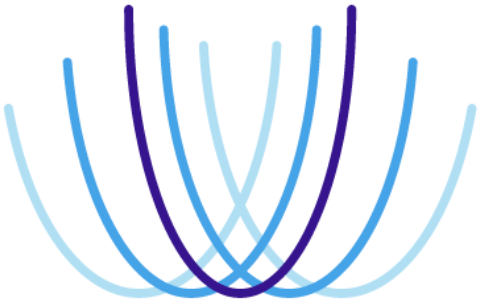
**Google Summer of Code - 2022**
**Organisation: CVXPY (NumFocus)**



CVXPY

**Personal Details:**


➔ Name: Parth Bansal
➔ Email: parthbansal1536@gmail.com (can also be used to iMessage)
➔ Mobile/Whatsapp: (+91)8127043111
➔ Github: parthb83
➔ LinkedIn: Parth Bansal
➔ Blog: parthb83.hashnode.dev
➔ Country: India
➔ Institute: Indian Institute of Technology (BHU) Varanasi
➔ Primary Language: English, Hindi
➔ Timezone: Indian Standard Time (IST) (UTC + 5:30)

## Project Title:

Improve CVXPY's performance benchmarks with continuous integration.

## About Me:

I am Parth Bansal, a Computer Science and Engineering undergraduate student at the Indian Institute of Technology ( BHU ) pursuing a Bachelor of Technology in my second year. I was introduced to the world of programming and software development in my first year. Since then, I have been very enthusiastic about deep diving into various fields of Computer Science. The areas that capture my interests are Data Structures, Algorithms, Operating Systems, Computer Vision and Data Science. For most of my programming journey, I have worked primarily with Web-based technologies, my niche and C/C++ programs, and I have recently been diving into Deep Learning as well.

## Project Abstract:

- A small benchmark suite already exists for CVXPY. The suite measures both the time to compile a problem into the solver standard form, and the time spent in the solver.
- Benchmarks are needed so that we can better understand how long it takes cvxpy to compile an issue. "Compile" here means converting user-provided input into the standard-form required by low-level solvers.The benchmark suite needs to be extended substantially, with a wider range of problems.
- Determining how well CVXPY scales to larger problems (>1e5 variables or constraints) and testing performance for conventional problem formats (e.g., quadratic algorithms).
- It should be simple to see how performance changes as a result of a PR or commit, as well as to follow those changes over time.

## Goals:

- Expand the benchmark suite that already exists in CVXPY to accurately measure changes in performance and to incorporate these measures across a wider range of problems with large constraints.
- Provide a benchmarking CI or an equivalent system to track performance changes across commits.
- Speed up the compilation time of CVXPY problems into the standard-form required by low-level solvers.

## Benefits To The Community:

If the above goals are met at the end of this project, it will greatly benefit the community. Some of them are:

- With the new benchmarking suites, it would be really easier for project maintainers to accurately measure the performance change after a new PR or a commit and thus let them know if they accidentally slows down some parts of CVXPY while trying to speed up canonicalization in other contexts. Therefore, this will be a great tool for the maintainers in checking the performance quality of a PR.
- Convex Optimisation or more particularly CVXPY is widely used in fields such as portfolio optimisation, worst-case risk analysis, optimal advertising, electricity generation optimisation etc. There are hedge funds that plan hundreds of millions of dollars of trades through CVXPY every day. The number one thing holding back CVXPY's wider adoption is its canonicalization speed, especially compared to **MathOptInterface** in Julia. The increase in performance and decrease in compilation time will lead to wider adoption of CVXPY in these fields which will greatly benefit the project maintainers and its community.
- The wide adoption of CVXPY in the above fields will let their engineers and software developers solve convex optimisation problems in Python language, thereby enjoying the vast and active community support that Python entails leading to a better progress in these fields.

Other than above points, there would be many more benefits for both the Python and the convex optimisation community. Since this project has many long-term benefits, I would like to work on this project.

## Technical Details:

- The main goal of this project is to provide CVXPY with a suitable and expansive benchmarking suite to better understand the time it takes for cvxpy to compile a problem. "Compile" here means transform the user-provided input into the standard-form required by low-level solvers. Problem compilation time depends heavily (almost *entirely*) on how the user specifies a problem. To reduce this compilation time, we should minimise the number of cvxpy Constraint and Variable objects necessary to specify the model. The work performed by cvxpy is to allow the user to provide natural input using these constructions, then transform that input for low level solvers with very rigid input requirements. The more complex these transformations are, the more the user benefits from specifying vectorized models.

- We can take a benchmark that's written with some high-level cvxpy code and automatically (or almost automatically) rewrite the problem in terms of high-level cvxpy code that's mathematically equivalent but that should result in different compilation times. Some examples of this are:
    1. Using Python's `sum()` vs `cp.sum()`
    2. writing `A @ x <= b` as a list of Constraints `[A[i] @ x <= b[i] for i in range(b.size)]`
    3. using `cp.scalar_product(X, Y)` versus `cp.trace(X @ Y)` for symmetric matrices X, Y.
- There may be some real world optimization problems, which when specified naturally are computationally really expensive for CVXPY to compile. To tackle this issue, we can draw test problems from classic operations research applications that deal with the development and application of advanced analytical methods to improve decision-making. It addresses real world problems such as critical path analysis, resource allocation, network optimisation etc. which rely heavily on convex optimisation.
- We can get a wide variety of test cases for benchmarking from **CBLIB- The Conic Benchmarking Library** which hosts a variety of benchmark problems for conic mixed-integer and continuous optimization. We need to convert those problem statements into a code that's understandable for CVXPY.
- We can also look through all the examples given in the Jupyter Notebook of CVXPY and identify the optimization problems that are defined in a complicated way and then write functions to construct arbitrarily large versions of those examples.
- I am also thinking of integrating **pyperformance, pyperf** or **airspeed velocity(asv)** tools with github actions to benchmark the library over a lifetime. Runtime, memory consumption and even custom-computed values may be tracked. These tools provide easy to use APIs for benchmarking a python code and may also provide results through an interactive web UI.
- We can also try to use github actions to trigger a third party workstation to run the benchmarks since benchmarks are computationally expensive and the machines we get free access to by github aren't that powerful.

## Timeline:

- **Community Bonding Period (May 20 - June 12):**

Learn more about Python Benchmarking and Github Actions during this period. Discuss the technical details given above in great detail with the mentors. Create a detailed roadmap while exchanging various ideas with the mentors on how to expand CVXPY's benchmarking suite. I will also try to make contacts with Miles Lubin of JuMP and ask for his advice on writing test cases for Conic Benchmarking. I can also give a small presentation to the mentors at the end of this period to show my preparedness for this project before the start of Coding Period.

**[Coding Period]**

- **Week-1 and Week-2 (June 13 - June 26):**

During this week I'll try to integrate the **pyperformance, pyperf** or **airspeed velocity (asv)** tool for benchmarking the library. This will help us in coming weeks as we can check the performance changes with the tests that will be written for benchmarking in upcoming weeks.

- **Week-3 (June 27 - July 3):**

This week will be used to use Github Actions to trigger a remote third-party custom workstation to run our benchmarks.

- **Week-4 and Week-5 (July 4 - July 17):**

During this period, I will try to code real world convex optimization testing cases from **Operations Research Applications** and determine which ones can be computationally expensive for CVXPY to compile.

- **Week-6 (July 18 - July 24)**

This week will be kept as a buffer to rectify any unpredictable delays that might occur in the previous weeks. I can also use this week to learn more about the **CBLIB.**

- **Phase-1 Evaluation (July 25 - July 29):**

This period will be used to discuss and review project progress with the mentors. Also create a project report on the development thus far.

- **Week-7 and Week - 8 (July 30 - August 12):**

This week will be used to translate the conic benchmarking test cases that are given in CBLIB into a form that is understandable for CVXPY. I can also write this test cases to use various low level solvers like **'ECOS'**, **'SCS'** etc.

- **Week-9 and Week-10 (August 13 - August 26):**

In this period, I will try to go through examples given in the Jupyter Notebook in the CVXPY repository and try to find the ones that are defined in a complicated way. I'll write a function to create arbitrarily large versions of these problems for benchmark testing.

- **Week-11 and Week-12 (August 27 - September 4):**

This will be a buffer period before the Phase-2 evaluations to update the relevant documentation changes and also to work upon some new testing ideas that may be given by the mentors. This week will also be kept to rectify any unnecessary delays in the project.

- **Phase-2 Evaluations (September 5 - September 12):**

Compile the final project report on the work done and discuss and analyse it with the mentors. Discuss the changes in the compilation speed of CVXPY and also the benchmarking reports with the maintainers.

## Why do I want to do this project?

- The CVXPY project is an excellent opportunity for me to deep dive into the world of convex optimization and Python benchmarking and better understand what happens under the hood of a compiler/interpreter that has always been a black box for me.
- The project has great mentors who are always willing to provide the best possible aid and are very responsive, friendly and ready to share their knowledge. This has been a fantastic experience for me. It allows me to interact with mentors and other contributors and gain more knowledge while staying connected with the community and contributing more to the project through discussion and code.
- Lastly, I'd like to state that I'm fascinated by the open-source community and it has played an instrumental role in helping develop my skills. This project will be my small way of giving back.

## Past Contributions:

The Pull Requests I made in CVXPY repository were:

| Pull Request | Status | Description |
|---|---|---|
| #1641 | Merged | Added a new atom function called **xexp()** which returns x*exp(x), where x is the value that is passed to the function. Closed issue #1517. |
| #1683 | Merged | Fixed the cummax bug, updated the cumsum test and added a new cummax test for the new canon change. Closed issue #1678. |

Some other Pull Requests in other Open Source Organisations that I made were:

| Pull Request | Organisation/Repository Name | Status |
|---|---|---|
| #2127 | GMLC-TDC/HELICS | Merged |
| #2128 | GMLC-TDC/HELICS | Merged |
| #2138 | GMLC-TDC/HELICS | Merged |
| #373 | lukew3/mathgenerator | Merged |
| #5387 | pymc-devs/pymc | Merged |
| #5415 | pymc-devs/pymc | Merged |

## Why am I suitable to work on this project?

- If you see my past contributions to other open source organisations, most of the Pull Requests made in them were associated with Github Actions, Workflows or Continuous Integration.
- In HELICS, my PRs were associated with making workflows for Mac, Windows and Ubuntu builds.
- In PyMC, both PRs were made to speed up the build time by installing dependencies using `mamba` instead of `miniconda`. Therefore I'm already familiar with Github Actions and how to make workflows which will certainly be useful for this project.

- I have also written several unit tests in Python while contributing to the CVXPY codebase. Any change in the codebase will be incorporated with a separate unit test to test that change extensively.

Other than these points, if my proposal is accepted, I'll give my utmost commitment to learn more about benchmarking in Python and to speed up CVXPY's compilation time to help the community.

## Time Commitment:

- My timezone is IST(UTC + 5:30) and will be same throughout my GSoC period. I am comfortable to work and meet according to any timezone that the mentors may prefer.
- I would be working throughout the GSoC period from 13th June to 13th November (20 week period).
- During the GSoC period during my summer vacations(June 13 to July 24), I'll commit approximately 5-6hrs per day and 40hrs per week. When my college reopens after vacation, I'll be able to give 3hrs per day and 20hrs per week till the GSoC period ends.
- I have no other job or internship commitments during this summer. If I'm selected for GSoC, it will be my only job this summer. If I complete this project early, I would be available for any voluntary work that CVXPY wants me to do.

## Projects:

- **StonX** | **Tech Stack Used:** Django, SQLite, Bootstrap, HTML, CSS

Created a stock market trading web application with real time stock price quotations and mock trading with virtual wallet. Shown latest news of stock world and real time quotations using APIs.

- **Exploratory Project** | **Tech Stack Used:** Python, Pytorch, NumPy

Currently working on a semester long exploration project on the topic of "**Blind Action Recognition**" in UCF50 dataset under Prof. Tanima Dutta. Implemented Slow-Fast Network by Facebook AI Research (FAIR) and Deep Spatio Temporal Manifold Network (STMN) for Action Recognition.