How to use LMITOOL

1 What is it?

LMITOOL is a software tool that allows to easily develop matlab code solving semidefinite programming problems. It comes with a graphical interface to three different SDP solvers.

The user first specifies the *names* of the variables of the problem, the *names* of the data matrices,]. Then, using a simple `matlab` syntax, the user codes up in a matlab syntax the linear matrix inequalities, equalities, and objective of the SDP. Then LMITOOL generates a `matlab` function that solves the problem. To solve a particular instance, the user must declare the data within `matlab`, and run the function generated by LMITOOL.

Warning: `lmitool` should be thought of as a debugging help, not a high-performance solver. If you're after performance, you should use "direct" codes for solving SDP, at the expense of more work.

2 How to use it?

Once `lmitool` is installed (see section 3), type the following at a Unix prompt:

```
> tklmitool
```

A window appears. Use the Help menu to understand how to program the problem.

Let us use an example. For $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times n_u}$, $C \in \mathbf{R}^{n_y\times n}$, we seek to solve the problem

$$\text{minimize } \mathbf{Tr}\, P \text{ subject to } F(P) \succeq 0, \quad P \succeq 0,$$

where the matrix variable P is symmetric, and $n \times n$, and

$$F(P) := \left[\begin{array}{cc} A^T P + PA + C^T C & PB \\ B^T P & I \end{array}\right].$$

Clearly, the above problem is an SDP in variables P_{ij}, $1 \leq i \leq j \leq n$.

Here is how `lmitool` can help you write a `matlab` function

```
[P,infostr] = ric_lmi(A,B,C)
```

that returns an optimal P if the problem is feasible, an empty matrix otherwise, and a string `infostr` containing convergence information.

First, we type the unix command `tklmitool`. The window contains three top fields:

- `Function name`
- `Variable list`
- `Data list`

and five larger fields at the bottom:

- `Preliminary computation`
- `Initial guess`
- `Linear Matrix Equalities`
- `Linear Matrix Inequalities`
- `Linear objective`

(It is possible that the first of the above fields, `Preliminary computation`, is not shown. If so, turn on the first button under the field `Config` at the top-level menu.)

The first thing is to fill the top fields. We have to choose a *name* for the function, say `ric_lmi` (a motivation for this choice will be clear later); a name for the variables, here `P`; and a name for the data, here `A`, `B`, `C`. Now `lmitool` understands the *syntax* of the function `ric_lmi`, but it does not understand the problem itself.

The next five fields should contain the description of the problem, just as a text (or symbolic) description.

- `Preliminary computation`: This field is optional, but is handy to define some intermediate quantities that might be useful in any or all other fields. For example, we might set

  ```
  [n,nu] = size(B);
  nu = eye(nu,nu);
  ```

  Just because we'll need `n`, `nu` and `Inu` later.

- `Initial guess`: The purpose of this field is mainly to define the *size* of the variables. By setting the variable P to any matrix of size $n \times n$, we are defining the size of P. Here, we might set

```
Pinit = zeros(n,n);
```

- `Linear Matrix Equalities`: In this field, we may define the equality constraints of the problem. Up to now, P is any matrix of size $n \times n$, so we must add the constraint that it is symmetric.

  ```
  LME = P-P';
  ```

  does the job.

- `Linear Matrix Inequalities`: In this field, we specify the LMI constraints. Here, we have two LMIs. So we may use `stack.m,` as follows:

  ```
  LMI1 = P;
  LMI2 = [A'*P+P*A+C'*C P*B; B'*P Inu];
  LMI = stack(LMI1,LMI2);
  ```

  Note that it is implicitely understood that $LMI \succeq 0$ is the corresponding constraint. Note also that we may use stack in a loop, if the number of constraints depends on the data of the problem.

- `Linear objective`: the objective is to maximize $\mathbf{Tr}\, P$, so

  ```
  OBJ = -trace(P);
  ```

Once the fields are all completed, you may

- save your work. The command `save` under the `file` menu will save the fields, under a file `ic_lim.lmt`.
- build `matlab` files. Go under the `Build` menu at the top level, and activate the `Build` command.

You're done! Now, go under `matlab`. Once the data A, B, C exists in the workspace, type

```
>> [P,infostr] = ric_lmi(A,B,C);
```

and the code will return P and `infostr`. You can check that, at the optimum, P satisfies the Riccati equation

$$A^T P + PA + C^T C - PBB^T P = 0.$$

If you wish, you may add code to the function `ric_lmi` by editing it. Advanced users may also edit the so-called evaluation function, named here `ric_lmi_eval`. This function is a `matlab` transcription of the five fields described above.

3 How to install the package

The software runs on a variety of machines, but for now I can only help you for the Unix machines. Hopefully, some of you will know how to work with NT or Linux too.

Copy to a directory `matlab` under your home directory, the file

```
/home/e/ee290n/matlab/startup.m
```

Add the following commands at the end of your `.cshrc` file:

```
set ee290nhome = /home/e/ee290n
set path = ( $path /usr/sww/bin )
set path = ( $path /usr/sww/bin/wish)
set path = ( $path $ee290nhome/bin )
set path = ( $path $ee290nhome/matlab )
set path = ( $path $ee290nhome/lmitool)
set path = ( $path $ee290nhome/lmitool/SDPHA-1.1)
set path = ( $path $ee290nhome/lmitool/semidef-beta)
set path = ( $path $ee290nhome/lmitool/lmitool-2.0)
set path = ( $path $ee290nhome/lmitool/tklmitool-2.0)
set path = ( $path $ee290nhome/lmitool/tklmitool-2.0/bin)
set path = ( $path $ee290nhome/lmitool/lmitool-2.0/tklmitool-2.0/lib/lmit-2.0)
```

You should now be ready to use `tklmitool`.