mpLCPsolver

Unfortunately, MarkDown files on GitHub do not properly display LaTeX. See the PDF version

Introduction

mpLCPsolver is a software package designed to solve the multiparametric Linear Complementarity Problem (mpLCP):

$$\begin{gathered} w - M(\theta)z = q(\theta) \\ w^\top z = 0 \\ w, z \geq 0 \end{gathered}$$

mpLCPsolver employs the methodology previously developed by N. Adelgren.[1]

Background

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

mpLCPsolver 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

mpLCPsolver depends on:

- Python 3 – Download from python.org or install with your favorite package manager
- Pyomo – Can be installed using pip
  - A solver for linear programs (LP's) ("visible" to pyomo), such as GLPK, CPLEX, Gurobi, Xpress, etc.
  - *Optionally* a solver for nonlinear programs (NLP's) (also "visible" to pyomo) can be provided. Some possibilities are Ipopt, Conopt, Baron, etc. When no NLP solver is provided, mpLCPsolver employs a variant of an ellipsoid method based on the work of Shah, et al.[2]
- PARI and CyPari2 – Can be installed using apt (or similar) and pip, respectively. **Note**, however, that testing of mpLCPsolver 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.

Using mpLCPsolver

mpLCPsolver is called from the command line as follows:

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

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 mpLCP decision variable vectors.
- k – an integer – the number of parameters present in the instance of mpLCP.
- 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

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

[2] Shah S., Mitchell J.E., and Kupferschmid M. An ellipsoid algorithm for equality-constrained nonlinear programs. Computers & Operations Research, 28.1 (2001): 85-92. (DOI)

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 mpLCP:

$$w - \begin{bmatrix} 0 & 0 & -2 & -1 \\ 0 & 0 & -5 & \theta_1 + 7 \\ 1 & 3 & 0 & 0 \\ 1 & -\theta_1 - 5 & & 0 \end{bmatrix} z = \begin{bmatrix} -\theta_2 - 1 \\ \theta_1 - \theta_2 - 1 \\ -18\theta_2 - 34 \\ -9\theta_2 - 17 \end{bmatrix}$$
$$w^\top z = 0$$
$$w, z \geq 0$$

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

The data file for this instance would be as follows:

```
h
4

k
2

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,-1
1,2,-1
2,0,-1
2,2,-1
2,1,1
3,0,-34
3,2,-18
```

```
4,0,-17
4,2,-9

Param_Space
1,1,-1
2,2,-1
3,1,1
4,2,1

Param_Space_RHS
3
3
1
1

END
```

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

- -linearSolver – A string indicating the solver to be used by pyomo when solving LP or MIP problems. (Default: "glpk")

  For a list of solvers "visible" to pyomo, in a terminal window type, "pyomo help -s".

- -nonlinearSolver – A string indicating the solver to be used by pyomo when solving NLP or MINLIP problems. (Default: "", i.e., empty string)

- -parPivot – A boolean indicating whether or not principal pivot operations carried out on the problem's tableau should be computed in parallel. (Default: False)

- -useCrissCross – A boolean indicating whether or not the Criss Cross method[3] for LCP should be used in attempt to find a starting basis. Note that if a starting basis is successfully discovered using the Criss Cross method, Phase 1 of the method of N. Adelgren[1] is skipped, which can provide a significant decrease in execution time. (Default: False)

- -nlpFeas – A boolean indicating whether or not NLP problems should be posed as feasibility problems rather than optimization problems. Specifically, the NLP's solved internally by mpLCPsolver do not need solved to optimality in order for the methodology employed by mpLCPsolver to function correctly. Only a feasible point with strictly positive (or negative, depending on the situation) objective value is needed. So, posing the problem as a feasibility problem by setting the objective as a constant and adding a constraint forcing the appropriate strict sign of the original objective may result in decreased execution times. (Default: False)

- -checkFwithEH – A boolean indicating whether or not to check if any of the defining constraints of an invariancy region can be shown to form (k-1)-dimensional boundaries of the region while building the special sets E and H. (Default: True)

- -checkDimWithF – A boolean indicating whether or not to check the dimension of an invariancy region while building the special set F. (Default: True)

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

[3]den Hertog, D., Roos, C., and Terlaky, T. The linear complimentarity problem, sufficient matrices, and the criss-cross method. Linear Algebra and Its Applications, 187 (1993): 1-14. (DOI)