Communicating with convexity

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 has a convex formulation that can be easily implemented computationally using the `cvxpy` *Python* module. This demonstrates `cvxpy` as an invaluable tool for both students and researchers in many areas of science and engineering. First let us introduce what convexity means.

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.

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.

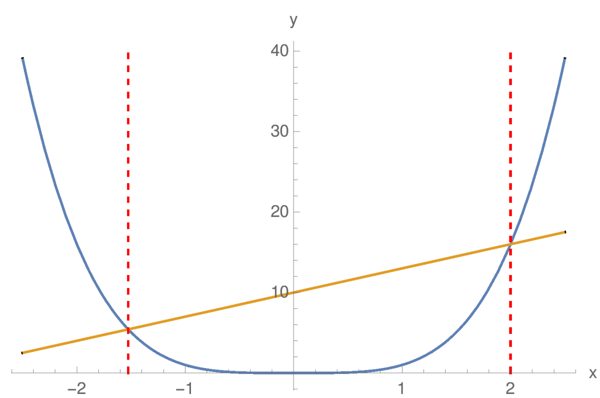

Figure 1. A chord showing convexity of the function x^4.

Disciplined convex programming

In order to solve convex optimisation problems, we used `cvxpy`, a symbolic programming module for *Python*[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 [2]. If DCP rules can be applied then interior point methods guarantee an optimal solution. Interior point methods work by applying Newtons algorithm to sequences of equally constrained problems. The algorithm works by picking a step and direction to travel in the interior of the feasible convex set, such that the solution progresses towards optimality, for more details on barrier methods and primal-dual methods see [1, p.561].

---

* r.gowers@warwick.ac.uk
† s.al-izzi@warwick.ac.uk
‡ t.pollington@warwick.ac.uk
§ r.hill.3@warwick.ac.uk
¶ keith.briggs@bt.com

EXAMPLE: POWER MINIMISATION IN COMMUNICATIONS

To demonstrate the applicability of cvxpy to a real-world example, we consider a system of n transmitters each of power p_i and m receivers, all in 2D Euclidean space[5]. In this problem we are constrained to having a minimum signal-to-interference-plus-noise ratio (SINR) at each receiver; the strength of the desired signal, S, relative to the interference power, I, plus the background noise, σ, at a receiver. How can we minimise the total power consumption, P, of transmitters, yet achieve this minimum SINR, γ_0, for all receivers? This question is relevant to telecoms companies 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 a maximum power constraint $P_{\max}$ of each transmitter:

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

The desired signal for receiver i is $S_i = G_{ik} p_k$, while the interference at 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 a DCP function.

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,

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

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, as shown in Fig. 2, 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:

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

where α is the pathloss coefficient and k is a proportionality constant specific to the receiver i. For free space $\alpha = 2$, while in urban environments $\alpha \sim 3.5$ [4]. Further physical complexities, such as the stochastic effects from Rayleigh fading can be easily incorporated.

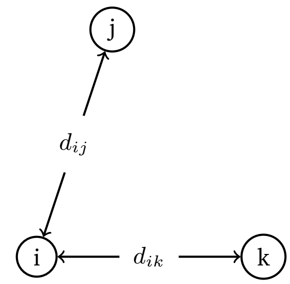

Figure 2. Path lengths between receiver i and transmitters j and k.

Example code

To indicate the brevity of code implementation in cvxpy we show a minimal working example below:

```
import cvxpy as cvx
import numpy as np

# Define variables
n = 3
k = 1.0
d = np.random.rand(n,n)
d_s = 0.5 * (d + d.T)

G = np.zeros((n,n))
G = k / d_s ** 3.5

# δ = np.identity(n)
Îť = np.identity(n)
# S = G * δ
S = G * Îť
I = G - S

# σ = 0.1 * np.ones(n)
ÏČ = 0.1 * np.ones(n)
# γ = 1.0
Îş = 1.0
Pmax = 1.0

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

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

# Define constraints
constraints = [p >= 0,
               # S * p - γ * (I * p + σ) >= 0,
               S * p - Îş * (I * p + ÏČ) >= 0,
               p <= Pmax]

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

```python
print('Solution status = {0}'.format(prob.status))
print('Optimal solution = {0:.3f}'.format(prob.
    value))
if prob.status == 'optimal':
  for j in range(n):
    print('Power {0} = {1}'.format(j,p.value[j]))
```

OUTLOOK

The authors hope that this article shows the utility of cvxpy for simple 'toy' problems of real-world relevance. We believe it is an invaluable resource for researchers wanting to test the applicability of convex optimisation in new fields.

The module could also be easily incorporated into a course on optimisation, providing students with an accessible environment to practise techniques for convex optimisation. Full code for other communication examples can be found at https://github.com/cvxgrp/cvxpy/tree/master/examples/communications.

Acknowledgements

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

Author Biography

Robert Gowers, Sami Al-Izzi, Timothy Pollington and Roger Hill are PhD students in the Mathematics for Real-World Systems CDT at the University of Warwick. Keith Briggs is a senior research mathematician at BT.

[1] Steven Boyd and L Vandenberghe. Convex Optimization. `http://stanford.edu/~boyd/cvxbook`, 2004.

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

[3] S Diamond and S Boyd. CVXPY: A Python-embedded modeling language for convex optimization. `www.cvxpy.org`.

[4] Masaharu Hata. Empirical formula for propagation loss in land mobile radio services. *Vehicular Technology, IEEE Transactions on*, 29(3):317–325, 1980.

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