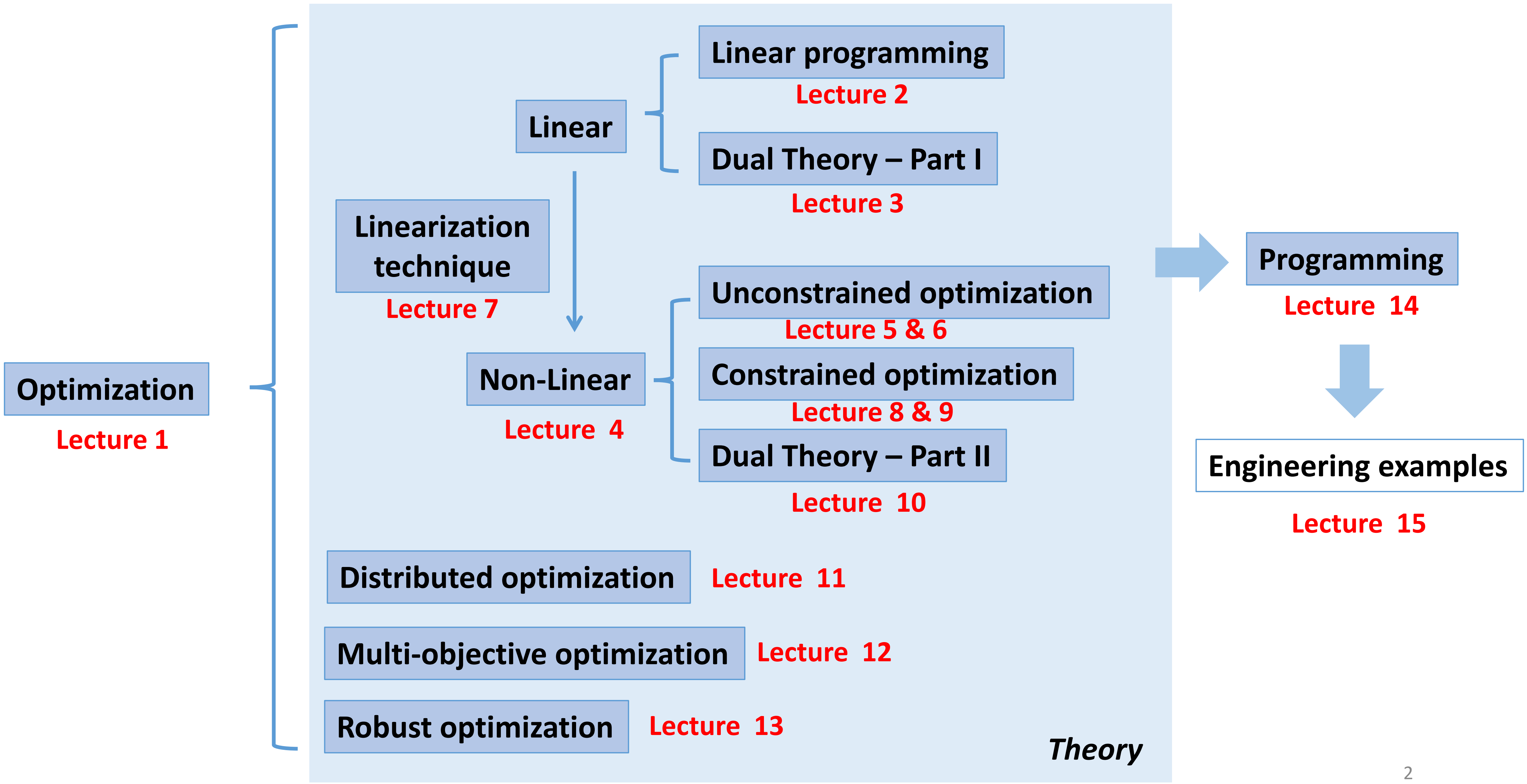
MAEG4070 Engineering Optimization

Lecture 14 Solving optimization using Matlab/Python

Yue Chen
MAE, CUHK
email: yuechen@mae.cuhk.edu.hk
Nov 21, 2022

Content of this course (tentative)



Convex optimization solvers

- **LP Solvers**
  - ✓ Cplex, Gurobi, GLPK, Excel, Matlab's linprog, …
- **Cone solvers**
  - ✓ Typically handle combinations of LP, SOCP, SDP cones
  - ✓ SDPT3, SeDuMi, CSDP, …
- **General convex solvers**
  - ✓ CVXOPT, MOSEK, …
- **Others**
  - ✓ Some solvers developed for specific purpose or application

Example-1 Linear Programming

A company has some resources to produce three products (denoted as A, B, C). Each product consumes a different mix of resources, and there will be a profit from selling the product. The endowment of resources and its relationship with products are:

	A	B	C	Endowment
Steel	3	4	2	600
Wood	2	1	2	400
Label	1	3	3	300
Machine	1	4	4	200
Profit	2	4	3	

Question: How to maximize the total profit?

Example-1 Linear Programming

$$\max_{x_1,x_2,x_3} \quad 2x_1 + 4x_2 + 3x_3$$
$$\text{s.t.} \quad 3x_1 + 4x_2 + 2x_3 \leq 600$$
$$2x_1 + x_2 + 3x_3 \leq 400$$
$$x_1 + 3x_2 + 3x_3 \leq 300$$
$$x_1 + 4x_2 + 4x_3 \leq 200$$
$$x_1, x_2, x_3 \geq 0$$

Endowment limits

Production must larger than 0

Solve it by Matlab's **linprog**

[x, fval] = linprog(f, A, b, Aeq, beq, lb, ub) solves for the optimal solution x and optimal value fval of

$$\min_x f^T x$$
$$\text{s.t.} \quad A \cdot x \leq b$$
$$Aeq \cdot x = beq$$
$$lb \leq x \leq ub$$

Example 1 – Linear Programming

In this case $f = [2, 4, 3]$

$$A = \begin{bmatrix} 3 & 4 & 2 \\ 2 & 1 & 3 \\ 1 & 3 & 3 \\ 1 & 4 & 4 \end{bmatrix}$$

$$b = [600, 400, 300, 200]^T$$

$$Aeq = [], beq = []$$

$$lb = [0, 0, 0]^T, ub = []$$

Code:

```
f = [2,4,3];
A = [3,4,2; 2,1,3; 1,3,3; 1,4,4];
b = [600,400,300,200];
lb = [0,0,0];
[x,fval] = linprog(f,A,b,[],[],lb);
```

More examples: https://www.mathworks.com/help/optim/ug/linprog.html

Example 2 – Constrained Nonlinear optimization

Consider this optimization problem:

$$\min_{x_1, x_2} \; 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

$$\text{s.t. } x_1 + 2x_2 \le 1$$

$$2x_1 + x_2 = 1$$

Solve it by Matlab's **fmincon**

[x,fval] = fmincon(fun,x0,A,b,Aeq,beq,lb,ub) solves for the optimal solution x and optimal value fval of

$$\min_x f(x)$$

$$\text{s.t. } A \cdot x \le b$$

$$Aeq \cdot x = beq$$

$$lb \le x \le ub$$

x0 is the initial point.

Example 2 – Constrained Nonlinear optimization

In this case

$$\begin{aligned} A &= [1,2], b = 1 \\ Aeq &= [2,1], beq = 1 \\ lb &= [], ub = [] \end{aligned}$$

Code:

```
fun = @(x)100*(x(2)-x(1)^2)^2 + (1-x(1))^2;
x0 = [0.5, 0];  % start point
A=[1,2];
b = 1;
Aeq = [2,1];
beq = 1;
[x,fval] = fmincon(fun,x0,A,b,Aeq,beq);
```

More examples: https://www.mathworks.com/help/optim/ug/fmincon.html

Drawback

- To apply these solvers, we need to transform a problem into an equivalent one that has a standard form. For example, we need to get the values of matrix/vector A, b, Aeq, beq, etc.
- For some problems without a standard form, we can apply some techniques (e.g. linearization technique) to turn it into a solvable form.
- For engineering problems, writing code to carry out this transformation is often painful.
- **Modeling systems** can partly automate this step.

Modeling systems

A modeling system can

- Automates most of the transformation to standard form; supports
  - ✓ Declaring optimization variables
  - ✓ Describing the objective function
  - ✓ Describing the constraints
  - ✓ Choosing the ***solver***

- Call the solver and returns the result (optimal, infeasible, …)

Modeling systems

YALMIP

- First matlab-based object-oriented modeling system with special support for convex optimization
- Can use different solvers, e.g. Cplex, Gurobi,etc; can handle some nonconvex problems

AMPL & GAMS

- Developed in 1980s, widely used in traditional OR

CVXPY/CVXMOD

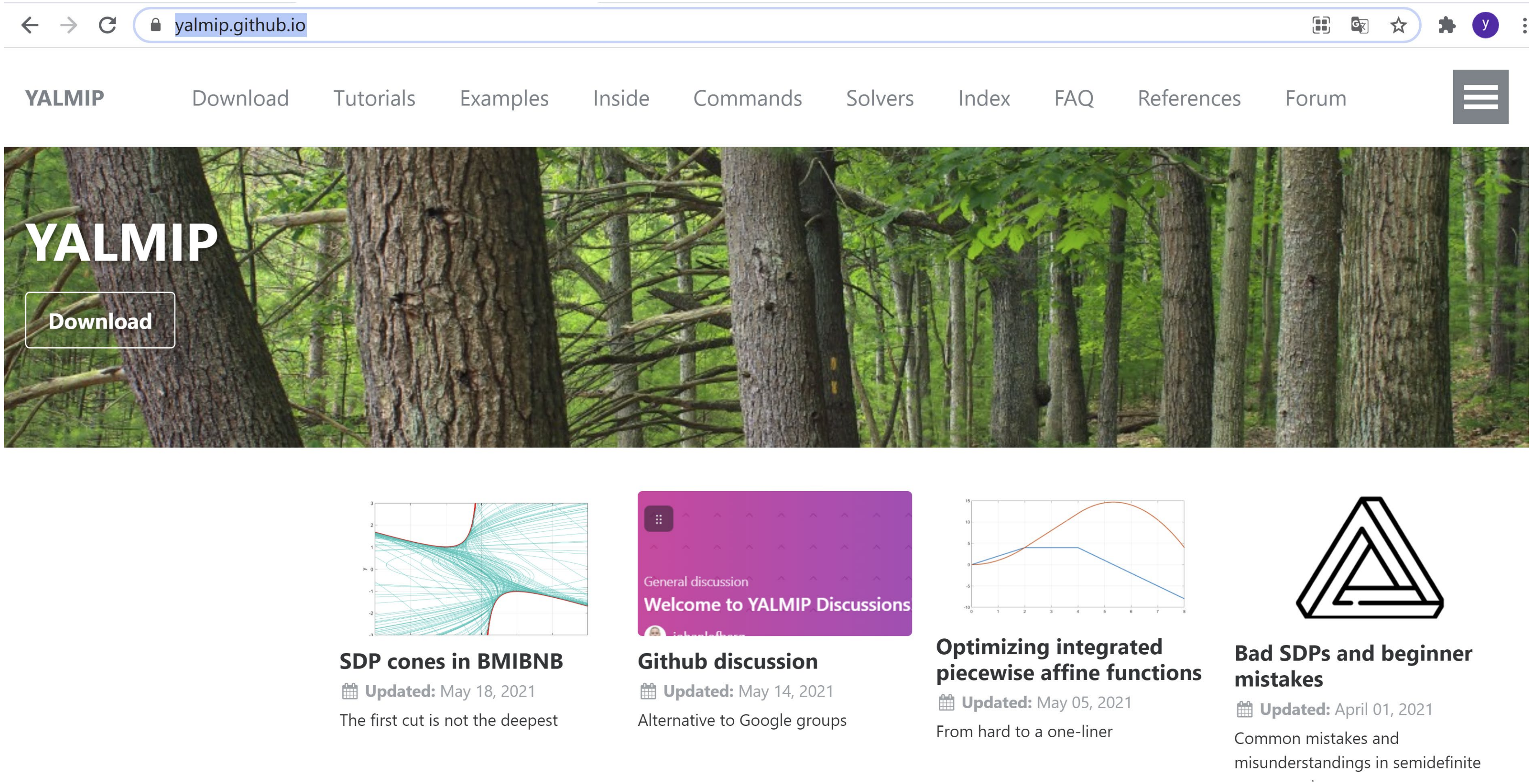
- Python based, has cone and custom solvers

CVX

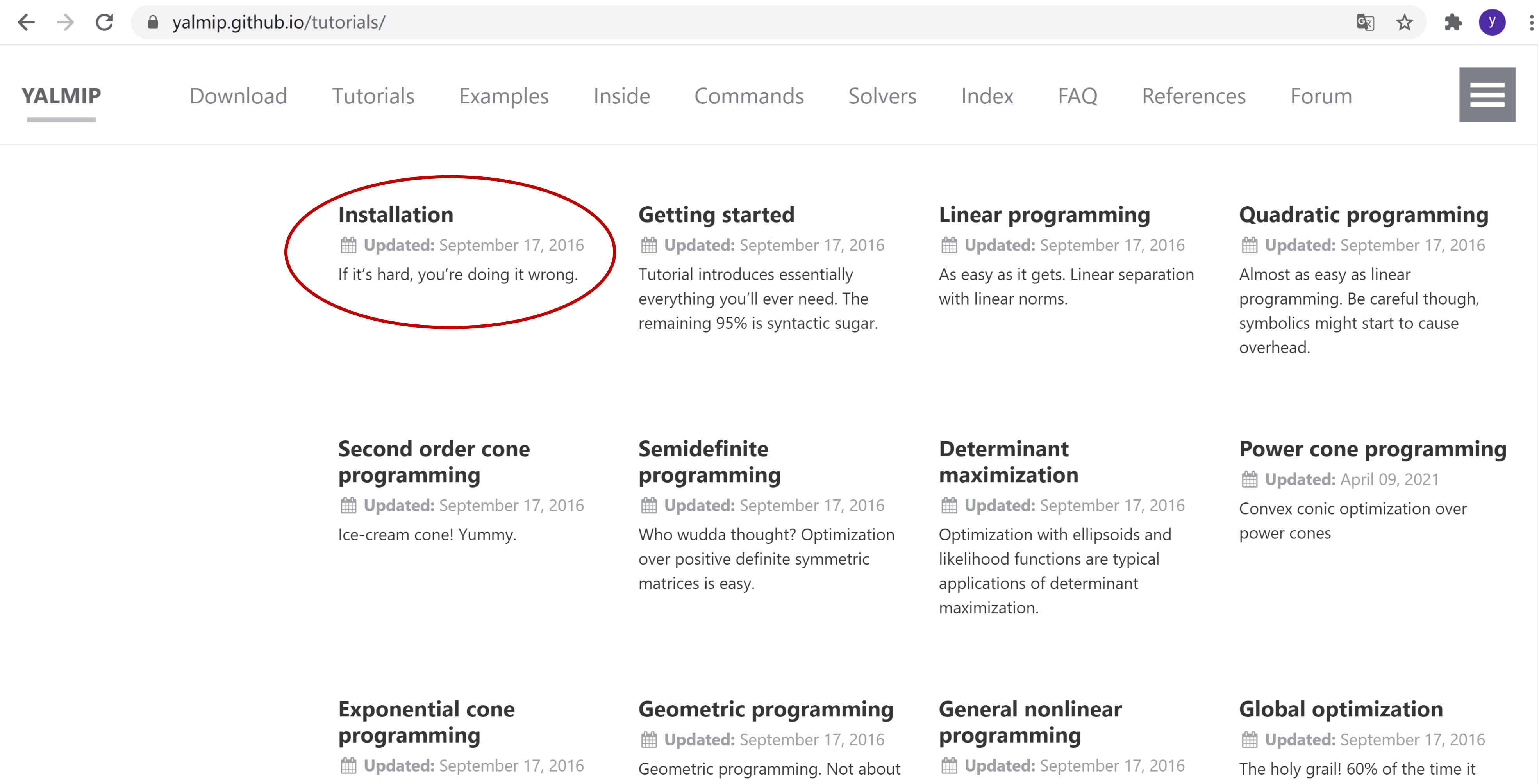
- Matlab based, uses SDPT3/SeDuMi

Yalmip

You can go to this website to download and install YALMIP for free.



Yalmip



yalmip.github.io/tutorials/
YALMIP
Download
Tutorials
Examples
Inside
Commands
Solvers
Index
FAQ
References
Forum
Installation
Updated: September 17, 2016
If it's hard, you're doing it wrong.
Getting started
Updated: September 17, 2016
Tutorial introduces essentially everything you'll ever need. The remaining 95% is syntactic sugar.
Linear programming
Updated: September 17, 2016
As easy as it gets. Linear separation with linear norms.
Quadratic programming
Updated: September 17, 2016
Almost as easy as linear programming. Be careful though, symbolics might start to cause overhead.
Second order cone programming
Updated: September 17, 2016
Ice-cream cone! Yummy.
Semidefinite programming
Updated: September 17, 2016
Who wudda thought? Optimization over positive definite symmetric matrices is easy.
Determinant maximization
Updated: September 17, 2016
Optimization with ellipsoids and likelihood functions are typical applications of determinant maximization.
Power cone programming
Updated: April 09, 2021
Convex conic optimization over power cones
Exponential cone programming
Updated: September 17, 2016
Geometric programming
Updated: September 17, 2016
Geometric programming. Not about
General nonlinear programming
Updated: September 17, 2016
Global optimization
Updated: September 17, 2016
The holy grail! 60% of the time it

Yalmip – variable declaration

First, we need to define the variables.

sdpvar is used to define YALMIPs symbolic decision variables.

Syntax

$x = [x_1, \dots, x_n]$

```
x = sdpvar(n)
x = sdpvar(n,m)
x = sdpvar(n,m,'type')
x = sdpvar(n,m,'type','field')
x = sdpvar(dim1,dim2,dim3,...,dimn,'type','field')
sdpvar x
```

$$x = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

Examples

A square real-valued **symmetric** matrix is obtained with

```
P = sdpvar(n,n) % SYMMETRIC!
```

A fully parameterized (i.e., not necessarily symmetric) square matrix

```
P = sdpvar(n,n,'full')
```

More information: https://yalmip.github.io/command/sdpvar/

Yalmip – variable declaration

binvar is used to define decision variables constrained to be binary (0 or 1).

intvar used to define decision variables with integer elements.

Syntax

```
x = binvar(n)
x = binvar(n,m,)
x = binvar(n,m,'type'
x = binvar(n,m,'type','field')
binvar x
```

Syntax

```
x = intvar(n)
x = intvar(n,m,)
x = intvar(n,m,'type'
x = intvar(n,m,'type','field')
intvar x
```

More information:
https://yalmip.github.io/command/binvar/
https://yalmip.github.io/command/intvar/

Yalmip – objective function & constraints

```
% Define variables
x = sdpvar(10,1);

% Define constraints
Constraints = [sum(x) <= 10, x(1) == 0, 0.5 <= x(2) <= 1.5];
for i = 1 : 7
  Constraints = [Constraints, x(i) + x(i+1) <= x(i+2) + x(i+3)];
end

% Define an objective
Objective = x'*x+norm(x,1);
```

Variable declaration

Constraints $\sum_{i=1}^{10} x_i \le 10, x_1 = 0, 0.5 \le x_2 \le 1.5$

Objective function

largest column sum of A, max(sum(abs(A)))

Yalmip – solve the optimization

optimize is the common function for solving optimization problems.

Syntax

Defined in previous slide

```
diagnostics = optimize(Constraints,Objective,options)
```

Examples

A linear program can be solved with the following piece of code

```
x = sdpvar(length(c),1);
F = [A*x<=b];
h = c'*x;
optimize(F,h);
solution = value(x);
```

Yalmip – solve the optimization

A diagnostic structure is returned which can be used, e.g, to check feasibility as reported by the solver (see yalmiperror for the possible return values)

```
diagnostics = optimize(F);
if diagnostics.problem == 0
 disp('Solver thinks it is feasible')
elseif diagnostics.problem == 1
 disp('Solver thinks it is infeasible')
else
 disp('Something else happened')
end
```

Yalmip – solve the optimization

sdpsettings is used to communicate options to YALMIP and solvers. It is used as the third argument in commands such optimize, optimizer, solvesos, solvemoment and solvemp.

Syntax

```
options = sdpsettings('field',value,'field',value,...)
optimize(Constraints, Objective, options)
```

For example

Select solvers, can change it to ‘cplex’, ‘gurobi’, etc.
Need to install the solvers first

```
ops = sdpsettings('solver','sedumi','sedumi.eps',1e-12);
```

More information: https://yalmip.github.io/command/sdpsettings/

Yalmip – output some useful information

Consider a Lyapunov stability problem

```
A = randn(5,5);A = -A*A';
P = sdpvar(5,5);
F = [A'*P+P*A <= 0, P >= eye(5)];
obj = trace(P);
```

Variable declaration

Constraints

Objective

Exporting this to a model in sedumi format is done by specifying the solver and calling export in the same way as optimize would have been called.

```
[model,recoverymodel] = export(F,obj,sdpsettings('solver','sedumi'));
model =
       A: [50x15 double]
       b: [15x1 double]
       C: [50x1 double]
       K: [1x1 struct]
    pars: [1x1 struct]
```

Command “model.A” can output the corresponding matrix

Yalmip – output some useful information

Syntax

```
[KKTsystem, details] = kkt(Constraint,Objective,z)
```

Comments

The command derives the KKT system for a linear or quadratic program parametrized in the variable **z**. The second output contains information about the analyzed problem, primal and dual variables, and possibly derived bounds on primal and dual variables.

The KKT system will contain a complementarity constraint which can be addressed by YALMIP using either integer programming or global nonlinear programming. Both methods require bounds on the dual variables. YALMIP tries to derive these bounds by default and add them to the KKT system. If this is unsuccessful (see **details.dualbounds**) you must manually add reasonable bounds on the variable **details.duals**)

Yalmip – output some useful information

Example

The following example derives the KKT conditions of a linear program in the decision variable **x**, with a cost depending on a parameter **z**. In this case, kkt successfully derives upper bounds the dual variables.

```
% min c(z)'*x s.t Ax<=b
A = randn(6,2);
b = rand(6,1);
c = rand(2,1);

x = sdpvar(2,1);
z = sdpvar(1);
c = c + randn(2,1)*z;

[Constraints,details] = kkt([A*x <= b, -1 <= z <= 1],c'*x,z);
```

parameter

CVXPY – Python based modeling system

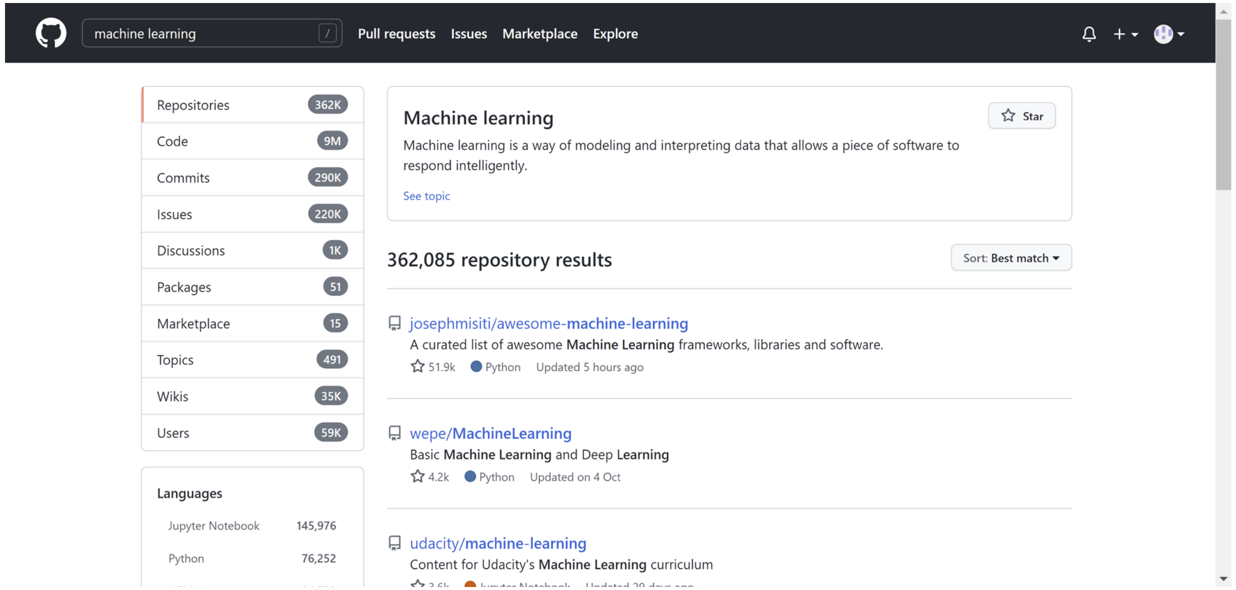
```python
import cvxpy as cp
import numpy as np

# Generate a random non-trivial linear program.
m = 15
n = 10
np.random.seed(1)
s0 = np.random.randn(m)
lamb0 = np.maximum(-s0, 0)
s0 = np.maximum(s0, 0)
x0 = np.random.randn(n)
A = np.random.randn(m, n)
b = A @ x0 + s0
c = -A.T @ lamb0
# Define and solve the CVXPY problem.
x = cp.Variable(n)
prob = cp.Problem(cp.Minimize(c.T@x),
                  [A @ x <= b])
prob.solve()
# Print result.
print("\nThe optimal value is", prob.value)
print("A solution x is")
print(x.value)
print("A dual solution is")
print(prob.constraints[0].dual_value)
```

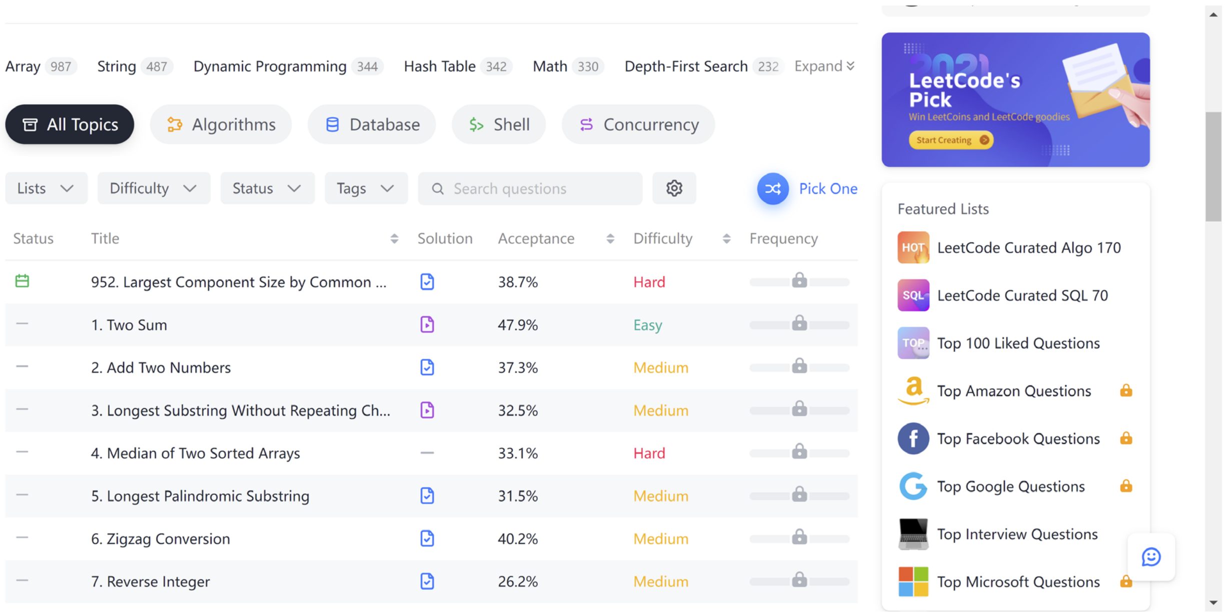
Define variable and solve the problem

More information: https://www.cvxpy.org/index.html

Some useful websites



Github



LeetCode

Thanks!