In this tutorial, we will show how to use the sparse LP solver. This solver aims to solve the following optimisation problem

$$\begin{array}{ll} \min & \|x\|_0 \\ s.t & A_{eq}x = b_{eq} \\ & A_{ieq}x \leq b_{ieq} \\ & l \leq x \leq u \end{array} \tag{1}$$

It has been proved that zero-norm is a non-convex function and the minimisation of zero-norm is a NP-hard problem. Non-convex approximations of zero-norm extensively developed. For a complete study of non-convex approximations of zero-norm, the reader is referred to [1].

The method is described in [1]. The sparse LP solver contains one convex ($\ell_1$norm) and 6 non-convex approximation of zero-norm

- Capped-L1 norm
- Exponential function
- Logarithmic function
- SCAD (Smoothly Clipped Absolute Deviation) function
- ℓ_p norm with $p < 0$
- ℓ_p norm with $0 < p < 1$

The tutorial consist of two parts. Part 1 shows a basic usage of the solver. In part 2 provides an application of the code for finding the minimal set of reactions subject to a LP objective. Ready-made scripts are provided for both parts.

Example of using sparseLP solver on randomly data

One randomly creates a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x_0 \in \mathbb{R}^n$. The right hand side vector b is then computed by $b = A * x_0$.

```
n = 100;
m = 50;
x0 = rand(n,1);
constraint.A = rand(m,n);
constraint.b = constraint.A*x0;
constraint.lb = -1000*ones(n,1);
constraint.ub = 1000*ones(n,1);
constraint.csense = repmat('E', m, 1);
```

There are three optional inputs for the method. The two first ones : maximum number of iterations (*nbMaxIteration*) and threshold (*epsilon*) are stopping criterion conditions. *theta* is the parameter of zero-norm approximation. The greater the value of *theta*, the better the approximation of the zero-norm. However, the greater the value of *theta*, the more local solutions the problem (1) has. If the value of *theta* is not given then the algorithm will use a default value and update it gradually.

```
params.nbMaxIteration = 100;    % stopping criteria
params.epsilon = 10e-6;      % stopping criteria
params.theta   = 2;          % parameter of l0 approximation
```

Call the solver with a chosen approximation

```
solution = sparseLP('cappedL1',constraint,params);
% solution = sparseLP('cappedL1',constraint);
```

Finding the minimal set of reactions subject to a LP objective

Load a COBRA model

```
model = load('iLC915.mat');
```

We will firstly find the optimal value subject to a LP objective

```
%% Maximize
% max c'v
% s.t    Sv = b
%        l <= v <= u
% Define the LP structure
[c,S,b,lb,ub] = deal(model.c,model.S,model.b,model.lb,model.ub);
[m,n] = size(S);
csense = repmat('=',m,1);
LPproblem = struct('c',-c,'osense',1,'A',S,'csense',csense,'b',b,'
    lb',lb,'ub',ub);
% Call solveCobraLP to solve the LP
LPsolution = solveCobraLP(LPproblem,params);
```

We will now find the minimum number of reactions needed to attain the same max objective found previously. Then one will add one more constraint: $c^T v = c^T v_{FBA} =: f_{FBA}$.

```
constraint.A = [S ; c'];
constraint.b = [b ; c'*vFBA];
constraint.csense = repmat('=',m+1,1);
constraint.lb = lb;
constraint.ub = ub;
```

Call the sparseLP solver to solve the problem

$$\begin{array}{ll} \min & \|v\|_0 \\ s.t & Sv = b \\ & c^T v = f_{FBA} \\ & l \leq v \leq u \end{array} \tag{2}$$

```
% Try all non-convex approximations of zero norm and take the best
    result
approximations = {'cappedL1','exp','log','SCAD','lp-','lp+'};
bestResult = n;
bestAprox = '';
for i=1:length(approximations)
  solution = sparseLP(char(approximations(i)),constraint);
  if solution.stat == 1
    if bestResult > length(find(abs(solution.x)>eps))
      bestResult=length(find(abs(solution.x)>eps));
      bestAprox = char(approximations(i));
      solutionL0 = solution;
    end
  end
end
```

References

[1] *Le Thi et al., DC approximation approaches for sparse optimization, European Journal of Operational Research, 2014, http://dx.doi.org/10.1016/j.ejor.2014.11.031.*