Laboratory 1

PÎNDICHI Elena and JAROUSSEAU Arthur

September 29, 2024

1 Set manipulation and visualization

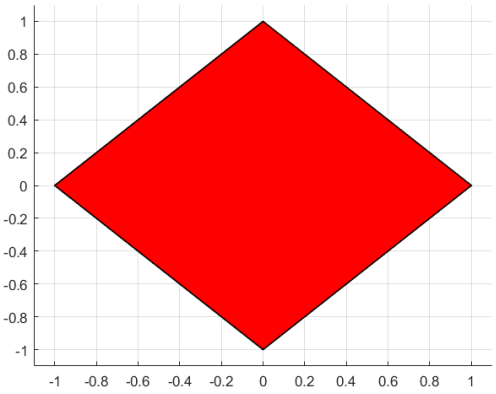
1.1 Polyhedra definitions in Matlab

A convex polyhedron is the intersection of a finite set of halfspaces of $\mathbb{R}^n$. A convex polytope is a bounded convex polyhedron, which means it is defined as a bounded intersection of a finite number of hyperplanes. We wanted to represent the following polytope, for which we have chosen our own set of matrices:

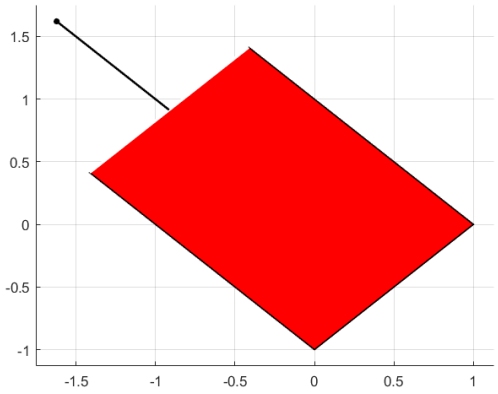
$$P = \{x : Ax \leq b, A_e x = b_e\} \tag{1}$$

the chosen halfspace is represented by A and b below:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & -1 \\ -1 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{2}$$



(a) Bounded Polytope



(b) Unbounded Polytope

In order to achieve these points, we first drew the shape on a paper and we determined the slope for each pair of points. Hence, we obtained a equation depending on the 2 elements from the state and we could write the halfspace equation. As seen in Fig. 1a, the red part is the intersection of all 4 halfspaces. In the second figure, Fig. 1b, the line indicates the infinite expansion of the halfspaces reunion.

For the dual representation, we chose another object, a simple rectangle for which we defined the vertices and we created another figure. But this time, the algorithm computed the convex hull and it removed the redundant vertices. The convex hull is used, in order to be sure that we still remain in the convex problem.

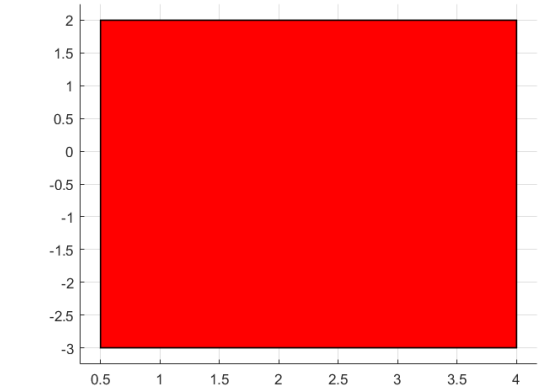


Figure 2: Rectangle

After we received 2 different polytopes, we tried to concatenate them. (Fig. 1a and Fig. 2) We can observe that, if we try to access the A matrix of the concatenated figure, it will have 2 matrices for the halfspace representation, which means it will contain both of the polytopes.

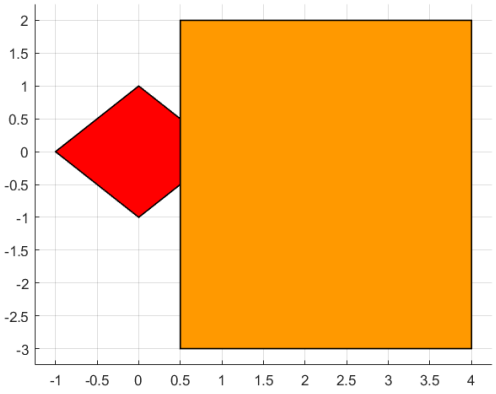
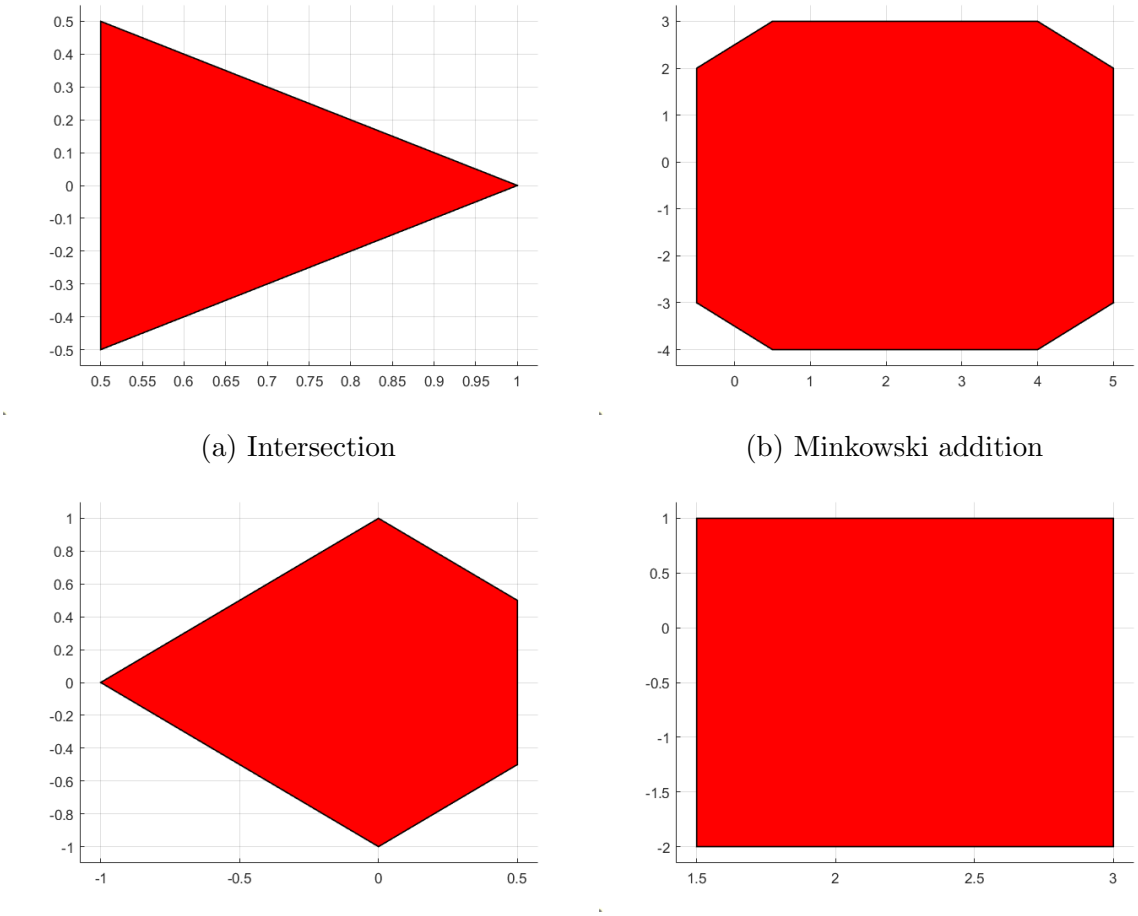


Figure 3: Concatenation

1.2 Polyhedral operations

We tried to implement operations on Fig. 1a and Fig 2 polytopes. The first operation was the *inclusion*, which returned '0' as expected, due to the fact that they are not fully contained in one and another. We tested the *equality* which ensured us that they are different polytopes. The *intersection* test returned the only part that is common to both of the polytopes, as seen in Fig. 4a. The *Minkowski addition* returns the larger polytope returned after adding each point from one polytope to the other, as in Fig. 4b.

We also computed the *set difference* between them and we have seen that it will return only the remaining part of the polytope from which you are extracting the common part, in Fig. 4c.



(a) Intersection

(b) Minkowski addition

(c) Set Difference

(d) Difference

Figure 4: Operations

2 Solving optimization problems

2.1 Optimization problems with Yalmip in Matlab

For the LP problem we used some tools defined in Matlab in order to create a parametric value of x and we received the solution by calling the *optimization* function. For the QP problem, we created a H matrix, which is supposed to be positive definite, meaning that the eigenvalues have to be positive. H has to be positive definite in order make sure we are in a convex problem. The optimal point for the LP problem was $x^* = \begin{bmatrix} 4 & -3 \end{bmatrix}^\top$ and for QP was $x^* = \begin{bmatrix} 0.5 & 0.2007 \end{bmatrix}^\top$.

2.2 Optimization problems using CasADi in Matlab

When working with CasADi toolbox, we could see that the time to compute and receive the solution does not differ that much than when we used the Yalmip toolbox and we can confirm that the solutions are the same. We could observe that we received the optimal solution, which was located in the vertices, and the linear cost passes through it and is moving on the direction of the constraints, as seen in Fig. 5. For the LP problem, the optimal point is $x^* = \begin{bmatrix} 0.5 & 2 \end{bmatrix}^\top$.

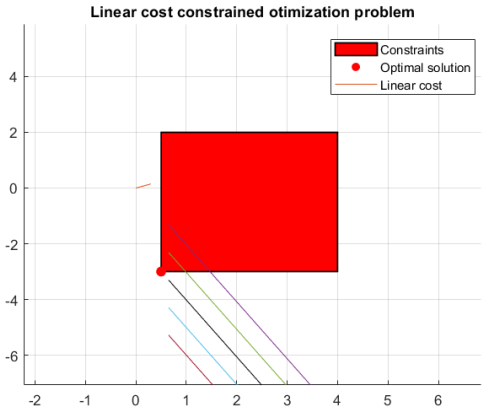


Figure 5: LP problem

For the QP problem, we received the unconstrained optimal point $x^* = \begin{bmatrix} 0.5 & -0.346426 \end{bmatrix}^\top$ as seen in Fig. 6.

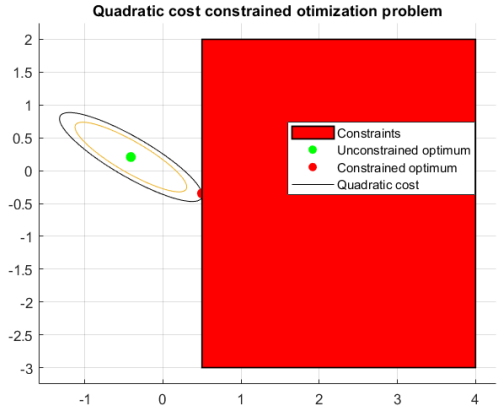


Figure 6: QP problem

3 Exercises

We considered a LTI dynamical system affected by bounded additive disturbances characterized by the matrices:

$$A = \begin{bmatrix} -0.5 & 0.1 \\ 0.2 & -0.8 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \delta = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{3}$$

We were careful to choose the A matrix in continuous time so that the eigenvalues will be stable. We first chose a matrix that had it's stable values between $[-1; 0]$. Then we chose a sample period $T_s = 0.1$ so that we discretize, and the eigenvalues respected the stable condition implied by discrete time, which means the stable values must be a part of the $[-1; 1]$ interval. We then chose a null error and we constructed the ultimate bounds set given by:

$$\Omega_{UB}(\epsilon) = \{x : |V^{-1}x| \leq (I_2 - |\Lambda|)^{-1}|V^{-1}B|\delta + \epsilon\}. \tag{4}$$

We wrote the matrices to represent the bounded polytope defined in 4 and we plotted the ultimate bounds set and its iteration in one step under the dynamics.

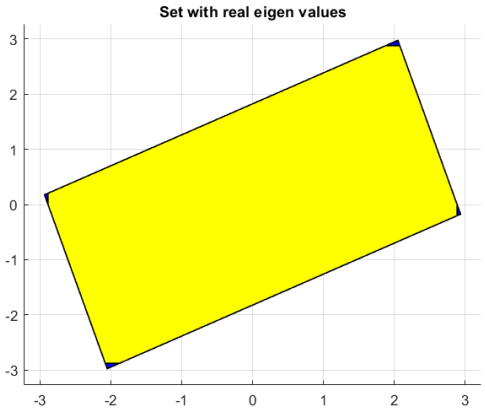


Figure 7: UB set - real eigenvalues

We checked the positive invariance via set of inclusion. As understood from the [1] article, we verified if the dynamic system contains the system state at any time, both in future and in past. In Fig. 7, we can see the blue parts is the ultimate bounds set and the yellow part is the positive invariance set. Hence, for all the future states and disturbances, the solution will still be contained and a part of the set.

References

[1] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.