Convex optimisation in communications with cvxpy

Robert P. Gowers,[1] Sami C. Al-Izzi,[1] Timothy M. Pollington,[1] Roger J. W. Hill,[1] and Keith Briggs[2]
[1]*Department of Mathematics, University of Warwick*
[2]*BT, Adastral Park*

"Nothing takes place in the world whose meaning is not that of some maximum or minimum."

LEONARD EULER (1707-1783)

INTRODUCTION

Convexity is an important property in optimisation. This is because if a problem is convex then the task of finding a global minimum is reduced to that of finding a local minimum. The importance of finding these minima efficiently in science and engineering has driven the development of software packages, such as `cvxpy`.

Here we show that an interesting problem in communications not only has a convex formulation (as shown in [1]), but can be easily implemented computationally using the `cvxpy` Python module. The ease of implementation makes `cvxpy` an invaluable tool for both students and researchers in many areas of science and engineering.

Convex Functions

A function f, where $f : \mathbb{R}^n \to \mathbb{R}$, is *convex* if $\forall x, y \in$ $\mathbf{dom} f$ and $0 \leq \theta \leq 1$ [1, p.67]:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y). \quad (1)$$

This means that the function is less than or equal to linear, as shown in Fig 1.

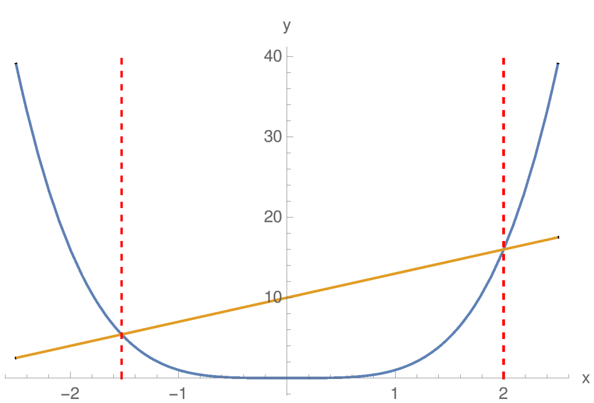

Figure 1. A chord showing convexity of the function x^4.

Convex optimisation

A *convex optimisation problem* has three components:

- a convex *objective function* $f_0(x)$,
- m convex *inequality constraint functions* $f_i(x)$,
- k convex *equality constraint functions* $g_j(x)$,

where $f, g : \mathbb{R}^n \to \mathbb{R}$ [1, p.141]. We seek an optimum, $x^* \in \mathbb{R}^n$, where $f_0(x^*)$ is a minimum. Formally the problem is defined as:

$$\begin{array}{lll} \text{minimise} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & i \in \{1, .., m\} \\ & g_j(x) = 0, & j \in \{1, ..., k\}. \end{array} \quad (2)$$

Any local optimum x^* for a convex optimisation problem is also a global optimum [1, pp.138-139], however an optimum may not be unique.

Disciplined convex programming

`cvxpy` is a symbolic programming module for *Python* developed at Stanford[3]. It uses a set of rules, called *disciplined convex programming* (DCP), to determine whether a function is convex. This is implemented using predefined classes containing functions with their curvature and sign, and using general composition theorems from convex analysis. If this is the case, then interior point methods guarantee an optimal solution.

EXAMPLE: POWER MINIMISATION IN COMMUNICATIONS

For a real world example consider a system of n transmitters and m receivers distributed in $2D$ Euclidian space. Each of the transmitters has power p_i. How can we minimise the total power consumption P of transmitters, yet achieve a minimum Signal to Interference plus Noise Ratio (SINR), γ_0, for all receivers? This question is relevant to a telecoms company who want to offer a service at a minimum quality standard. We formulate the problem with a given square path gain matrix G, background noise level σ and maximum power value that any transmitter can reach $P_{\max}$:

$$\begin{aligned} \underset{p}{\text{minimise}} \quad & \sum_j p_j \\ \text{subject to} \quad & p_j \leq P_{\max} \\ & p_j \geq 0 \\ & \gamma_i \geq \gamma_0 \end{aligned}$$

where symbols $\succ$, $\succeq$ denote element-wise inequalities between vectors or matrices.

The desired signal to receiver i is denoted by $S_i = G_{ik}p_k$, while the interference to i is $I_i = \sum_{j \neq k} G_{ij}p_j$. However DCP does not allow division of the optimisation variable p_i so

$$\gamma_i \geq \gamma_0 \iff \frac{S_i}{\sigma_i + I_i} \geq \gamma_0, \quad \forall i$$

was rearranged to:

$$S_i - \gamma_0(\sigma_i + I_i) \geq 0, \quad \forall i$$

which is affine, and hence DCP.

Path gain

The path gain G_{ij} represents the proportion of power that reaches receiver i from transmitter j. Supposing that the desired signal to receiver i is from transmitter k, the power received is,

$$p_{ik}^{\text{rec}} = G_{ik}p_k. \tag{3}$$

The signal-to-interference-plus-noise ratio (SINR) is the strength of the desired signal S_i relative to the interference power I_i plus the background noise σ_i at i. Denoting the SINR at i by γ_i, we have

$$\gamma_i = \frac{G_{ik}p_k}{\sigma_i + \sum_{j \neq k} G_{ij}p_j} = \frac{S_i}{\sigma_i + I_i}. \tag{4}$$

In a physical context, the path gain from transmitter j to receiver i will depend on the distance, d_{ij} between them. Assuming isotropic propagation from each transmitter, the fraction of power from j that reaches i is given by,

$$G_{ij} = k_i / d_{ij}^{\alpha} \tag{5}$$

where α is the pathloss coefficient, k is a proportionality constant specific to the receiver i. For free space $\alpha = 2$, while for an urban environment $\alpha \sim 3.5$ **CITE THIS**. Further physical complexities, such as the stochastic effects from Rayleigh fading can be easily implemented.

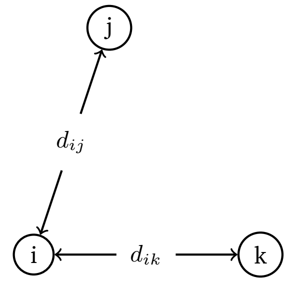

Figure 2. Path lengths between receiver i and transmitters j and k.

Example code

To indicate the brevity of code implementation in cvxpy we show an abridged version below:

```
# Declare variables
I = np.zeros((n,m)) # interference power matrix
S = np.zeros((n,m)) # signal power matrix
delta = np.identity(n)
S = G * delta # using gains matrix G
I = G - S

# Declare optimisation variable
p = cvx.Variable(n)

# Define objective function
obj = cvx.Minimize(
      cvx.sum_entries(p))

# Declare constraints
constraints =     [p >= 0,
                   S*p-alpha*(I*p+sigma)>=0,
                   p <= Pmax]

# Solve problem
prob = cvx.Problem(obj, constraints)
prob.solve()
```

OUTLOOK

The authors hope that this article shows the ease with the which cvxpy can be brought to bare on simple “toy” problems of real world relevance. We also believe that this is an invaluable resource for researchers wanting test the applicability of convex optimization in new fields.

The module could also be easily incorporated into a course on optimization, providing students with a simple environment with which to implement techniques for convex optimization.

Acknowledgements

We would like to thank S. Johnson and S. Diamond for helpful discussions.

APPENDIX

Full code can be found at https://github.com/cvxgrp/cvxpy/tree/master /examples/communications.

[1] Steven Boyd and L Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[2] S. Diamond. Stanford dcp analyzer. `http://dcp.stanford.edu/analyzer`, 2013.

[3] S Diamond and S Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 2016.

[4] Claude E. Shannon and Warren Weaver. *The mathematical theory of communication*. University of Illinois Press, 1949.