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

If a set C is *convex*, then for any two points $x, y \in C$, any point z along the x, y line must also $\in C$ [1, p.23]. More formally, for $x, y \in C$ and $\theta \in [0, 1]$:

$$\theta x + (1 - \theta)y \in C. \tag{1}$$

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

$$f(\theta x + (1 - \theta)y) \le \theta f(x) + (1 - \theta)f(y). \tag{2}$$

This means that the function is less than or equal to linear, as shown in Fig. 1. A concave function can be made convex by the operation $f \to -f$.

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:

$$\begin{array}{lll} \text{minimise} & f_0(x) & \\ \text{subject to} & f_i(x) \le 0, & i \in \{1, .., m\} \\ & g_j(x) = 0, & j \in \{1, ..., k\}. \end{array} \tag{3}$$

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.

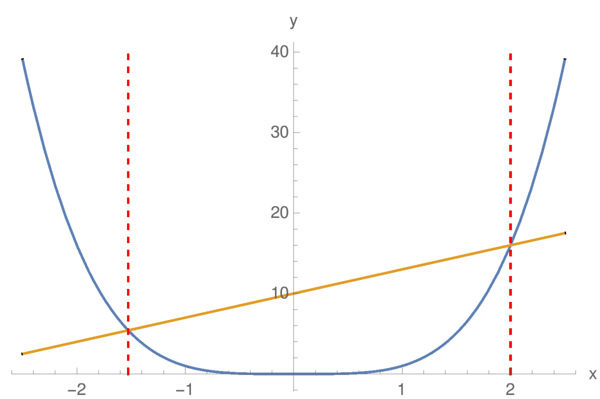

Figure 1. The function $f(x) = x^4$ is plotted in blue. The convexity of this function can be seen by picking any two values of x and noting that $f(x)$ will always lie below or on the chord connecting these two points.

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

* 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

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].

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_k, $(k = 1, 2, ..., n)$ 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 or γ) at each receiver

$$\gamma = \frac{S}{I + \sigma} \tag{4}$$

where S is the power of the desired signal, I is the interference power and σ is the background noise at a receiver.

How can we minimise the total power consumption, P, of transmitters, yet achieve this minimum SINR, γ_0, for all i receivers $(i = 1, 2, ..., m)$? 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 vector σ, a maximum power constraint P_{max} of each transmitter and the fact that all transmitters must have positive power:

$$\begin{aligned} \underset{p}{\text{minimise}} \quad & \sum_k p_k \\ \text{subject to} \quad & p_k \le P_{\max} \\ & p_k \ge 0 \\ & \gamma_i \ge \gamma_0. \end{aligned}$$

Supposing that we want to receive a signal at receiver i from transmitter k, we define the recipient matrix, δ, as

$$\delta_{ij} = \begin{cases} 1, & j = k \\ 0, & j \ne k \end{cases},$$

and our signal potential matrix $\hat{S}$ as $\hat{S} = G * \delta$. This means that signal received at transmitter i is, $S_i = \hat{S}_{ik} p_k = G_{ik} p_k$. While the interference potential matrix is defined as $\hat{I} = G - \hat{S}$, giving the interference at i is $I_i = (\hat{I} * p)_i = \sum_{j \ne k} G_{ij} p_j$. We must note that DCP does not allow division of the optimisation variable p_i, as it cannot guarantee curvature, so

$$\gamma_i \ge \gamma_0 \Longleftrightarrow \frac{S_i}{\sigma_i + I_i} = \frac{G_{ik} p_k}{\sigma_i + \sum_{j \ne k} G_{ij} p_j} \ge \gamma_0, \quad \forall i$$

is rearranged to,

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

which is affine, and hence a DCP function.

Path gain

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, whilst remaining within the DCP framework.

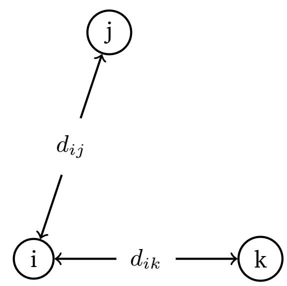

Figure 2. Path lengths between receiver i and transmitters j and k.

Implementation

For ease of terminology, we will make a few simplifying assumptions. First, we set the number of transmitters equal to the number of receivers, so $n = m$. We also pair receiver i to transmitter k such that $i = k$. This means that the recipient matrix δ is the $n \times n$ identity matrix.

For paired transmitters and receivers $(k = i)$, we randomly sample distances d from a scaled $\beta(2, 2)$ distribution, such that $d_{ii} \sim 0.2 \times \text{Beta}(2, 2)$. For unpaired transmitters and receivers $(k \ne i)$, we randomly sample distances d from a uniform distribution, such that $d_{ik} \sim \text{Uniform}(0, 1)$.

We have chosen the background noise vector σ as constant for each receiver, with $\sigma_i = 5$.

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

feasibility_count = 0
```

```
total_power = 0

for i in range(5000):
  np.random.seed(i)

  # Define variables
  n = 3 # number of transmitters = number of
    receivers
  k = 0.025 # receiver coefficient, assume uniform
  δ = np.identity(n) # identity matrix
  d = np.random.rand(n,n) # transmitters placed
    distance d∈U[0,1] from receivers
  d_diag = np.random.beta(2,2,size=n)*0.20
  d = np.tril(d) + np.tril(d, -1).T-np.diag(d)*δ+
    d_diag*δ # make the matrix symmetric with
    smaller diagonal elements
  # print(d)

  G = np.zeros((n,n)) # gain matrix G
  G = k / d ** 3.5

  # print(G)

  S = G * δ # desired signal matrix
  I = G - S # interference matrix

  σ = 5.0 * np.ones(n)
  γ = 1.0 # minimimum acceptable SINR
  Pmax = 1.0 # total power for the n transmitters

  # 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,
                 p <= Pmax]

  # Solve problem and print solution
  prob = cvx.Problem(obj, constraints)
  prob.solve()
  powers = np.asarray(p.value)

  # 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:.3f}'.format(j,powers
    [j][0]))

  if prob.status == 'optimal':
    feasibility_count += 1
    total_power += p.value

print(feasibility_count)
print(total_power/feasibility_count)
```

OUTLOOK

The authors hope that this article shows the utility of `cvxpy` for simple 'toy' problems of real-world relevance. `cvxpy` can handle problems with up to $\sim 10^3$ transmitters, however this takes $\sim 10^4 - 10^5$ms to run, hence for large real-world problems a faster implementation would likely be required (e.g. C++ based library). However there is obviously a trade off in terms of ease of understanding with these packages.

We believe that `cvxpy` 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.