MATLAB Implementation for "Fast Iterative Method for SOAV Minimization Problem with Linear Equality and Box Constraints and Its Linear Convergence"

Mitsuru Toyoda (Tokyo Metropolitan University, toyoda atmark tmu.ac.jp)

This note describes MATLAB implementation for "Fast Iterative Method for SOAV Minimization Problem with Linear Equality and Box Constraints and Its Linear Convergence".

Usage

- The numerical experiment is conducted by executing the main file "`main_run_LP_ADMM.m`".
- NOTE: In the MEX source files "`soav_*.cpp`", which are the implementation of the proposed algorithms and the conventional ADMM, Eigen library (`https://eigen.tuxfamily.org/`) is used for basic linear algebra. The main file "`main_run_LP_ADMM.m`" assumes Eigen library is located at a directory `C:\Library\cpp\eigen-3.3.8`. If you use another directory or version of Eigen, please modify the include path indicated in "`main_run_LP_ADMM.m`".

Description of Files

This section describes the codes used in the numerical experiment. For further details, refer to the codes themselves.

- Main file:
  - `main_run_LP_ADMM.m`
    The main file to execute numerical experiment. In the first part of the file, some options, including `ConsiderQuadraticCost` for considering a quadratic cost and `gamma` meaning the parameter γ in ADMM, are available.
- Sub-Routine files:
  - `sub_MATaverage.m`
    This MATLAB function calculates the average value of a variable in multiple MAT-files. This is used to calculate the average computation time.
  - `sub_instance_make.m`
    This is a MATLAB function to make a MAT-file which includes resulting instances calculated by its arguments, which are tables of n_w and N.
  - `sub_simulate_LP_ADMM.m`
    This program calls proposed algorithms, conventional ADMM algorithm, and LP/QP approach for each instance MAT-file made by `sub_instance_make.m`.
- SOAV minimization functions:
  - `soav_bisec.cpp`
    This code is the implementation of the proposed bisection-search based algorithm. It has the following compile options:
    * A compile option `ADAPTIVE_ON` is activated, the adaptive γ updating is used.
    * A compile option `PDOptimalityCriteria_ON` uses the stopping criteria based on the primal and dual feasibility (This option is deactivated in the numerical experiment of the manuscript).

    The arguments are passed in the form of

    ```
    [x_opt,fval,exitflag,output]
     = soav_bisec(u,w,lb,ub,Aeq,beq,Q,c,gamma,y0,z0,MexOptions),
    ```

    where `u` $= [\boldsymbol{u}_1^{\mathrm{T}}, \ldots, \boldsymbol{u}_{n_w}^{\mathrm{T}}]$, `w` $= [w_1, \ldots, w_{n_w}]^{\mathrm{T}}$, `lb` $= \underline{\boldsymbol{x}}$, `ub` $= \overline{\boldsymbol{x}}$, `Aeq` $= \boldsymbol{A}$, `beq` $= \boldsymbol{b}$, `Q` $= \boldsymbol{Q}$, `c` $=$

$\boldsymbol{c}$, `gamma` $= \gamma$, `y0` $= \boldsymbol{y}_0$, `z0` $= z_0$, and `MexOptions = [fvalTrue,fvalTol,ConstraintTol]`. The tolerance parameters are defined in the manuscript. The outputs `x_opt` and `fval` are an optimal solution and a resulting objective function value, respectively. `exitflag` is an exit-flag whose value 1 means the stopping criteria are fulfilled and -1 means the number of iteration reaches the predefined limit. `output` provides the information, such as the constraint violation and the computation time.

- `soav_table.cpp`

  This code is the implementation of the proposed table based algorithm. The usage is the same as that of `soav_bisec.cpp`. Note it has no `ADAPTIVE_ON` option.

- `soav_conventional.cpp`

  This code is the implementation of the conventional ADMM algorithm. Since the conventional algorithm was proposed for a problem without a quadratic cost, the arguments `Q` $= \boldsymbol{Q}$ and `c` $= \boldsymbol{c}$ are not required to be provided.

- `soav_LP_QP.m`

  A MATLAB function to make a Gurobi model and call Gurobi LP/QP solver. The arguments are passed in the form of

  ```
  [x_opt,fval,output] = soav_LP_QP(u,w,lb,ub,Aeq,beq,Q,c,options),
  ```

  where `options.Method` indicates an algorithm, and `output` returns the information associated with the LP/QP solver.