Benchmark for compilation high-dimensional convex problems via cvxpy

Disclaimer: Github's markdown may not render properly LaTeX equations of this readme, so, please, take a look at its `.pdf` version. Otherwise, we advise you to download the `.md` file and render it locally on your machine.

Here you find code analysing an impact of a dimension of a conic optimisation problem on compilation time via the cvxpy library. The compilation time is spent on a transformation of an original problem into its canonic form that is required for the majority of modern conic solvers. This benchmark shows that the compilation time drastically increases for high-dimension conic optimisation problems (more than 2000 variables). The main goal of the presented here script is to provide the cvxpy community with a stress test example that could help to accelerate the compilation of high-dimension conic optimisation problems.

In brief

The vectorized conic optimisation problem is formulated in the following way

$$\min_{\mathbf{\Sigma},\mathbf{P}} F(\mathbf{\Sigma},\mathbf{P}),$$
$$f(\mathbf{\Sigma},\mathbf{P}) \leq 0,$$

where F is an objective function, the vectors $\mathbf{\Sigma}$ and $\mathbf{P} \geq 0$ have the size of 4N and N elements respectively and represent main variables of the problem, where N is a certain integer parameter, on which a dimension of the problem is dependent. The elementwise function f represents a conic semi-definitive constrain that takes 4 components of the vector $\mathbf{\Sigma}$ and 1 component of the vector $\mathbf{P}$. Let us denote, for example, the first 4 components of the vector $\mathbf{\Sigma}$ and the first component of the vector $\mathbf{P}$ as $\boldsymbol{\sigma}$ and p respectively. The expression of the function f is following

$$f(\boldsymbol{\sigma}, p) = \sqrt{\frac{3}{2}\mathbf{s}^T\mathbf{s}} - \sigma_0 - pH,$$

where $\mathbf{s} = \mathbf{DEV} \cdot \boldsymbol{\sigma}$ is a deviatoric part of the variable $\boldsymbol{\sigma}$, the 4x4 matrix $\mathbf{DEV}$ and scalars σ_0 and H are certain constants.

The explicit expression of the objective function F is written in this way

$$F(\mathbf{\Sigma},\mathbf{P}) = \frac{1}{2}(\mathbf{\Sigma}^{\text{elas}}_{n+1} - \mathbf{\Sigma})^T\mathbb{S}(\mathbf{\Sigma}^{\text{elas}}_{n+1} - \mathbf{\Sigma}) + \frac{1}{2}H(\mathbf{P}^{\text{elas}}_{n+1} - \mathbf{P})^T(\mathbf{P}^{\text{elas}}_{n+1} - \mathbf{P}),$$

where $\mathbf{\Sigma}^{\text{elas}}_{n+1} = \mathbf{\Sigma}_n + \mathbf{C} \cdot d\boldsymbol{\varepsilon}$ and $\mathbf{P}^{\text{elas}}_{n+1} = \mathbf{P}_n$ are problem parameters, $\mathbf{C}$ is a 4x4 constant matrix and $d\boldsymbol{\varepsilon}$ is a vector of parameters, values of which are provided

via the standard binary file format in NumPy (.npy), the matrix $\mathbb{S}$ of the size 4N×4N is the block-diagonal matrix with $\mathbf{C}^{-1}$ on its diagonal. In this particular case, it is stated that $\mathbf{P}_n$ and $\mathbf{\Sigma}_n$ are identical to zero.

The parameter vector $d\boldsymbol{\varepsilon}$ has the same size as the variable $\mathbf{\Sigma}$, i.e. 4N elements. If we choose a certain value of N, we can state that the problem has 5N variables($|\mathbf{\Sigma}| = 4N + |\mathbf{P}| = N$), N constraints and 9N parameters ($|d\boldsymbol{\varepsilon}| = 4N + |\mathbf{\Sigma}_n| = 4N + |\mathbf{P}_n| = N$). Changing the value of N allows us to easily analyse how the compilation via cvxpy depends on the dimension of the problem. At the moment, there is provided only one vector $d\boldsymbol{\varepsilon}$ of the size 17736, so the maximal value of N is equal to 4434.

Notwithstanding the above, it is still possible to simplify the initialisation of the parameter vector $d\boldsymbol{\varepsilon}$. As the benchmark is focused on compilation time only, there is no interest in solving the original problem. Consequently, we can define the vectorized convex problem through a solution of N identical scalar convex problems. In this case, it is required, for example, only 4 first values of the vector $d\boldsymbol{\varepsilon}$ in order to initialise all parameters of the vectorized problem.

More context

The benchmark is based on the idea of solving an equilibrium problem of an elastoplastic solid using the convex optimisation theory, the so-called convex plasticity approach. The whole repository aims at providing an effective implementation of such an idea using the FEniCSx environment and the cvxpy library. It is not necessary to be familiar with the plasticity theory of solid mechanics to understand the benchmark. Here only the formulation of the conic optimisation problem will be described and implemented. In order to better understand the convex plasticity approach and the connection between convex optimisation and plasticity theories, we advise you to look at the report.

The original scalar conic optimisation problem can be formulated as follows:

$$\min_{\boldsymbol{\sigma},p} F(\boldsymbol{\sigma}, p),$$
$$f(\boldsymbol{\sigma}, p) \leq 0,$$

where F is an objective function, $\boldsymbol{\sigma}$ and $p \geq 0$ represent respectively a four-dimensional stress vector and a scalar variable associated with cumulative plastic strain, the inequality $f(\boldsymbol{\sigma}, p) \leq 0$ represents a conic semi-definitive constrain or, in terms of plasticity theory, a yield criterion.

Many of the yield criteria represent second-order (e.g. von Mises and Drucker-Prager yield criteria) or semidefinite (e.g. Rankine yield criterion) cone manifolds in the $(\boldsymbol{\sigma}, p)$ space. This allows us to consider the previous problem as a conic optimisation one. The von Mises yield criterion is used in this benchmark, for which the expression of the function f is following

$$f(\boldsymbol{\sigma}, p) = \sqrt{\frac{3}{2}\mathbf{s}^T\mathbf{s}} - \sigma_0 - pH,$$

where $\mathbf{s} = \mathbf{DEV} \cdot \boldsymbol{\sigma}$ is a deviatoric part of the variable $\boldsymbol{\sigma}$, the 4x4 matrix $\mathbf{DEV}$, the uniaxial strength σ_0 and the isotropic hardening modulus H are certain constants.

The explicit expression of the objective function F is written in this way

$$F(\boldsymbol{\sigma}, p) = \frac{1}{2}(\boldsymbol{\sigma}_{n+1}^{\text{elas}} - \boldsymbol{\sigma})^T\mathbf{S}(\boldsymbol{\sigma}_{n+1}^{\text{elas}} - \boldsymbol{\sigma}) + \frac{1}{2}H((p_{n+1}^{\text{elas}})^2 - p^2),$$

where $\boldsymbol{\sigma}_{n+1}^{\text{elas}} = \boldsymbol{\sigma}_n + \mathbf{C} \cdot d\boldsymbol{\varepsilon}$ and $p_{n+1}^{\text{elas}} = p_n$ are parameters of the problem based on its values from previous iterations of an algorithm of the convex plasticity approach. In this particular example, it is stated that $\boldsymbol{\sigma}_n$ and p_n are identical to zero. The matrix $\mathbf{C}$ is a 4x4 stiffness matrix containing material constants and $d\boldsymbol{\varepsilon}$ is a 4-dimensional vector of parameters, values of which are provided as an input of the benchmark. The matrix $\mathbf{S} = \mathbf{C}^{-1}$ is an inverted stiffness matrix. The objective function F has a physical meaning: it represents free energy of an elastoplastic solid.

In this benchmark, the solution of the conic optimisation problem is trivial and can be written as follows:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_{n+1}^{\text{elas}},$$
$$p = p_{n+1}^{\text{elas}}.$$

Values of a numerical solution of the elastoplasticity problem solved by the finite element method are known in nodes of a finite element mesh. The described above conic optimisation problem has to be solved at these nodes. As all the scalar conic problems are independent, it is possible to collect them into one vectorized conic problem. The formulation of the vectorized conic optimisation problem was given in the "In brief" section of this tutorial.

In order to provide the community with the simplest example of solving a vectorized conic optimisation problem via the cvxpy library, direct FEM calculations are avoided. Instead, precalculated values of the parameter vector *deps* are provided in the form of a NumPy array. This data is used to initialize the parameters of the conic optimisation problem.

The benchmark implements only one solution of the conic optimisation problem, whereas the convex plasticity approach implies solving the original problem several times, for several iterations, with different values of parameters of the problem. At this stage, it is enough to analyse the canonicalization of the convex optimisation problem via cvxpy, as the compilation occurs in the very first iteration of the approach.