EPFL | MGT-418 : Convex Optimization | Software Setup

Instructions – Fall 2021

We will implement the computational exercises using CVXPY (a Python embedded modelling language for convex optimization problems) and solve them with the solvers GUROBI and MOSEK. The types of optimization problems that can be handled with these solvers are listed in the lecture slides. This document explains how to install the solvers and gives some basic examples of how to work with CVXPY.

Installation Guide

- CVXPY supports Python 3 on Linux, macOS, and Windows. You can use pip or conda for the installation. You may want to isolate your installation in a virtualenv, or a conda environment. To install CVXPY see https://www.cvxpy.org/install/.
- To obtain a free academic license, see http://www.gurobi.com/academia/for-universities. Carefully follow the instructions to activate your license. To install GUROBI for Python follow the instructions in https://www.gurobi.com/documentation/9.1/quickstart_mac/cs_python_installation_opt.html. Install GUROBI version 7.5.2 or newer such that you can `import gurobipy` in Python.
- To install MOSEK, see https://www.mosek.com/downloads, and to get a free academic license, see https://www.mosek.com/products/academic-licenses. Carefully follow the instructions to activate your license. You can verify that the installation was successful by `import mosek` in Python.

Introduction to CVXPY

- To import CVXPY we write `import cvxpy as cp`.
- Decision variables in CVXPY are declared by the command `cp.Variable`. To designate a vector $\boldsymbol{x} \in \mathbb{R}^n$ and a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ as decision variables, respectively, we write

  ```
  x = cp.Variable(n)    X = cp.Variable((n, m))
  ```

  In CVXPY, to create matrix that is constrained to be symmetric, we write

  ```
  X = cp.Variable((n, n), symmetric=True)
  ```

- To specify linear constraints of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{d}$, we write

  ```
  constraints = [A @ x == b, C @ x <= d]
  ```

  The parameter matrices and vectors can be NumPy ndarrays or NumPy matrices.
- To specify quadratic constraints of the form $\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} + \boldsymbol{r} \leq \boldsymbol{0}$, we write

  ```
  constraints = [cp.quad_form(x, P) + q.T @ x + r <= 0]
  ```

- To specify SOCP constraints of the form $\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\|_2 \leq \boldsymbol{c}^\top \boldsymbol{x} + d$, we write

  ```
  constraints = [cp.SOC(c.T @ x + d, A @ x + b)]
  ```

- To specify SDP constraints of the form $\boldsymbol{X} \succeq 0$, we write

  ```
  X = cp.Variable((n, n), PSD=True)
  ```

- We can specify the objective function $\boldsymbol{c}^\top \boldsymbol{x}$ as

  ```
  objective = c.T @ x
  ```

- To call the solver MOSEK and solve the optimization problem (*i.e.*, minimizing the objective subject to constraints), we write

  ```
  prob = cp.Problem(cp.Minimize(objective), constraints)
  prob.solve(solver=cp.MOSEK, verbose=True)
  ```

- For further information on CVXPY, please refer to https://www.cvxpy.org/tutorial/index.html#.