upLCPsolver

Unfortunately, MarkDown files on GitHub do not properly display LaTeX. If desired, see the PDF version.

Introduction

upLCPsolver is a software package designed to solve the uni-parametric Linear Complementarity Problem (upLCP):

$$w - M(\theta)z = q(\theta)$$
$$w^\top z = 0$$
$$w, z \geq 0$$

upLCPsolver offers a simplification of the methodology developed for multi-parametric Linear Complementarity Problems (mpLCP) by Nathan Adelgren.[1] Specifics of the methodology employed within upLCPsolver are outlined in a paper submitted for publication in March 2022. This document will be updated with a reference if/when that work is published.

Background

upLCPsolver is written in Python and is released as open source code under the (enter license information here). The code has been written by Nathan Adelgren.

upLCPsolver has only been tested on Linux systems, but should be compatible with other operating systems as long as all dependencies listed below can be met.

Dependencies

upLCPsolver depends on:

- Python 3 – Download from python.org or install with your favorite package manager
- PARI and CyPari2 – Can be installed using apt (or similar) and pip, respectively. **Note**, however, that testing of upLCPsolver with PARI version 2.11 (the version available via the apt repository at the time of this writing) *was not successful*. Successful testing was conducted using PARI version 2.14, compiled from source. Instructions for compiling PARI from source can be found in Section 3 of this document.

Additionally, the following Python libraries are employed by upLCPsolver:

- sys
- re
- logging
- time
- multiprocessing
- random
- os
- collections
- math
- copy

Using upLCPsolver

upLCPsolver is called from the command line as follows:

```
> python3 upLCP_solver.py /path/to/data/file (options)
```

[1] Adelgren N. Advancing Parametric Optimization: On Multiparametric Linear Complementarity Problems with Parameters in General Locations. Springer, 2021. (DOI)

Data File The data file may have any extension, but must be a text file containing the following (in the specified order):

- h – an integer – the dimension of upLCP decision variable vectors.
- k – an integer – the number of parameters present in the instance of upLCP. (Value must be 1)
- M_data – a matrix – describes the nonzero contents of $M(\theta)$ in the following format:
  - Each row of M_data must consist of four entries (comma delimited):
    1. row index
    2. column index
    3. parameter index (0 indicates the constant term)
    4. coefficient

  See below for an example.
- q_data – a matrix – describes the nonzero contents of $q(\theta)$ in the same format used above for M_data.
- Param_Space – a matrix – it is assumed in this implementation that the set Θ of attainable parameter values can be represented as a system of linear inequalities in the form $Av \leq b$. Hence, "Param_Space" describes the nonzero entries of matrix A in the following format:
  - Each row of Param_Space consists of three entries (comma delimited):
    1. row index
    2. column index
    3. coefficient
- Param_Space_RHS – a vector – provides the elements of the right-hand-side vector b, as described above. Note that Param_Space_RHS should be given in column format and even zero elements must be included.
- END – specifies the end of the file.

As an example, consider the following instance of upLCP:

$$w - \begin{bmatrix} 0 & 0 & -2 & -1 \\ 0 & 0 & -5 & \theta+7 \\ 1 & 3 & 0 & 0 \\ 1 & -\theta-5 & & 0 \end{bmatrix} z = \begin{bmatrix} 2 \\ \theta+2 \\ 20 \\ 10 \end{bmatrix}$$
$$w^\top z = 0$$
$$w, z \geq 0$$

with $\theta \in \Theta = [-3, 1]$.

The data file for this instance would be as follows:

```
h
4

k
1

M_data
1,3,0,-2
1,4,0,-1
2,3,0,-5
2,4,0,7
2,4,1,1
3,1,0,1
3,2,0,3
4,1,0,1
4,2,0,-5
4,2,1,-1
```

```
q_data
1,0,2
2,0,2
2,1,1
3,0,20
4,0,10

Param_Space
1,1,-1
2,1,1

Param_Space_RHS
3
1

END
```

Options Options can be passed to upLCPsolver as command line flags in the form "-Flag Value". Most options are used to set the value of an individual parameter. Available options are:

- -numThreads – A positive integer, less than or equal to the number of available threads, specifying the number of threads to use when partitioning the parameter space. (Default: the number of available threads)
- -parStart – A boolean indicating whether or not the initial search interval should be divided into "numThreads" subintervals and have the partitioning search begin by processing each subinterval in parallel. We note that during testing we observed that finding initial bases was quite time consuming, and thus, using all available threads to find initial bases caused poor performance. (Default: False)
- -showProgress – A boolean indicating whether or not information about the intervals being processed should be displayed throughout execution. (Default: True)

Note: At the command line, appropriate values for booleans are assumed to be "T" and "F".

Full Example of Calling upLCPsolver from the Command Line:

```
> python3 upLCP_solver.py /path/to/data/file -numThreads 4 -parStart F -showProgress T
```

Licensing