A

Modeling the Perron-Frobenius Eigenvalue by Optimization Software

In this appendix, we introduce a software implementation for modeling the Perron-Frobenius eigenvalue function encountered in the specific optimization problems studied in this monograph. The software is useful for the numerical evaluation of the Perron-Frobenius eigenvalue function as well as for the rapid prototyping of the formulation and solution of various wireless network optimization problems.

A.0.1 A software implementation

The software implementation of a Perron-Frobenius eigenvalue function Matlab routine is created using the widely-used `cvx` optimization software package in [38]. The `cvx` software is an optimization parser-solver that runs in Matlab and is freely available for download from the World Wide Web. The parser in the `cvx` software can automatically identify optimization problems that are appropriately modeled using a set of libraries known as the *atom library* in `cvx`. The *atom library* contains implementation of convex functions that are commonly encountered in practice. In this way, the modularity of the *atom library* software makes it easy to model and solve specific optimization problems by adding new `cvx` atoms in the atom library.

In particular, the `cvx` atoms that implement the mathematical functions use a software paradigm known as the *disciplined convex programming* ruleset (or DCP ruleset for short). These are rules that are drawn from basic principles of convex optimization (cf. [15]), and a violation of these modeling rules can lead to a parsing error. As such, creating new `cvx` atom has to follow the DCP ruleset.

Let us examine the following convex function encountered in (2.6) and also in (2.8) given by:

$$f(\boldsymbol{\eta}) = \rho\big(\,\mathrm{diag}(e^{\boldsymbol{\eta}})\mathbf{B}\big). \tag{A.1}$$

As discussed in Chapter 2, this function is log-convex [46, 61]. Furthermore, Corollary 2.5 characterizes the solution to an inverse eigenvalue problem in nonnegative matrix theory, and this solution can in fact be efficiently obtained by solving a convex optimization problem involving (A.1) (cf. (2.8) in Chapter 2 and also [78]). In the following, we describe how to model and solve this convex optimization problem numerically using the `cvx` software.

To model this function in `cvx` and to follow the DCP ruleset in `cvx`, we proceed to evaluate (A.1) by leveraging the linear Perron-Frobenius theorem (Theorem 2.1) to rewrite (A.1) as a function:

$$f : \mathcal{R}^L \to \mathcal{R}_+, \quad f(\boldsymbol{\eta}) \triangleq \inf\{\lambda \,|\, \mathrm{diag}(e^{\boldsymbol{\eta}})\mathbf{B}\mathbf{z} \leq \lambda \mathbf{z}\},$$

where $\mathbf{z} \in \mathcal{R}^L$ is an (auxiliary) optimization variable.

Notably, (A.1) can be evaluated numerically by solving a geometric program (i.e., a class of convex optimization problems [27, 15, 14]) that the `cvx` software can handle. Using the `cvx` software modeling format, we list down the Matlab routine code of the Perron-Frobenius eigenvalue function as follows:

```
function cvx_optval = spectral_radius( eta, B )
s = size( B, 1 );
cvx_begin gp
    variables rho z( s )
    minimize( rho );
    subject to
        diag( exp(eta) ) * B * z <= rho * z;
cvx_end
```

After putting the above `function cvx_optval` code into a subdirectory folder `@cvx` (i.e., the atom library) located in the `cvx` software installation directory on the computer, the `cvx` software automatically recognizes a function call of `spectral_radius` whenever it is found in the constraints or the objective function of a convex optimization problem that conforms to the DCP ruleset.

Let us give an illustrate example using the `cvx` Matlab code to solve the following convex optimization problem:

$$\begin{array}{rl} \text{maximize} & \boldsymbol{\omega}^\top \tilde{\boldsymbol{\gamma}} \\ \text{subject to} & \log \rho\big(\operatorname{diag}(e^{\tilde{\boldsymbol{\gamma}}})\mathbf{B}_l\big) \leq 0, \quad l = 1, \ldots, 3 \\ \text{variables:} & \tilde{\boldsymbol{\gamma}}. \end{array} \tag{A.2}$$

Using the `cvx` atom function `spectral_radius` given in the above, the `cvx` Matlab code to solve (A.2) is given by:

```
cvx_begin
    variables gamma_tilde(3)
    maximize( w' * gamma_tilde );
    subject to
        log( spectral_radius( gamma_tilde, B1 ) ) <= 0;
        log( spectral_radius( gamma_tilde, B2 ) ) <= 0;
        log( spectral_radius( gamma_tilde, B3 ) ) <= 0;
cvx_end
```

Note that, in the above, `w` is a given constant positive vector, and `B1`, `B2` and `B3` are all constant nonnegative matrices with entries given in terms of the problem parameters. The optimal primal solution of (A.2) is numerically stored in the vector `gamma_tilde`. Exploring the use of the `dual variable` software option in the `cvx` Matlab code to access the optimal Lagrange dual solution of (A.2) and to compare that numerically with the Schur product of the Perron-Frobenius eigenvectors of some nonnegative matrix (say, to verify Collorary 2.5) is left as an exercise for the reader.