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 (\(\text{POP} - \mathbb{C}\))
2. The Lasserre hierarchy for (\(\text{POP} - \mathbb{C}\))
3. Application to Optimal Power Flow in Alternating Current (ACOPF)
4. Conclusion and future work

https://github.com/JulieSliwak/MathProgComplex.jl (MIT license)
A tool for Polynomial Optimization Problems with Complex variables ($POP - \mathbb{C}$)
Polynomial Optimization Problems with Complex Variables \( (POP - \mathbb{C}) \)

\[
\begin{align*}
\min & \quad \sum_{\alpha, \beta} p_{\alpha, \beta}^0 \overline{z}^\alpha z^\beta \\
\text{s.t.} & \quad \sum_{\alpha, \beta} p_{\alpha, \beta}^i \overline{z}^\alpha z^\beta \geq 0 \quad \forall i = 1..p \\
& \quad z \in \mathbb{C}^n
\end{align*}
\]

- 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 \((POP – ℂ)\)

• Our modeler provides a structure and methods for working with \((POP – ℂ)\).

• The algebraic operations \((+,-,\ast,/,\text{conj},|.|)\) are implemented.

• The base type is \textbf{Variable}, from which \textbf{Exponents} and \textbf{Polynomial} can be constructed by calling the respective constructors or with algebraic operations.

• The \textbf{Point} type holds the variables at which polynomials can be evaluated.
# Basic structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Definition</th>
<th>Notation</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable</strong></td>
<td>A pair (String, Type) where Type can be Complex, Real or Bool</td>
<td>$z$</td>
<td>$x = \text{Variable}(&quot;x&quot;, \text{Complex})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$y = \text{Variable}(&quot;y&quot;, \text{Complex})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$w = \text{Variable}(&quot;w&quot;, \text{Real})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$u = \text{Variable}(&quot;binary&quot;, \text{Bool})$</td>
</tr>
<tr>
<td><strong>Exponent</strong></td>
<td>A product of Variables</td>
<td>$\prod z_i^{\alpha_i} z_i^{\beta_i}$</td>
<td>$\text{expo1} = x^2 \text{conj}(y)^3$, $\text{expo2} = xy$</td>
</tr>
<tr>
<td><strong>Polynomial</strong></td>
<td>A sum of Exponents times complex coefficient</td>
<td>$\sum c_k \prod k_i z_{k_i}^{\alpha_{k_i}} z_{k_i}^{\beta_{k_i}}$</td>
<td>$p(x, y) = (1 + 4im)\text{expo1} + 3\text{expo2}$</td>
</tr>
<tr>
<td><strong>Point</strong></td>
<td>A dictionary (variable =&gt; value) to evaluate a polynomial</td>
<td>$\begin{pmatrix} z_1 \ \vdots \ z_k \end{pmatrix} = \begin{pmatrix} \text{value}_1 \ \vdots \ \text{value}_k \end{pmatrix}$</td>
<td>$pt = \text{Dict}(x \Rightarrow 1 + 2im, y \Rightarrow 3im)$</td>
</tr>
</tbody>
</table>

$$\text{evaluate}(p, pt) = -145 + 28im$$
# Polynomial Optimization Problems

<table>
<thead>
<tr>
<th>Structure</th>
<th>Definition</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint</td>
<td>A Polynomial with complex bounds</td>
<td>$3x + y + 2 \leq 3 + 5im$</td>
</tr>
</tbody>
</table>
| Problem    | $(POP - \mathbb{C}) \left\{ \begin{array}{l} \text{several Variables} \\ \text{a Polynomial objective} \\ \text{several named Constraints} \end{array} \right.$ | $\begin{align*} 
\min \\
\text{s.t.} \\
3x + y + 2 &\leq 3 + 5im \\
2 - im &\leq y^2 + 5xy + 2 \leq 3 + 7im \\
\bar{x}y &= 0 \\
x &\in \mathbb{C}, y \in \mathbb{R} 
\end{align*}$ |
Conversion to real numbers

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

\[
\begin{align*}
\min & \quad \frac{(1-i)}{2} v_1 + \frac{(1+i)}{2} \overline{v_1} \\
\text{s.t.} & \quad 0.95 \leq v_1 \overline{v_1} \leq 1.05 \\
& \quad v_1 \in \mathbb{C}
\end{align*}
\]

\[
\begin{align*}
\min \quad & \quad v_{1Re} + v_{1Im} \\
\text{s.t.} & \quad 0.95 \leq v_{1Re}^2 + v_{1Im}^2 \leq 1.05 \\
& \quad v_{1Re}, v_{1Im} \in \mathbb{R}
\end{align*}
\]

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

\[
\text{pb\_poly\_real} = \text{pb\_cplx2real}(\text{problem\_poly})
\]
Conversion to real numbers

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

\[
\begin{align*}
\min & \quad \frac{1-i}{2} v_1 + \frac{1+i}{2} \bar{v}_1 \\
\text{s.t.} & \quad 0.95 \leq v_1 \bar{v}_1 \leq 1.05 \\
& \quad v_1 \in \mathbb{C}
\end{align*}
\]

\[
\begin{align*}
\min & \quad v_{1\text{Re}} + v_{1\text{Im}} \\
\text{s.t.} & \quad 0.95 \leq v_{1\text{Re}}^2 + v_{1\text{Im}}^2 \leq 1.05 \\
& \quad v_{1\text{Re}}, v_{1\text{Im}} \in \mathbb{R}
\end{align*}
\]

Future work: conversion using polar form

\[
\begin{align*}
\min & \quad \frac{1-i}{2} v_1 + \frac{1+i}{2} \bar{v}_1 \\
\text{s.t.} & \quad 0.95 \leq v_1 \bar{v}_1 \leq 1.05 \\
& \quad v_1 \in \mathbb{C}
\end{align*}
\]

\[
\begin{align*}
\min & \quad r_1(\cos(\theta_1) + \sin(\theta_1)) \\
\text{s.t.} & \quad 0.95 \leq r_1^2 \leq 1.05 \\
& \quad r_1, \theta_1 \in \mathbb{R}
\end{align*}
\]
Resolution

JuMP

\[
\text{m, jumpvar} = \text{get\_JuMP\_cartesian\_model}(\text{pb, solver})
\]

\[
\text{solve(m)}
\]

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

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
Resolution

<table>
<thead>
<tr>
<th>JuMP</th>
<th>AMPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>m, jumpvar = get_JuMP_cartesian_model(pb, solver)</td>
<td>export_to_dat(pb, amplexportpath, point)</td>
</tr>
<tr>
<td>solve(m)</td>
<td>run_knitro(amplexportpath, amplscriptpath)</td>
</tr>
<tr>
<td></td>
<td>pt_knitro = read_Knitro_output(amplexportpath, pb)</td>
</tr>
<tr>
<td></td>
<td>feas,ctr = get_minslack(pb, pt_knitro)</td>
</tr>
<tr>
<td></td>
<td>objective = get_objective(pb, pt_knitro)</td>
</tr>
</tbody>
</table>

| 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

<table>
<thead>
<tr>
<th><strong>JuMP</strong></th>
<th><strong>AMPL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>m, jumpvar = get_JuMP_cartesian_model(pb, solver)</code></td>
<td><code>export_to_dat(pb, amplelexportpath, point)</code></td>
</tr>
<tr>
<td><code>solve(m)</code></td>
<td><code>run_knitro(amplelexportpath, amplscriptpath)</code></td>
</tr>
<tr>
<td></td>
<td><code>pt_knitro = read_Knitro_output(amplelexportpath, pb)</code></td>
</tr>
<tr>
<td></td>
<td><code>feas,ctr = get_minslack(pb, pt_knitro)</code></td>
</tr>
<tr>
<td></td>
<td><code>objective = get_objective(pb, pt_knitro)</code></td>
</tr>
</tbody>
</table>

| | Final objective value | 1.3398072127613e+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
Lasserre hierarchy for $(POP - \mathbb{C})$
SemiDefinite Programming (SDP) relaxations of \((POP - \mathbb{C})\)

\[
\begin{align*}
\min_{\alpha, \beta} f(z) &= \sum_{\alpha, \beta} f_{\alpha, \beta}^0 z^\alpha z^\beta \\
\text{s.t. } g_i(z) &= \sum_{\alpha, \beta} g_{\alpha, \beta}^i z^\alpha z^\beta \geq 0 \quad \forall i = 1..m \\
z &\in \mathbb{C}^n
\end{align*}
\]

Several SDP relaxations tighter and tighter (convergent hierarchy)

\[
\begin{align*}
\text{(SDP)}: \min_{X} C \cdot X &\leq b_i \quad \forall i = 1..m \\
\quad X &\succeq 0 \\
\text{(dSDP)}: \max_{y} b^T y &\leq \sum_{i} A_i y_i + S = C \\
\quad S &\succeq 0
\end{align*}
\]
# Moment matrices

\[ z_d = (1 \ z_1 \ z_2 \ \ldots \ \ z_{n-1} z_{n-1}^{d-1} \ z_n^d)^T \]

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

<table>
<thead>
<tr>
<th>Order ( d )</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( z_d )</td>
<td>(1)</td>
<td>((1 \ z_1 \ z_2))</td>
<td>((1 \ z_1 \ z_2 \ z_{12} \ z_1^2 \ z_2^2))</td>
</tr>
</tbody>
</table>
| \( \mathcal{M}_d(z) \) | (1) | \[
\begin{pmatrix}
1 & z_1 & z_2 \\
\overline{z}_1 & |z_1|^2 & \overline{z}_1 z_2 \\
\overline{z}_2 & \overline{z}_2 z_1 & |z_2|^2
\end{pmatrix}
\] | \[
\begin{pmatrix}
1 & z_1 & z_2 & z_1 z_2 & z_1^2 & z_2^2 \\
\overline{z}_1 & |z_1|^2 & \overline{z}_1 z_2 & |z_1|^2 z_2 & |z_1|^2 z_1 & \overline{z}_1 z_2^2 \\
\overline{z}_2 & \overline{z}_2 z_1 & |z_2|^2 & |z_2|^2 z_1 & \overline{z}_2 z_1^2 & |z_2|^2 z_2 \\
\overline{z}_1 \overline{z}_2 & |z_1|^2 \overline{z}_2 & |z_2|^2 \overline{z}_1 & |z_1|^2 |z_2|^2 & |z_1|^2 z_1 \overline{z}_2 & |z_2|^2 \overline{z}_1 z_2 \\
\overline{z}_1^2 & |z_1|^2 \overline{z}_1 & \overline{z}_1^2 z_2 & |z_1|^2 \overline{z}_1 z_2 & |z_1|^4 & \overline{z}_1^2 z_2^2 \\
\overline{z}_2^2 & \overline{z}_2^2 z_1 & |z_2|^2 \overline{z}_2 & |z_2|^2 z_1 \overline{z}_2 & \overline{z}_2^2 z_1^2 & |z_2|^4
\end{pmatrix}
\] |

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

\[
\begin{align*}
\min \quad & f(z) \\
\text{s.t.} \quad & g_i(z) \geq 0 \quad \forall i = 1..m \\
& z \in \mathbb{C}^n
\end{align*}
\]

\[
\begin{align*}
\min \quad & f(z) \\
\text{s.t.} \quad & g_i(z)M_{d-k_i}(z) \geq 0 \quad \forall i = 1..m
\end{align*}
\]

Let us denote \( y_{\alpha\beta} = M_d(z)[\alpha, \beta] \)

\[
\begin{align*}
\min \quad & f(y) \\
\text{s.t.} \quad & M_{d-k_i}(g_iy) \geq 0 \quad \forall i = 1..m \\
& M_d(y) = z_d z_d^H
\end{align*}
\]
SDP relaxation

\[
\begin{align*}
\begin{cases}
\min & f(z) \\
\text{s.t.} & g_i(z) \geq 0 \quad \forall i = 1..m \\
& z \in \mathbb{C}^n
\end{cases}
\iff
\begin{cases}
\min & f(z) \\
\text{s.t.} & g_i(z)\mathcal{M}_{d-k_i}(z) \succeq 0 \quad \forall i = 1..m \\
& \mathcal{M}_d(y) = z_dz_d^H
\end{cases}
\end{align*}
\]

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

\[
\begin{align*}
\iff
\begin{cases}
\min & f(y) \\
\text{s.t.} & \mathcal{M}_{d-k_i}(g_iy) \succeq 0 \quad \forall i = 1..m \\
& \mathcal{M}_d(y) \succeq 0 \\
& \text{rank}(\mathcal{M}_d(y)) = 1
\end{cases}
\end{align*}
\]
SDP relaxation

$$\begin{align*}
\begin{cases}
\min & f(z) \\
\text{s.t.} & g_i(z) \geq 0 \quad \forall i = 1..m \\
& z \in \mathbb{C}^n
\end{cases}
\iff
\begin{cases}
\min & f(z) \\
\text{s.t.} & g_i(z) M_{d-k_i}(z) \succeq 0 \quad \forall i = 1..m \\
& M_d(y) = z_d z_d^H
\end{cases}
\end{align*}$$

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

$$\begin{align*}
\begin{cases}
\min & f(y) \\
\text{s.t.} & M_{d-k_i}(g_iy) \succeq 0 \quad \forall i = 1..m
\end{cases}
\Rightarrow
\begin{cases}
& M_d(y) \succeq 0 \\
& \text{rank}(M_{\alpha\beta}(y)) = 1
\end{cases}
\end{align*}$$
SDP relaxation

\[
\begin{align*}
\begin{cases}
\min & f(z) \\
\text{s.t.} & g_i(z) \geq 0 \quad \forall i = 1..m \\
& z \in \mathbb{C}^n
\end{cases} & \iff \\
\begin{cases}
\min & f(z) \\
\text{s.t.} & g_i(z)M_{d-k_i}(z) \succeq 0 \quad \forall i = 1..m \\
& M_d(y) = z_dz_d^H
\end{cases}
\end{align*}
\]

Let us denote \( y_{\alpha\beta} = M_d(z)[\alpha, \beta] \)

\[
\begin{align*}
\begin{cases}
\min & f(y) \\
\text{s.t.} & M_{d-k_i}(g_iy) \succeq 0 \quad \forall i = 1..m \\
& M_d(y) \succeq 0 \\
& \text{rank}(M_d(y)) = 1
\end{cases} & \iff \\
\begin{cases}
\min & f(y) \\
\text{s.t.} & M_{d-k_i}(g_iy) \succeq 0 \quad \forall i = 1..m \\
& M_d(y) \succeq 0
\end{cases}
\end{align*}
\]

\text{Order } d \text{ 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 specified to simplify the problems (for example if $x$ solution $\Leftrightarrow -x$ solution).
Workflow process

$(POP - \mathbb{C})$

Lasserre Hierarchy module
(sparse, multi-order…)

SDP relaxation
(primal or dual)

Resolution with Mosek.jl
OR
Export to use another language/solver

Chordal extension and clique decomposition
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 \((POP - C)\) 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 \( \forall n \in N \):
  \[
  S_n^{load} + \sum_{l=(n,d)} S_l^{orig}(V) - \sum_{l=(o,n)} S_l^{dest}(V) = S_n^{gen}
  \]
- 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: \( |S_l^{orig}(V)|^2, |S_l^{dest}(V)|^2 \leq (S_l^{max})^2 \forall l \in L \)

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

Matpower

PSSE

Grid Optimization Competition

... 

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

Conversion to \((POP - \mathbb{R})\) 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 contingencies are anticipated.
- It can be formulated as a Mixed-Integer Polynomial Optimization Problem with Complex numbers ($MIP_{OP} - \mathbb{C}$).

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

More information: [https://gocompetition.energy.gov/](https://gocompetition.energy.gov/)
Future works

.
Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

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

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

Computation of feasible solutions with AMPL and Knitro

Computation of lower bounds using Lasserre hierarchy

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

...
Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

…

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

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

Resolution with JuMP

Computation of feasible solutions with AMPL and Knitro

Computation of lower bounds using Lasserre hierarchy
Resolution of ACOPF

Matpower

PSSE

Grid Optimization Competition

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

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

Computation of feasible solutions with AMPL and Knitro

Computation of lower bounds using Lasserre hierarchy

Second-Order Cone Programming (SOCP) relaxations
Conclusion and prospects

• Tool for **Polynomial Optimization Problems with Complex numbers** \((PO\!P - \mathbb{C})\). In addition to the **local resolution** (JuMP, AMPL), the **Lasserre hierarchy** for \((PO\!P - \mathbb{C})\) 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)?

⇒ 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