A Library of Examples for CVXPY

Judson Wilson

June 4, 2014

1 Introduction

CVXPY [DCB14] is a new modeling language for convex optimization that allows users to easily express and solve convex optimization problems using the Python programming language. The package is inspired by CVX [GB14, GB08], a similar project implemented for the MATLAB programming environment.

The primary goal of this project was to create a library of example programs for CVXPY. The author aims to provide a useful, aesthetically pleasing collection of examples which demonstrates features of CVXPY and methods of problem translation in the context of a wide range of applications. Also, the process of creating examples was used to test both the general functionality and usability of CVXPY in these application areas.

The example programs were developed in IPython Notebook [PG07], a web-based interactive interface to Python. This interface presents the user with an interactive document, composed of formatted text and mathematical expressions (in \LaTeX{} format), mixed with blocks of code, output text, and output plots. The CVXPY project maintainers can export customized HTML and PDF versions of the example notebooks, formatted to match the project website and documentation.

The examples were sourced from the CVX project website. They were originally written in the MATLAB language using comment blocks for documentation. The efforts of this project consisted of converting the code for use with CVXPY and reformatting the code and comment blocks using the added functionality of the IPython Notebook format.

2 Overview of challenges

Challenges first arose when installing CVXPY and its required dependencies. After several unsuccessful attempts, the author found a set of packages which included most of the dependencies in an easy-to-use installer. The remaining packages required modifications to the installer source files to install properly. The general procedural information and required source modifications were passed on to the authors of CVXPY and to the authors of the solvers used by CVXPY. These actions directly led to updates to the documentation and installer source code of the projects, which will improve useability for future first-time users.
of these packages. Once this step was completed, work began on the main task of porting CVX examples to CVXPY.

While an initial look at CVXPY might lead a user to believe that it is very similar to CVX, the differences quickly become apparent in practice. These differences are rooted in the fact that Python and MATLAB are two very different programming languages, and CVXPY was designed specifically to explore different approaches to the basic design goal it shares with CVX, rather than simply port CVX to a new language. Furthermore, CVX benefits from many more years of development and has a much wider range of atoms to choose from, giving the user more flexibility in problem description. Converting code from CVX to CVXPY is not as simple as looking up corresponding atoms. Examples of such differences will be demonstrated throughout this paper.

At the language level, MATLAB was built specifically to be a scientific computing language with simple data types and dedicated operators for both matrix and element-wise operations. Python is a general purpose scripting language, and scientific computing requires use of classes and methods from the SciPy [JOP+14] package to implement features that are built into the native syntax of MATLAB.

At a higher level, CVXPY does not yet have the vast array of features that CVX includes. As a simple example, consider a trace operator constraint of the form \( \text{Tr}(A) = 1 \), where \( A \in \mathbb{R}^{n \times n} \). In CVX this may be written as:

\[
\text{trace}(A) == 1
\]

whereas an equivalent constraint in CVXPY requires significantly more creativity:

\[
\text{sum}([A[i, i] \text{ for } i \text{ in range(n)]}) == 1
\]

This expression also bears less resemblance to the underlying mathematical expression. Porting-challenges such as these will be highlighted within the discussion of each example.

3 Content

The CVX website contains a large collection of examples from several topics. For this project, a diverse subset was chosen for conversion to CVXPY. The goals were to keep each example true to the original, for each example to function identically to the original (unless there was an obvious error, or a simple, obvious improvement could be made), and for every change made to be an improvement. Second to these objectives, the author attempted to convert as many examples as possible, from a wide range of topics, rather than devote too much time to a small number of examples.

The sections below are organized by example. Each section gives a short overview of the optimization problem, highlights any useful strategies or unique features demonstrated, and describes noteworthy difficulties encountered during conversion.
3.1 Antenna array design: Minimize beamwidth of an array with arbitrary 2-D geometry

This example illustrates a quasiconvex problem in the field of antenna array design. The design objective is to minimize beamwidth while constraining sensitivity within the beam and outside of the beam.

This example illustrates how to solve a quasiconvex problem using bisection. It also demonstrates an example of converting complex-valued mathematical expressions into a form accepted by CVXPY, which can be quite tedious. For instance, in CVX we can write the following single-line constraint:

\[
\text{abs}(A \times x) \leq 1
\]

where \(A\) is a complex constant and \(x\) is a complex variable. In CVXPY, the quantities must all be real. The actual equivalent expression in CVXPY is too long to give a proper treatment in this paper. Refer to the example itself for more details. Adding complex valued variables to CVXPY would greatly reduce the effort required to translate problems like these to a form that CVXPY will accept.

3.2 Circuit design: Sizing of clock meshes

This example illustrates a circuit design problem, where a clock mesh circuit is optimized to minimize power consumption under a time constant constraint. One specific benefit of this example is that it demonstrates how to implement semidefinite constraints.

The example also makes extensive use of the \texttt{reshape} function, which produces an output array with the same elements of the input array, but “reshaped” such that the array has new dimensions. Originally, no corresponding CVXPY atom existed for directly creating expressions which reshaped arrays, so the following code was used in the example to produce the same effect using loops:

```python
# Constrain matrix G to equal a reshaping
# of column vector GG*x
for j in range(n):
    constraints.append(
        G[:, j] == (GG*xt)[(j*n):(j+1)*n] )
```

With the newly added \texttt{reshape} atom in CVXPY, this code block was replaced with the following constraint:

\[
G == \text{cvx.reshape}(GG*xt,n,n)
\]

which is considerably easier to formulate and comprehend.

The \texttt{reshape} atom was specifically added in response to this example, which demonstrates why this project is a valuable test of user experience and application.
3.3 Filter design: Chebychev design of an FIR filter given a desired $H(\omega)$

In this filter design example, FIR filter coefficients are determined to achieve a minimum absolute difference between a target frequency response, $H(\omega)$, and actual frequency response. This is a simple approach to a real-world problem that traditionally took much more experience to solve.

This problem, like the antenna array example in §3.1, was formulated with complex-valued variables and constants, and required conversion to an equivalent real-valued form.

3.4 Entropy maximization

This short example demonstrates the use of the entropy atom, $\text{entr}$, in a linear inequality constrained entropy maximization problem. Highlighting the existence of lesser known convex (or concave, in this case) atoms such as these may inspire new use cases.

3.5 Sparse covariance estimation for Gaussian variables

This example demonstrates a method to obtain a sparse estimate of the inverse covariance matrix of a Gaussian random vector using samples from its distribution. The example shows how to formulate an SDP in CVXPY, and how to implement an expression using the trace of a matrix, as discussed in §2.

While converting the example, a small number of critical yet easily overlooked mistakes were identified in the original example from CVX. These errors were reported to the CVX project maintainers, and were corrected in the CVXPY version.

3.6 Computing a sparse solution of a set of linear inequalities

This example illustrates the use of an $\ell_1$-norm heuristic method to find a sparse vector solution to a system of inequalities, as well as a similar method which employs iterative weighting of the vector elements. The implementation is rather simple, but the example is a useful tool to demonstrate these valuable heuristic methods.

3.7 $\ell_1$ trend filtering

This example demonstrates how to create piecewise-linear trend estimates of time series data using an $\ell_1$ filtering method. The implementation is straightforward, except that one sparse format employed in the example must be converted for compatibility with CVXPY. This example demonstrates an efficient sparse matrix format conversion which can be used to ensure that CVXPY can leverage the sparsity.

Also, while porting this example, inconsistencies were found in CVXPY in the way convergence failures were handled using different solvers. This issue was reported to the CVXPY maintainers. This demonstrates further testing value added by this project.
3.8 Nonnegative matrix factorization

This example demonstrates an approximate method for solving a nonconvex problem using an
alternating iterations method. In this method, at each iteration one variable is held constant,
and in the next iteration a different variable is held constant. The example illustrates the
type conversions needed using such a method, unlike the CVX implementation where this
automatic.

4 Ongoing and future work

The author plans to convert a number of additional examples. Also, in the process of
converting examples and writing this paper, it has become clear that there is value in writing
documentation tailored specifically for users who already have experience with MATLAB and
CVX. This documentation would detail the issues mentioned in this paper. The author may
pursue this obvious extension of the project.

5 Summary and conclusions

A diverse set of examples has been ported from the CVX project to the CVXPY project.
These examples not only demonstrate the power of CVXPY to users, but also can be used
to suggest strategies of problem formation.

Several potential improvements of CVXPY were identified. For instance, the lack of sup-
port for complex-valued variables in expressions adds significant difficulty and opportunity
for error when formulating certain problems in CVXPY. Such problems often appear in ap-
plications where complex-valued mathematics are fundamental to the field, such as Electrical
Engineering. Also, atoms such as \texttt{trace} that do not yet exist in CVXPY were shown to
be a potentially useful addition to the project. One such atom, \texttt{reshape}, has already been
implemented by the CVXPY authors as a result of this work.

The project resulted in other benefits as well. For instance, issues and potential remedies
were identified with the CVXPY installation process, and documentation and software up-
dates were produced as a result. Errors were identified in at least one example in the CVX
project, and remedies were suggested to the CVX project administrators.

Overall, this project is a significant net contribution to CVXPY and other software used
for convex optimization.

Acknowledgments

The author would like to give thanks to Steven Diamond for suggesting the project and pro-
viding technical support, to Professor Stephen Boyd and the Teaching Assistants of EE364b,
and to all who contribute to the free software used for the project including CVXPY, CVX,
and Python.
References


