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

…

ACOPF $(\boldsymbol{POP}-\mathbb{C})$

Conversion to $(\boldsymbol{POP}-\mathbb{R})$ using **rectangular form**

Computation of **feasible solutions** with AMPL and Knitro

Computation of **lower bounds** using Lasserre hiearchy

Second-Order Cone Programming (SOCP) relaxations

Conclusion and prospects

- Tool for **Polynomial Optimization Problems with Complex numbers** $(POP - \mathbb{C})$. In addition to the **local resolution** (JuMP, AMPL), the **Lasserre hierarchy** for $(POP - \mathbb{C})$ is implemented with several options to compute lower bounds.

- The application to **Optimal Power Flow problems in Alternating Current** (ACOPF) demonstrates the convenience of such a toolbox. May be convenient for other problems.

- **Still in development**

- Creation of Julia packages?
- Contribution into existing Julia packages (PolyJuMP.jl, SumOfSquares.jl, MultivariatePolynomials.jl)?

$\Rightarrow$ We are looking for Julia developers to support RTE research around this tool.

Thank you for your attention!

Any questions?

•

{manuel.ruiz, julie.sliwak}@rte-france.com
gilles.bareilles@artelys.com

https://github.com/JulieSliwak/MathProgComplex.jl