YALMIP: Optimization Made Easy!

and

Modeling Languages/Layers for Optimization in general

Pierre Haessig
CentraleSupélec Rennes, April 6th, 2017
Where is human time spent?

1) Formalize the problem into a Mathematical Optimization problem → **core skill** of the researcher

2) Transform to a canonical form (**solver specific API**) 

   *[now some computing time…]*

3) Retrieve results out of the canonical format (**again solver specific**)

*YALMIP can help the researcher focus on its **core skill***
## Modeling Languages/Layers for Optimization

<table>
<thead>
<tr>
<th>Environment</th>
<th>Software/Toolbox/ Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone</td>
<td>AMPL, GAMS (~1990)</td>
</tr>
<tr>
<td>Matlab</td>
<td>YALMIP, CVX (~ 2000)</td>
</tr>
<tr>
<td>Python</td>
<td>Pyomo, PuLP (MILP only), CVXPY</td>
</tr>
<tr>
<td>Julia</td>
<td>JuMP, Convex.jl</td>
</tr>
</tbody>
</table>
Relationship to the optimization solver

• When using a modeling environment, the solver is *mostly hidden* from the user.

• The choice of the modeling layer is (mostly) independent of the choice of solver.

For ex. YALMIP provides interfaces for most common solvers:

- Gurobi, CPLEX, MOSEK (Commercial)
- GLPK, Ipopt, SEDUMI (Free)
- linprog from Matlab Optimization Toolbox
Matlab Tooboxes

- **YALMIP**: 2001 – present, J. Löfberg
  from Linköping University, (post-doc at ETH Zürich)

- **CVX**: 2005 – present, M. Grant and S. Boyd.
  from Stanford University
  *(compatibility problem with Matlab 2017)*
  
  - M. Grant joined Continuum Analytics in 2015
    (platforms for Data Science, mostly Python).
YALMIP quickstart

1) **Download** ZIP archive and unzip the archive in some folder. ([https://yalmip.github.io/download/](https://yalmip.github.io/download/))

2) Add YALMIP folders (with subfolders) to the **MATLAB path**. (cf. [https://yalmip.github.io/tutorial/installation/](https://yalmip.github.io/tutorial/installation/))

3) Start using it! You can look at the “Getting Started” tutorial. ([https://yalmip.github.io/tutorial/basics/](https://yalmip.github.io/tutorial/basics/))
Demo: Grid-connected PV-storage system

cf. PVgrid.m script
Wrap up: advantages/drawbacks

Key advantages:

● Shorter starting time (for students), shorter development time

● Increased agility (→ better research!)
  – Quickly compare solvers
  – Quickly compare different problem models (e.g. LP vs. QP)

But maybe:

● Computational overhead?
  – e.g. less efficient when recycling the problem (like for MPC)?
Application to Embedded Optimization

Example of an **Autonomous Driving RC Car** “BARC Project”

- Implementation with Julia + **JuMP** (+ **ROS** + ...)
- Presentation by Jon Gonzales (Berkeley MPC Lab) at JuliaCon 2016 [https://www.youtube.com/watch?v=bX4TXWO7dA0](https://www.youtube.com/watch?v=bX4TXWO7dA0)
Standalone commercial modeling languages

- **AMPL** (A Mathematical Programming Language) [http://ampl.com/](http://ampl.com/)
  - started ~1985 at Bell labs
  - “AMPL Optimization LLC” spun-off in 2002.

- **GAMS** (General Algebraic Modeling System) [https://www.gams.com](https://www.gams.com)
  - started in 1970s at the World Bank (an economic modeling group)
  - commercial product by “GAMS Developement Corp.” since 1987
Some references, in chronological order


