L'optimisation convexe pour la conception et l'analyse des lois de commande

—

Formalisation grâce aux *LMI*

Thibault HILAIRE

sous la direction de Philippe CHEVREL

– ARMINES –

16 août 2003

Table des matières

Chapitre 1

Présentation

Le formalisme LMI est aujourd'hui un outil efficace pour la résolution de nombreux problèmes d'automatique et de commande. Les progrès réalisés à la fin des années 80 dans les domaines des algorithmes d'optimisation convexe, ont permis le développement d'outils de résolution numérique (LMI ToolBox, LMITool, SeDuMi, ...).
Une part importante de programmes actuels s'attache à formuler les problèmes de commande avec le formalisme LMI. Celui-ci permet d'écrire, sous une même forme, des problèmes qui appartiennent à des classes distinctes, et permet donc, en quelque sorte, d'obtenir une approche unifiée pour l'analyse et la commande [1]. De plus, il est possible d'empiler les contraintes LMI pour préciser les solutions recherchées.

1.1 Un bref historique des LMI dans la commande

L'histoire des LMI remonte à plus d'un siècle, maintenant. Dans les années 1890, Lyapunov montre que l'équation différentielle $\dot{x}(t) = Ax(t)$ est stable si et seulement si il existe une matrice P définie positive qui vérifie $A^\top P + PA < 0$ (inégalité de Lyapunov). Cette inégalité est une LMI particulière. Lyapunov a aussi montré que cette inégalité peut être résolue analytiquement (on peut se ramener à un système d'équation linéaire), en considérant une matrice $Q > 0$ et en résolvant l'équation $A^\top P + PA + Q = 0$.
Dans les années 40-50, Lur'e, Postnikov et d'autres en Union Soviétique appliquèrent la méthode de Lyupanov à de véritables problèmes de commande, et résolurent les LMI qui se posaient à eux "à la main" quand celles-ci étaient de faible taille.
Dans les années 60, Yakubovic, Popov, Kalman et d'autres réussirent à ob-

tenir un critère graphique (lemme Positif Réel, voir chapitre 3.3) permettant de résoudre les LMI qui se posaient à Lur'e. Cela permit de montrer comment résoudre certaines familles de LMI par des méthodes graphiques. Puis, ces LMI furent résolues en étudiant les solutions d'une équation de Riccati (ARE).
En 1971, Willems se demandait s'il existait un algorithme de calcul permettant de résoudre numériquement un problème sous formulation LMI [2] [3]. La réponse arrive une dizaine d'année plus tard, avec, par exemple, les travaux de Pyatnitskii et Skorodinskii : il est possible de reformuler un problème LMI en un problème d'optimisation convexe, que l'on peut résoudre numériquement : cela peut ainsi amener une solution pour les LMIs qu'on ne peut résoudre analytiquement. L'algorithme Ellipsoïde, qui permet de résoudre les problèmes d'optimisation convexe en temps polynomial est alors utilisé.
Enfin, en 1984, Karmarkar présente un nouvel algorithme beaucoup plus efficace que l'algorithme Ellipsoïde, redécouvre des méthodes de programmation linéaire, et ouvre la voie à la méthode des points intérieurs de Nesterov et Nemirovskii.

Chapitre 2

Bases mathématiques

Dans un premier temps, nous allons redéfinir, dans un cadre plus théorique, les notions nécessaires à l'optimisation convexe. Il s'agira de poser les bases mathématiques des problèmes rencontrés, pour mieux envisager les solutions.
Nous définirons les notions de convexité, d'optimum. Nous recentrerons alors l'étude sur les problèmes d'optimisation convexe utilisant le formalisme des LMI[1].

2.1 Notations

Dans les définitions de ce chapitre, nous utiliserons les notations suivantes :

- $\mathbb{S}$ est l'ensemble des matrices symétriques

$$\mathbb{S} \triangleq \left\{ M \mid \exists\ n \in \mathbb{N}^*\ tq\ M = M^\top \in \mathbb{R}^{n \times n} \right\}$$

- $\mathbb{V}$ désignera un espace vectoriel de dimension finie (il pourra être normé), et E un sous-ensemble de $\mathbb{V}$
- les notations > 0 et ≥ 0 peuvent aussi bien se rapporter à des réels qu'à des matrices. Dans ce dernier cas, > 0 signifie *définie positive* (pour $M \in \mathbb{R}^{n \times n}$, $M > 0$ si et seulement si $u^\top M u > 0$ pour tout u non nul de $\mathbb{R}^n$) et ≥ 0 signifie *semie-définie positive.*

[1] Linear Matrix Inequalities

2.2 Convexité

Définition 1 (Ensemble convexe)
D'une manière générale, un ensemble E est convexe si et seulement si :

$$\forall \lambda \in [0,1],\ \forall (x_1, x_2) \in E^2,\ \lambda x_1 + (1-\lambda) x_2 \in E$$

D'un point de vue géométrique, on peut conclure que, si deux points sont dans un convexe E, alors, tous les points du segment reliant ces deux points sont dans E.

Définition 2 (Fonction convexe)
Une fonction $f : E \to \mathbb{R}$ est dite convexe si et seulement si :

$$\begin{cases} E \text{ est un ensemble convexe} \\ \forall \lambda \in [0,1], \mid \forall (x_1, x_2) \in E^2,\ f(\lambda x_1 + (1-\lambda) x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \end{cases}$$

La fonction est dite strictement convexe, si l'inégalité est stricte pour $x_1 \neq x_2$

On peut noter que, *par définition*, l'ensemble de définition d'une fonction convexe doit être convexe.

Définition 3 (Fonction quasi-convexe)
Une fonction $f : E \to \mathbb{R}$ est dite quasi-convexe si et seulement si :

$$\begin{cases} E \text{ est un ensemble convexe} \\ \forall \lambda \in [o,1],\ \forall (x_1, x_2) \in E^2,\ f(\lambda x_1 + (1-\lambda) x_2) \leq \max(f(x_1), f(x_2)) \end{cases}$$

On peut remarquer que toute fonction convexe est quasi-convexe.

Définition 4 (Minimum local)
La fonction $f : E \to \mathbb{R}$ admet un minimum local en $x_0 \in E$ si et seulement si :

$$\exists \epsilon > 0 \setminus \forall x \in E,\ \|x - x_0\|_E < \epsilon \Rightarrow f(x_0) \leq f(x)$$

Ce minimum est bien sûr global si $\forall x \in E,\ f(x_0) \leq f(x)$.

On peut bien sûr, faire la même définition avec le maximum.

La propriété suivante explique tout l'intérêt que l'on peut porter aux fonctions convexes.

Proposition 1
Soit $f : E \to \mathbb{R}$ une fonction convexe. Si f admet un minimum local x_0, alors ce minimum est global. De plus, si f est strictement convexe, ce minimum est unique.

démonstration :

Prenons $f : E \to \mathbb{R}$ convexe qui admet un mimimum local en x_0. Soit ϵ tel que

$$\forall x \in E, \|x - x_0\|_E < \epsilon \Rightarrow f(x_0) \leq f(x)$$

Soit $x \in E$. En choisissant $\lambda \in [0, 1]$ suffisamment petit :

$$\lambda \leq \frac{\epsilon}{\|x - x_0\|_E}$$

$x_0 + \lambda(x - x_0)$ est dans la boule de centre x_0 et de rayon ϵ, et :

$$f(x_0) \leq f(x_0 + \lambda(x - x_0)) = f((1 - \lambda)x_0 + \lambda x)$$

$$f(x_0) \leq (1 - \lambda)f(x_0) + \lambda f(x)$$

Ainsi

$$0 \leq \lambda(f(x) - f(x_0))$$

et $f(x_0) \leq f(x)$. On peut donc en conclure que x_0 est minimum global de f. De plus, si f est strictement convexe, les inéqualités de la démonstration deviennent strictes, et le minimum est **unique**.

Cette propriété est très importante, car **elle stipule qu'il *suffit* de trouver un optimum local de la fonction convexe, pour trouver l'optimum global**. Comme beaucoup d'algorithmes de recherche d'optimum ne peuvent assurer que la convergence vers un optimum local, cette propriété des fonctions convexes améliore ainsi la portée des algorithmes d'optimisation.

Définition 5 (Problème d'optimisation convexe)
Un problème d'optimisation convexe est un problème de la forme

$$\min_{x \in \Omega} f(x)$$

Avec

* $f : E \to \mathbb{R}$ *est une fonction. C'est la fonction objectif à minimiser*
* $\Omega = \left\{ x \in E \mid \left\{ \begin{array}{ll} G_i(x) \leq 0 & i = 1, \ldots, p \\ H_j(x) = 0 & j = 1, \ldots, q \end{array} \right. \right\}$ *est l'espace de contrainte*
* $\forall i = 1, \ldots, p$ *les fonctions* G_i *sont convexes*
* $\forall j = 1, \ldots, q$ *les fonctions* H_j *sont affines*

Les fonctions G_i *et* H_j *sont à valeurs dans* $\mathbb{R}$ *ou dans* $\mathbb{R}^{n \times n}$.

Ce problème d'optimisation est dit *convexe* car la contrainte est convexe. Cela permet, à partir de deux points vérifiant la contrainte, d'utiliser les points situés sur le "segment" formé par ces deux points, en étant assurés qu'ils vérifient eux aussi la contrainte.
Démonstration :

Soient $(x_1, x_2) \in \Omega$. $\forall \lambda \in [0,1]$, on a :

- $\forall i = 1, \ldots, p, \quad G_i(\lambda x_1 + (1-\lambda)x_2) \leq \lambda G_i(x_1) + (1-\lambda)G_i(x_2) \leq 0$ (car G_i est convexe et car x_1 et x_2 sont dans Ω)
- $\forall j = 1, \ldots, q, \quad H_j(\lambda x_1 + (1-\lambda)x_2) = \lambda H_j(x_1) + (1-\lambda)H_j(x_2) = 0$

Donc $\lambda x_1 + (1-\lambda)x_2 \in \Omega$ et la contrainte définit un ensemble convexe.

Il est évident qu'on pourra d'autant plus facilement résoudre numériquement un problème d'optimisation convexe que la fonction objectif sera "sympatique". Par exemple, **si cette fonction f était convexe, nous pourrions alors nous contenter de la minimiser localement, pour assurer résoudre le problème**.
Les algorithmes que nous verrons dans le chapitre 4 demandent à la fonction objectif d'être convexe pour pouvoir assurer un résultat.

2.3 LMI

<u>Définition</u> 6 (Linear Matrix Inequality)
Une LMI (ou Inégalité Matricielle Affine en français) est une contrainte de la forme :

$$F(x) \triangleq F_0 + \sum_{i=0}^{m} x_i F_i > 0 \tag{2.1}$$

où

* $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$ *est appelé variable de décision (les* x_i *sont les inconnues du problème)*
* $F_0, \ldots, F_m$ *sont des matrices symétriques de* $\mathbb{R}^{n \times n}$

On peut aussi avoir une LMI non stricte si l'équation 2.1 n'est pas stricte ($F(x)$ semie-définie positive seulement).

Il est aussi possible, à partir d'autres formes d'inégalités, de revenir à une LMI définie en 2.1 : la mise sous forme LMI d'un problème d'optimisation demande donc d'écrire les contraintes du problème sous d'inégalités matricielles affines en fonction des variables d'optimisation [4].

• Par exemple, les inégalités matricielles $F(x) < 0$ et $F(x) > G(x)$, avec F et G des fonctions affines, peuvent être ré-écrites commes des LMI, avec $-F(x) > 0$ et $F(x) - G(x) > 0$.

• Si l'on considère un sytème de LMI

$$F^{(1)}(x) > 0, \ldots, F^{(p)}(x) > 0$$

il est possible de les regrouper en une seule LMI $F(x) > 0$ avec :

$$F(x) \triangleq \begin{pmatrix} F^{(1)}(x) & 0 & \ldots & 0 \\ 0 & F^{(2)}(x) & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & F^{(p)}(x) \end{pmatrix} > 0$$

Ainsi, nous ne distinguerons pas une LMI, d'un système de LMI.

• D'autres inégalités peuvent aussi s'écrire sous la forme d'une LMI. L'inégalité $F(X) > 0$ où $F : \mathbb{V} \to \mathbb{S}$ est une fonction affine, peut aussi s'écrire sous la forme d'une LMI, en décomposant l'inconnue X dans une base $(E_i)_{1 \leq i \leq m}$ de $\mathbb{V}$:

$$X = \sum_{i=0}^{m} x_i E_i$$

d'où

$$F(X) = F\left(\sum_{i=0}^{m} x_i E_i\right) = F_0 + \sum_{i=0}^{m} x_i F(E_i) = F_0 + \sum_{i=0}^{m} x_i F_i$$

Ainsi, l'inégalité de Lyapunov $A^\top X + XA + Q > 0$ est une LMI (pour Q symétrique). Nous verrons d'autres exemples dans le chapitre 3.

• La proposition suivante permet de mettre certaines inégalités non-linéaires sous forme LMI :

Proposition 2 (Complément de Schur)
Soit $F : \mathbb{V} \to \mathbb{S}$ une fonction affine avec :

$$F(x) = \begin{pmatrix} Q(x) & S(x) \\ S^\top(x) & R(x) \end{pmatrix}$$

où $Q(x)$ est une matrice carrée.
Alors $F(x) > 0$ si et seulement si

$$\begin{cases} R(x) > 0 \\ Q(x) - S(x)\,[R(x)]^{-1}\,S(x)^\top > 0 \end{cases} \tag{2.2}$$

démonstration :

Soit $M \in \mathbb{R}^{n\times n}$ s'écrivant sous la forme $M = \begin{pmatrix} Q & S \\ S^\top & R \end{pmatrix}$ avec Q carrée.
On peut écrire :

$$\begin{aligned} M &= \begin{pmatrix} I & 0 \\ -F & I \end{pmatrix}^\top \begin{pmatrix} I & 0 \\ F & I \end{pmatrix}^\top M \begin{pmatrix} I & 0 \\ F & I \end{pmatrix} \begin{pmatrix} I & 0 \\ -F & I \end{pmatrix} \\ &= \begin{pmatrix} I & 0 \\ -F & I \end{pmatrix}^\top \begin{pmatrix} Q + F^\top S^\top + SF + F^\top RF & S + F^\top R \\ S^\top + RF & R \end{pmatrix} \begin{pmatrix} I & 0 \\ -F & I \end{pmatrix} \end{aligned}$$

La matrice M est définie positive si et seulement si le 2$^{\text{e}}$ terme de la dernière équation est défini positif. Si M est définie positive, alors R est inversible, et avec $F = -R^{-1}S^\top$, on a :

$$M > 0 \iff \begin{pmatrix} Q - SR^{-1}S^\top & 0 \\ 0 & R \end{pmatrix} > 0$$

($Q - SR^{-1}S^\top$ est appelé ***complément de Schur*** de R dans M).
D'où $M > 0 \iff \begin{cases} R > 0 \\ Q - SR^{-1}S^\top > 0 \end{cases}$

On peut donc conclure que l'ensemble des inégalités de la forme de 2.2 peuvent s'écrire sous une LMI. Cette proposition nous permet d'étendre le nombre de problèmes qui peuvent se mettre sous une forme LMI.

• Une inégalité portant sur la norme d'une matrice peut aussi s'écrire sous forme LMI :
Soit $Z : \mathbb{V} \to \mathbb{R}^{p\times q}$ une fonction affine. On a :

$$\begin{aligned} \forall x \in \mathbb{V}, \quad \|Z(x)\| < 1 &\iff I - Z(x)Z(x)^\top > 0 \\ &\iff \begin{pmatrix} I & Z(x) \\ Z(x)^\top & I \end{pmatrix} > 0 \end{aligned}$$

• Il est possible d'éliminer dans certains cas des variables grâce au lemme d'élimination [5] :

Proposition 3 (Lemme d'élimination)
Soit la LMI suivante :

$$A(x) + B(x)XC^\top(x) + C(x)X^\top B^\top(x) > 0$$

Il est possible d'éliminer la variable X si x et X sont indépendants. La LMI est alors équivalente à :

$$\begin{cases} \tilde{B}^\top(x)A(x)\tilde{B}(x) & > 0 \\ \tilde{C}^\top(x)A(x)\tilde{C}(x) & > 0 \end{cases}$$

où $\tilde{B}$ et $\tilde{C}$ sont les compléments orthogonaux de B et C respectivement. ($\tilde{M}$ est le complément orthogonal de M si et seulement si $M\tilde{M} = 0$ et $M\tilde{M}^\top$ est de rang maximal)

On pourra trouver une preuve de ce lemme dans [2].

• Il est aussi possible d'éliminer une variable d'une LMI lorsque celle-ci comporte un terme semi-défini.
Soit le problème LMI :

$$\text{Trouver } x \in \mathbb{R}^m \text{ tel que } F(x) = F_0 + \sum_{i=1}^{m} x_i F_i > 0 \tag{2.3}$$

On suppose que l'on a $F_m \geq 0$ et que F_m est de rang $r \leq n$.
Nous allons montrer que ce problème est équivalent à un problème LMI avec $m-1$ variables.
F_m est semi-définie, donc il existe une matrice U de rang plein vérifiant $F_m = UU^\top$. Nous avons $F(x) > 0$ si et seulement si

$$\begin{pmatrix} \tilde{U}^\top \\ U^\top \end{pmatrix} F(x) \begin{pmatrix} \tilde{U} & U \end{pmatrix} > 0$$

On définit

$$\hat{x} = (x_1, \ldots, x_{m-1})^\top \in \mathbb{R}^{m-1}$$

et

$$\hat{F}(\hat{x}) \triangleq F_0 + \sum_{i=1}^{m-1} x_i F_i$$

On a alors $F(x) = \hat{F}(\hat{x}) + x_m UU^\top$, donc

$$\begin{pmatrix} \tilde{U}^\top \\ U^\top \end{pmatrix} F(x) \begin{pmatrix} \tilde{U} & U \end{pmatrix} = \begin{pmatrix} \tilde{U}^\top \hat{F}(\hat{x})\tilde{U} & \tilde{U}^\top \hat{F}(\hat{x})U \\ U^\top \hat{F}(\hat{x})\tilde{U} & U^\top \hat{F}(\hat{x})U + x_m (U^\top U)^2 \end{pmatrix}$$

car $U^\top \tilde{U} = 0$
En applicant le lemme de Schur, on a $F(x) > 0$ si et seulement si :

$$\begin{cases} \tilde{U}^\top \hat{F}(\hat{x})\tilde{U} > 0 \\ U^\top \hat{F}(\hat{x})U + x_m \left(U^\top U\right)^2 - U^\top \hat{F}(\hat{x})\tilde{U} \left(\tilde{U}^\top \hat{F}(\hat{x})\tilde{U}\right)^{-1} \tilde{U}^\top \hat{F}(\hat{x})U > 0 \end{cases}$$

Or, on peut choisir x_m suffisamment grand pour avoir la 2^e inégalité satisfaite. Ainsi, le problème défini en 2.3 est équivalent à :

$$\text{Trouver } \hat{x} \in \mathbb{R}^{m-1} \text{ tel que } \tilde{U}^\top \hat{F}(\hat{x}) \tilde{U} > 0$$

On a donc pu éliminer une variable du problème LMI grâce au terme semi-défini.

2.4 LMI et convexité

Une LMI $F(x) = F_0 + \sum_{i=0}^{m} x_i F_i > 0$ définit une contrainte convexe. Il est alors possible de définir un problème d'optimisation convexe où la contrainte est une LMI. C'est d'ailleurs souvent le cas dans différents problèmes relatifs à la commande.

On rencontre principalement trois problèmes d'optimisation convexe [6] (ou problèmes proches) formulés sous forme LMI (on considèrera ici trois LMI F, G et H)

- **Problème de faisabilité**

  $$\text{Trouver une solution } x \in \mathbb{R}^m \text{ à la LMI } F(x) > 0$$

- **Problème de minimisation d'un objectif linéaire**

  $$\min_{x \in \Omega} C^\top x \quad \text{avec} \quad \Omega = \{x \mid F(x) > 0\}$$

- **Problème de valeur propre généralisé**

  $$\min_{(x,\lambda) \in \Omega} \lambda \quad \text{avec} \quad \Omega = \left\{ (x, \lambda) \in \mathbb{R}^m \times \mathbb{R} \mid \left\{ \begin{array}{l} \lambda F(x) - G(x) > 0 \\ F(x) > 0 \\ H(x) > 0 \end{array} \right. \right\}$$

On peut tout de même noter que ce problème est quasi-convexe seulement.

Dans la plupart des cas, ces problèmes n'ont pas de solutions analytiques, mais, du fait de leur convexité, ils peuvent être résolus numériquement par des algorithmes d'optimisations (chapitre 4).

Chapitre 3

Différents problèmes pouvant se formuler sous forme LMI

3.1 Stabilité

On considère le système défini par :

$$\dot{x}(t) = Ax(t), \quad A \in \mathbb{R}^{n \times n} \tag{3.1}$$

Lyapunov a montré que ce système est asymptotiquement stable (toutes les fonctions $x : \mathbb{R} \to \mathbb{R}^n$ qui satisfont 3.1 vérifient $\lim_{t \to \infty} x(t) = 0$) si et seulement si il existe une matrice $P \in \mathbb{R}^{n \times n}$ symétrique telle que :

$$\begin{cases} P > 0 \\ A^\top P + PA < 0 \end{cases}$$

Ceci est bien sûr équivalent à la LMI (matricielle) :

$$\begin{pmatrix} P & 0 \\ 0 & -A^\top P - PA \end{pmatrix} > 0$$

3.2 Gain statique avec saturation

On considère le système linéaire

$$\dot{x} = Ax + Bu$$

avec $A \in \mathbb{R}^{n \times n}$ et $B \in \mathbb{R}^{n \times m}$.
Il existe un gain statique stabilisant K tel que, pour tout condition initiale $x(0)$ vérifiant $\|x(0)\| \leq 1$, la commande $u = Kx$ vérifie $\|u(t)\| \leq u_{\max}, \forall t \geq 0$

si et seulement si il existe $Q \in \mathbb{R}^{n\times n}$ et $Y \in \mathbb{R}^{m\times n}$ qui vérifient les conditions suivantes [7] :

$$Q = Q^\top$$

et

$$\begin{array}{rcl} Q & \geq & 0 \\ -AQ - QA^\top - BY - Y^\top B^\top & > & 0 \\ \begin{pmatrix} u_{\max}^2 I & Y \\ Y^\top & Q \end{pmatrix} & \geq & 0 \end{array}$$

3.3 Lemme Borné Réel et Positif Réel

Ces deux lemmes importants en synthèse H_∞ peuvent se reformuler grâce aux LMI (on pourra retrouver une synthèse de ces lemmes dans [8] et [5].

Proposition 4 (Lemme Positif Réel)
Le système

$$\begin{array}{rcl} \dot{x}(t) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) + Du(t) \end{array}$$

est passif, i.e.

$$\int_0^T u^\top(t) y(t)\, dt \leq 0$$

pour toute solution x telle que $x(0) = 0$. Les propositions suivantes sont équivalentes à la passivité :

* *la fonction de transfert $G(s) = C(sI - A)^{-1}B + D$ est positive réelle,* i.e.

$$\forall s \mid \Re(s) > 0,\ G(s) + \bar{G}(s) \geq 0$$

* *il existe une solution P symétrique définie positive à l'inéquation de Riccati :*

$$A^\top P + PA + (PB - C^\top)(D + D^\top)^{-1}(PB - C^\top)^\top \leq 0$$

* *la LMI suivante admet une solution en P :*

$$P > 0$$

$$\begin{pmatrix} A^\top P + PA & PB - C^\top \\ B^\top P - C & -D^\top - D \end{pmatrix} \leq 0$$

Proposition 5 (Lemme Borné Réel)
Le système

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

est non expansif, i.e.

$$\int_0^T y^\top(t)y(t)\ dt \leq \int_0^T u^\top(t)u(t)\ dt$$

pour toute solution x telle que $x(0) = 0$.
Les propositions suivantes sont équivalentes (avec $\gamma > 0$) :

*

$$\int_0^T y^\top(t)y(t)\ dt \leq \gamma \int_0^T u^\top(t)u(t)\ dt$$

* *la matrice de transfert $G(s) = C(sI - A)^{-1}B + D$ vérifie*

$$\|G(s)\|_\infty \leq \gamma$$

* *la LMI suivante admet une solution en P :*

$$P > 0$$

$$\begin{pmatrix} A^\top P + PA + \frac{1}{\gamma}C^\top C & PB + \frac{1}{\gamma}C^\top D \\ B^\top P - \frac{1}{\gamma}D^\top C & \frac{1}{\gamma}D^\top D - \gamma I \end{pmatrix} \leq 0$$

* *la LMI suivante admet une solution en P :*

$$P > 0$$

$$\begin{pmatrix} A^\top P + PA & PB & C^\top \\ B^\top P & -\gamma I & D^\top \\ C & D & -\gamma I \end{pmatrix} < 0$$

3.4 Calcul des normes H_2 et H_∞

On considèrera un système G (bouclé ou non bouclé) :

$$G = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$$

3.4.1 Norme H_2

La norme H_2 d'un tel système est donnée par

$$\begin{aligned} \|G(s)\|_2 &= \sqrt{tr(CP_BC^\top)} \\ &= \sqrt{tr(B^\top P_C B)} \end{aligned}$$

où P_C et P_B sont les grammiens de commandabilité et d'observabilité définis par :

$$P_C = \int_0^\infty e^{A\tau} BB^\top e^{A^\top \tau}\, d\tau$$

$$P_B = \int_0^\infty e^{A^\top \tau} C^\top C e^{A\tau}\, d\tau$$

qui sont solutions des équations :

$$AP_C + P_C A^\top + BB^\top = 0$$

$$A^\top P_B + P_B A + C^\top C = 0$$

On peut aussi écrire

$$\begin{aligned} \|G(s)\|_2 &= \sqrt{\inf_{P_C = P_C^\top} \{trace\,(CP_BC^\top) \mid AP_C + P_C A^\top + BB^\top < 0\}} \\ &= \sqrt{\inf_{P_B = P_B^\top} \{trace\,(B^\top P_B B) \mid A^\top P_B + P_B A + C^\top C < 0\}} \end{aligned}$$

Cette formulation se prête alors bien à une écriture sous forme LMI, en utilisant le lemme de Schur.

Proposition 6 (Calcul de norme H_2 sous forme LMI)
La norme H_2 d'un système (A, B, C, D) peut se calculer grâce à une formulation LMI :

$$\|G(s)\|_2^2 = \min_{R=R^\top, P=P^\top} tr(R) \text{ avec } \begin{cases} \begin{pmatrix} AP + PA^\top & B \\ B^\top & -I \end{pmatrix} < 0 \\ \\ \begin{pmatrix} R & CP \\ PC^\top & P \end{pmatrix} > 0 \end{cases}$$

ou

$$\|G(s)\|_2^2 = \min_{R=R^\top, Q=Q^\top} tr(R) \text{ avec } \begin{cases} \begin{pmatrix} A^\top Q + QA & C^\top \\ C & -I \end{pmatrix} < 0 \\ \\ \begin{pmatrix} R & B^\top Q \\ QB & Q \end{pmatrix} > 0 \end{cases}$$

3.4.2 Norme H_∞

Contrairement à la norme H_2, il n'existe pas de méthode explicite pour calculer la norme H_∞ d'un système linéaire, il faut utiliser des méthodes itératives.
En utilisant le Lemme Réel Borné (cf chapitre 5), on peut formuler une méthode de calcul de norme H_∞ en utilisant une formulation LMI.

Proposition 7 (Calcul de norme H_∞ sous forme LMI)
La norme H_∞ d'un sytème (A, B, C, D) peut être calculer grâce à une formulation LMI :

$$\|G(s)\|_\infty \leq \min_{X>0, X=X^\top} \gamma \text{ avec } \begin{pmatrix} A^\top X + XA & XB & C^\top \\ B^\top X & -\gamma I & D^\top \\ C & D & -\gamma I \end{pmatrix} < 0$$

3.5 Problème standard optimal

On considère le système suivant :

$$\left(\begin{array}{c} \dot{x}(t) \\ \hline e(t) \\ y(t) \end{array}\right) = \left(\begin{array}{c|cc} A & B_w & B_u \\ \hline C_e & D_{ew} & D_{eu} \\ C_y & D_{yw} & D_{yu} \end{array}\right) \left(\begin{array}{c} x(t) \\ \hline w(t) \\ u(t) \end{array}\right) = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \left(\begin{array}{c} x(t) \\ \hline w(t) \\ u(t) \end{array}\right)$$

avec $x \in \mathbb{R}^n$, $w \in \mathbb{R}^p$, $u \in \mathbb{R}^q$, $e \in \mathbb{R}^r$ et $y \in \mathbb{R}^s$.

La mise sous forme standard permet de sélectionner les entrées et les sorties que l'on veut contrôler, en y incorporant des fonctions de pondération fréquentielles qui reflètent les objectifs de performance et certains objectifs de robustesse.

On supposera que $D_{yu} = 0$[1], que la paire (A, B_w) est stabilisable et que la paire (C_y, A) est détectable.

[1]Dans le cas où le système n'est pas strictement propre, le terme D_{yu} pourra être réintroduit *a posteriori*

De plus, on notera $P(s)$ le système bouclé et $\left(\begin{array}{c|c} A_{cl} & B_{cl} \\ \hline C_{cl} & D_{cl} \end{array}\right)$ la matrice système associée.

Le **problème standard H_2, H_∞ optimal** est le suivant : quel correcteur $K(s)$ stabilise le système bouclé de la figure 3.1 et minimisent $\|F_l(P,K)\|_2$ ou $\|F_l(P,K)\|_\infty$.

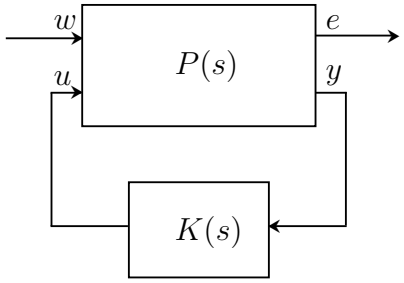

FIG. 3.1 – Problème standard

3.5.1 Problème H_∞ optimal

Supposons la matrice système du régulateur $K(s)$ donné par

$$K_s = \left(\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right)$$

Le système bouclé sera de la forme :

$$\left(\begin{array}{c|c} A_{cl} & B_{cl} \\ \hline C_{cl} & D_{cl} \end{array}\right) = \left(\begin{array}{cc|c} A + B_u D_K C_y & B_u C_K & B_w + B_u D_K D_{yw} \\ B_K C_y & A_K & B_K D_{yw} \\ \hline C_e + D_{eu} D_K C_y & D_{eu} C_K & D_{ew} + D_{eu} D_K D_{yw} \end{array}\right)$$

Le lemme réel borné nous permet d'affirmer que $\|F_l(P,K)\|_\infty < \gamma$ si et seulement si il existe une matrice X symétrique définie positive solution de l'inégalité matricielle :

$$\begin{pmatrix} A_{cl}^\top X + X A_{cl} & X B_{cl} & C_{cl}^\top \\ B_{cl}^\top X & -\gamma I & D_{cl}^\top \\ C_{cl} & D_{cl} & -\gamma I \end{pmatrix} < 0 \qquad (3.2)$$

Cette inégalité n'est pas une LMI en les variables X et K_s, car on retrouve des termes comportant le produit de ces deux variables. Il s'agit d'une BMI (cf chapitre 9). Le lemme d'élimination permet de retrouver un problème de

faisabilité LMI.
En supposant la paire (A, B_u) stabilisable et la paire (C_y, A) détectable, on a le résultat suivant :

Proposition 8 (Problème H_∞ optimal)
Le problème H_∞ standard admet une solution si et seulement si il existe deux matrices symétriques R et S satisfaisant le problème LMI suivant :

$$\min_{\gamma, R=R^\top, S=S^\top} \gamma$$

$$\begin{cases} \begin{pmatrix} \mathcal{N}_R & 0 \\ 0 & I_p \end{pmatrix}^\top \begin{pmatrix} AR + RA^\top & RC_e^\top & B_w \\ C_eR & -\gamma I_r & D_{ew} \\ B_w^\top & D_{ew}^\top & -\gamma I_p \end{pmatrix} \begin{pmatrix} \mathcal{N}_R & 0 \\ 0 & I_p \end{pmatrix} < 0 \\ \begin{pmatrix} \mathcal{N}_S & 0 \\ 0 & I_r \end{pmatrix}^\top \begin{pmatrix} A^\top S + SA & SB_w & C_e^\top \\ B_w^\top S & -\gamma I_p & D_{ew}^\top \\ C_e & D_{ew} & -\gamma I_r \end{pmatrix} \begin{pmatrix} \mathcal{N}_S & 0 \\ 0 & I_r \end{pmatrix} < 0 \\ \begin{pmatrix} R & I_n \\ I_n & S \end{pmatrix} \geq 0 \end{cases}$$

où $\mathcal{N}_R$ et $\mathcal{N}_S$ constituent les bases des noyaux de $\begin{pmatrix} B_u^\top & D_{eu}^\top \end{pmatrix}$ et $\begin{pmatrix} C_y & D_{yw} \end{pmatrix}$ respectivement.
Pour construire le correcteur, on déduit d'abord, de la solution du problème LMI précédent, une matrice X vérifiant (3.2). Une telle matrice X est donnée par

$$X = \begin{pmatrix} S & N \\ N^\top & -M^+RN \end{pmatrix}$$

où M et N sont deux matrices de rang plein telles que $RS + MN^\top = I$.
Ensuite, à X fixé, l'inégalité (3.2) devient une LMI en K_s, qu'il est possible de résoudre.

Toutefois,, dans le cas où une contrainte de structure du régulateur est introduite, il n'est plus possible d'utiliser le lemme d'élimination puisque la matrice X est indépendante de la structure de la matrice K_s, et le problème ne se réduit plus à un problème d'optimisation convexe sous formulation LMI.

3.5.2 Problème H_2 optimal

On cherche donc ici à trouver un régulateur K qui minimise la norme H_2 du système bouclé. Ce problème est équivalent à l'un ou l'autre des problèmes d'optimisation suivants :

$$\|G(s)\|_2^2 = \min_{R=R^\top, P=P^\top} tr(R) \text{ avec } \begin{cases} \begin{pmatrix} A_{cl}P + PA_{cl}^\top & B_{cl} \\ B_{cl}^\top & -I \end{pmatrix} < 0 \\ \\ \begin{pmatrix} R & C_{cl}P \\ PC_{cl}^\top & P \end{pmatrix} > 0 \end{cases}$$

ou

$$\|G(s)\|_2^2 = \min_{R=R^\top, Q=Q^\top} tr(R) \text{ avec } \begin{cases} \begin{pmatrix} A_{cl}^\top Q + QA_{cl} & C_{cl}^\top \\ C_{cl} & -I \end{pmatrix} < 0 \\ \\ \begin{pmatrix} R & B_{cl}^\top Q \\ QB_{cl} & Q \end{pmatrix} > 0 \end{cases}$$

Il est possible ensuite, suivant les différents cas (retour d'état, retour dynamique de sortie, ...), de faire apparaitre une LMI, en explicitant, en fonction de K_s les matrices A_{cl}, B_{cl}, C_{cl} et D_{cl}.

Retour d'état

Dans le cas du retour d'état, on a seulement $u = K_x x$, donc le système en boucle fermé s'écrit :

$$\begin{pmatrix} A_{cl} & B_{cl} \\ C_{cl} & D_{cl} \end{pmatrix} = \left(\begin{array}{c|c} A + B_u K_x & B_w \\ \hline C_e + D_{eu} K_x & 0 \end{array}\right)$$

Le problème H_2 standard est alors équivalent au problème d'optimisation :

$$\min_{R=R^\top, P=P^\top} tr(R)$$

$$\begin{cases} \begin{pmatrix} (A + B_u K_x)^\top P + P(A + B_u K_x) & (C_e + D_{eu} K_x)^\top \\ C_e + D_{eu} K_x & -I \end{pmatrix} < 0 \\ \\ \begin{pmatrix} R & B_w^\top P \\ PB_w & P \end{pmatrix} > 0 \end{cases}$$

Ce problème est un problème sous contrainte bi-linéaire en P et K_x. Mais il est possible de se ramener à un problème sous contrainte LMI par un

changement de variable.
On pose $W = P^{-1}$ et $Q = K_x P$ (P étant strictement définie positive, P est inversible). On transforme la première inégalité matricielle grâce au lemme de Schur, puis on multiplie l'expression obtenue à gauche et à droite par P^{-1}, puis on effectue le changement de variable. On retrouve, grâce au lemme de Schur, une inégalité matricielle, qui est maintenant linéair en W et Q

$$\begin{pmatrix} WA^\top + AW + Q^\top B_u^\top + B_u Q & (C_e W + D_{eu} Q)^\top \\ C_e W + D_{eu} Q & -I \end{pmatrix} < 0$$

Le changement de variable est aussi appliqué à la deuxième inégalité matricielle, après multiplication par $\begin{pmatrix} I & 0 \\ 0 & P^{-1} \end{pmatrix}$ à gauche et à droite.
Finalement, le problème H_2 avec retour d'état est équivalent au problème suivant :

$$\min_{R=R^\top, P=P^\top} tr(R)$$

$$\begin{cases} \begin{pmatrix} WA^\top + AW + Q^\top B_u^\top + B_u Q & (C_e W + D_{eu} Q)^\top \\ C_e W + D_{eu} Q & -I \end{pmatrix} < 0 \\ \\ \begin{pmatrix} R & B_w^\top \\ B_w & W \end{pmatrix} > 0 \end{cases}$$

Retour dynamique de sortie

Proposition 9 (Problème de synthèse H_2 par retour dynamique de sortie) *Le problème de synthèse H_2 par retour dynamique de sortie [9] est équivalent à :*

$$\min_{Y,X,W,L,F} tr(W)$$

$$\begin{cases} \begin{pmatrix} Y & I & B_w \\ I & X & XB_w + FD_{eu} \\ B_w^\top & B_w^\top X + D_{eu}^\top F^\top & W \end{pmatrix} > 0 \\ \\ \begin{pmatrix} R & YC_e^\top + L^\top D_{yw}^\top \\ C_e Y^\top + D_{yw} L & -I \end{pmatrix} < 0 \\ \\ \begin{pmatrix} S & C_e^\top \\ C_e & -I \end{pmatrix} < 0 \end{cases}$$

La construction du régulateur se fait alors par :

$$K(s) = L\left(s(I - XY) - M^\top\right)^{-1} F$$

avec $M = -Z - (YC_e^\top + L^\top D_{eu}^\top) C_e$.

Chapitre 4

Algorithmes et méthodes pour la résolution des problèmes LMI

Les solutions analytiques à ce type de problème n'existent que dans très peu de cas (et surtout avec des problèmes de petite taille). Nous présentons alors deux méthodes de résolutions algorithmiques qui ont permis l'essor de l'utilisation des problèmes LMI depuis les années 1980.

4.1 L'algorithme Ellipsoïde

Cet algorithme a été développé par Shor, Nemirpvskii et Yudin. Nous le présenterons dans le cas d'une optimisation de la fonction objectif convexe $f : \mathbb{V} \to \mathbb{R}$ sans contrainte, pour simplifier [10].

Définition 7 (sous-gradient)
Soit $f : \mathbb{V} \to \mathbb{R}$.
x^* *est un sous-gradient de* f *en* $x \in \mathbb{V}$ *si et seulement si :*

$$\forall\, y \in \mathbb{V},\ f(y) \geq f(x) + (y - x)\, x^*$$

Cette notion servira dans l'algorithme Ellipsoïde.
L'idée de cet algorithme réside en la génération d'une suite d'ellipsoïdes $(\mathcal{E}_i)_{i\in\mathbb{N}}$ de $\mathbb{V}$ contenant chacune l'optimum de f, et dont le volume converge vers 0.

Etape 0 :
On se donne $x_0 \in \mathbb{V}$ et $P_0 \in \mathbb{S}$ définie positive. Cela nous permet de définir

l'ellipsoïde $\mathcal{E}_0$:

$$\mathcal{E}_0 = \left\{ x \in \mathbb{V} \mid (x - x_0)^\top P_0^{-1} (x - x_0) \leq 1 \right\}$$

où x_0 précise le centre de l'ellipsoïde, et la matrice P_0 sa taille et son orientation. $\mathcal{E}_0$ doit contenir l'optimum pour assurer le fonctionnement de l'algorithme.

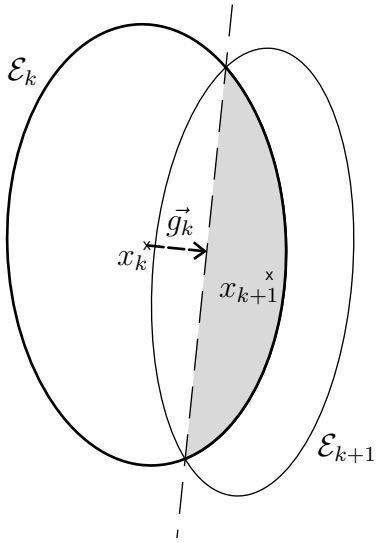

FIG. 4.1 – Itération de l'algorithme Ellipsoïde

Etape k+1 :

On va construire une nouvelle ellipsoïde $\mathcal{E}_{k+1}$ contenant l'optimum, à partir de l'ellipsoïde $\mathcal{E}_k$.

Si l'on considère le sous-gradient g_k de f au point x_k, celui-ci nous permet de partager $\mathbb{V}$ en deux demi-espaces, un contenant l'optimum, et un autre ne le contenant pas (l'ensemble $V = \left\{ x \in \mathbb{V} \mid g_k^\top (x - x_k) \leq 0 \right\}$ contient le minimum de f sur $\mathbb{V}$, par définition du sous-gradient).

On peut maintenant définir un ellipsoïde $\mathcal{E}_{k+1}$ contenant $\mathcal{E}_k \cap V$ (grisé sur la figure 4.1). x_{k+1} et P_{k+1} sont donnés par :

$$x_{k+1} \triangleq x_k - \frac{P_k g_k}{(m+1)\sqrt{g_k^\top P_k g_k}}$$

$$P_{k+1} \triangleq \frac{m^2}{m^2 - 1} \left(P_k - \frac{2}{(m+1) g_k^\top P_k g_k} P k g_k g_k^\top P_k \right)$$

où $m = \dim(\mathbb{V})$

On peut montrer que l'on a volume$(\mathcal{E}_k) \leq e^{-1/2m}$volume$(\mathcal{E}_0)$, ce qui nous assure la convergence numérique de la méthode en un temps polynomial. La preuve de cet algorithme peut être trouvée dans [2].

4.2 Méthodes des points intérieurs

Ces méthodes ont été développées par Nesterov et Nemirovskii vers la fin des années 80 [11]. Elles ont été rapidement adoptées, du fait de leur performance, et ont suscité de nombreuses améliorations. Ces méthodes reposent sur la transformation du problème convexe en problème non contraint par l'introduction d'une fonction *barrière* ϕ [6].

Définition 8 (Fonction barrière)
Une fonction barrière ϕ est une fonction $\mathbb{V} \to \mathbb{R}$ différentiable vérifiant les propriétés suivantes :

* *ϕ est strictement convexe sur la contrainte $\Omega = \{x | F(x) > 0\}$*
* *$\forall (x_n)_{n\in\mathbb{N}} \in \Omega^{\mathbb{N}}$ tel que $\lim_{n\to+\infty} x_n = x^* \in \partial\Omega$, alors on a*

$$\lim_{n\to+\infty} \phi(x_n) = +\infty$$

Si une telle fonction barrière existe, alors on peut transformer notre problème de minimisation sous contrainte non différentiable en problème d'optimisation différentiable non contraint, en minimisant sur $\mathbb{V}$ la fonction :

$$x \mapsto tf(x) + \phi(x)$$

t est appelé *paramètre de pénalisation.*

On peut montrer [2] que la fonction barrière suivante est une fonction possible pour la résolution du problème de faisabilité :

$$\phi(x) = \begin{cases} -\log(\det(F(x))) & \text{si } x \in \Omega \\ +\infty & \text{si } x \notin \Omega \end{cases}$$

Chapitre 5

Résolutions de problèmes LMI avec la *LMI Control Toolbox* pour MATLAB©

Matlab et Scilab représentent des outils universels de calcul numérique qui sont beaucoup employés en automatisme et en commande. Pour chacun d'entre eux, il existe plusieurs boîtes à outils basés sur différents algorithmes et différentes implémentations. On peut citer notamment :

- LMI Control ToolBox pour Matlab (basé sur l'algorithme des points intérieurs développé par Nesterov et Nemirovskii)
- LMITOOL [7] pour Scilab et Matlab (algorithme développé par Nesterov et Todd)
- SeDuMi, développé par F. Sturm, qui se révelle être parmi les plus performants des solveurs LMI

Nous allons traiter un problème numérique concret à partir de la LMI Control Toolbox pour Matlab.

5.1 Aperçu

La LMI Control Toolbox est une boîte à outils pour résoudre numériquement des problèmes généraux sous formulation LMI. Elle fournit des outils simples et performants pour la spécification et la manipulation de LMI, et un *solver* pour les trois problèmes LMI génériques.
Son but principal est de [12] :

- permettre une description directe d'une LMI, en définissant de manière naturelle chaque bloc

– donner accès au *solver* (pour une optimisation de code)
– faciliter la validation de résultats et la modification du problème

5.2 Premier exemple

5.2.1 Problème à résoudre

On cherche à résoudre le problème suivant [12] :

$$\min_{X \in \Omega} Tr(X)$$

avec

$$\Omega = \left\{ X \in \mathbb{R}^{3\times 3} \mid A^\top X + XA + XBB^\top X + Q < 0 \right\}$$

et

$$\begin{aligned} A &= \begin{pmatrix} -1 & -2 & 1 \\ 3 & 2 & 1 \\ 1 & -2 & -1 \end{pmatrix} \\ B &= \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \\ Q &= \begin{pmatrix} 1 & -1 & 0 \\ -1 & -3 & -12 \\ 0 & -12 & -36 \end{pmatrix} \end{aligned}$$

On réécrit d'abord le problème sous forme LMI, avant de pouvoir le résoudre numériquement :

$$\min_{X \in \Omega} Tr(X)$$

avec

$$\Omega = \left\{ X \in \mathbb{R}^{3\times 3} \mid \begin{pmatrix} A^\top X + XA + Q & XB \\ B^\top X & -I \end{pmatrix} < 0 \right\}$$

La résolution d'un problème LMI avec la LMI Control Toolbox se fait en plusieurs étapes :
– Initialisation de la LMI
– Déclaration des inconnues (taille et structure)
– Déclaration des coefficients des inégalités matricielles
– Finalisation de la LMI
– Résolution du problème associée à la contrainte LMI

5.2.2 Implémentation

- On initialise d'abord les matrices A, B et Q :

```
%initialisation des matrices
A=[-1 -2 1; 3 2 1; 1 -2 -1];
B=[1 0 1]';
Q=[1 -1 0; -1 -3 -12; 0 -12 -36];
```

- Puis on indique à la *LMI Control Toolbox* que l'on va commencer à décrire une LMI, en partant de zéro (on peut aussi partir d'une LMI existante, pour la modifier ou la compléter).

```
% mise en place de la contrainte sous formulation LMI
setlmis([])
```

- Ensuite, on indique avec la fonction `lmivar`[1] quelles sont les inconnues de ce problème, et de quel type elles sont (la LMI Control Toolbox permet de définir des inconnues matricielles qui ont différentes structures : définie par bloc, diagonale, symétrique, ...). Ici, l'inconnue est $X \in \mathbb{R}^{3\times 3}$, une matrice symétrique.

```
% inconnues
X=lmivar(1,[3 1]);
```

- On indique ensuite avec la fonction `lmiterm` les différents termes qui composent la LMI, un par un, selon s'il sont constants ou qu'ils dépendent de X.

```
% lmi
IQ=newlmi;
lmiterm([IQ 1 1 X],1,A,'s');     % XA+A'X'
lmiterm([IQ 1 1 0],Q);           % Q
lmiterm([IQ 1 2 X],1,B);         % XB
lmiterm([IQ 2 2 0],-1);          % -I
```

- En dernier point, avant de lancer la résolution numérique, on indique que la définition du problème LMI est terminée, et on indique la fonction objectif.

```
% fin de la paramétrisation du problème
LMI=getlmis;

% fonction objectif
obj=mat2dec(LMI,eye(3));         % objectif : trace(X)
```

[1] Nous détaillerons son fonctionnement dans le chapitre 5.3

5.2.3 Solution

On peut ensuite trouver une solution numérique à ce problème :

```
options=[1e-5,0,0,0,0];
[opj_opt,res]=mincx(LMI,obj,options);

X_opt=dec2mat(LMI,res,X)
```

Le résultat est le suivant :

```
 Solver for linear objective minimization under LMI constraints

 Iterations    :    Best objective value so far

     1
     2                  -8.511476
     3                 -13.063640
***                  new lower bound:   -34.023978
     4                 -15.768450
***                  new lower bound:   -25.005604
     5                 -17.123012
***                  new lower bound:   -21.306781
     6                 -17.882558
***                  new lower bound:   -19.819471
     7                 -18.339853
***                  new lower bound:   -19.189417
     8                 -18.552558
***                  new lower bound:   -18.919668
     9                 -18.646811
***                  new lower bound:   -18.803708
    10                 -18.687324
***                  new lower bound:   -18.753903
    11                 -18.705715
***                  new lower bound:   -18.732574
    12                 -18.712175
***                  new lower bound:   -18.723491
    13                 -18.714880
***                  new lower bound:   -18.719624
    14                 -18.716094
***                  new lower bound:   -18.717986
    15                 -18.716509
```

```
***                 new lower bound:   -18.717297
    16                 -18.716695
***                 new lower bound:   -18.716873

 Result:  feasible solution of required accuracy
          best objective value:   -18.716695
          guaranteed relative accuracy: 9.50e-006
          f-radius saturation:  0.000% of R = 1.00e+009


X_opt =

   -6.3542   -5.8895    2.2046
   -5.8895   -6.2855    2.2201
    2.2046    2.2201   -6.0771
```

En 16 itérations, nous trouvons un X_{opt}, avec une précision relative sur la valeur optimale de l'objectif de $9,5.10^{-6}$.

5.3 Les principales fonctions de la LMI Control Toolbox

Pour pouvoir comprendre un peu plus comment on peut se servir de la LMI Control Toolbox, nous allons expliquer une à une les principales fonctions. Toutefois, pour plus de détails, il faut se rapporter à [12].

5.3.1 Représentation d'une LMI

La LMI Control Toolbox utilise un objet de type `LMISYS` pour représenter une LMI. Cette représentation interne permet de manipuler des LMI de la forme :

$$\mathcal{N}^\top L(X_1,\ldots,X_k)\mathcal{N} < \mathcal{M}^\top R(X_1,\ldots,X_k)\mathcal{M}$$

où

- $X_1,\ldots,X_k$ sont les matrices inconnues, qui peuvent avoir des structures particulières (matrices symétriques, diagonales, etc.)
- L et R sont des matrices par bloc symétriques, dont la structure de chaque bloc est identique, et dont chaque bloc est composé de combinaisons affines de $X_1,\ldots,X_k$ et leur transposées
- $\mathcal{N}$ et $\mathcal{M}$ sont les facteurs extérieurs, de dimension identique

Cette représentation interne comprend donc les inconnues de cette LMI ainsi que leur forme, et les différents termes de celle-ci.
Une description d'une LMI doit commencer avec `setlmis` et finir avec `getlmis`. La fonction `setlmis` initialise la description de la LMI. Pour spécifier une nouvelle LMI, on tape :

```
setlmis([]);
```

Et pour partir d'une LMI `LMISO` existante, on tape :

```
setlmis(LMISO);
```

Quand la description est terminée, on récupère la représentation interne `LMISYS`, que l'on peut alors utiliser avec d'autres fonctions de LMI-Lab :

```
LMI=getlmis;
```

5.3.2 Déclaration des inconnues

Les variables matricielles (les inconnues de notre LMI) doivent être déclarées avec `lmivar` et caractérisées par leur structure.
La syntaxe générale de la fonction `lmivar` qui permet de déclarer une inconnue est :

```
X=lmivar(type,struct);
```

Les inconnues peuvent être de trois types :

- `type=1` : Matrice symétrique diagonale par bloc.

$$X = \begin{pmatrix} D_1 & 0 & \dots & 0 \\ 0 & D_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & D_r \end{pmatrix}$$

  où chaque bloc D_i est carré et est ou bien une matrice nulle, ou bien une matrice scalaire ou bien une matrice symétrique pleine
- `type=2` : Matrice rectangulaire quelconque.
- `type=3` : Matrices particulières. Ce troisième type permet de décrire indépendamment chaque coefficient de l'inconnue matricielle. Ils peuvent être égaux à 0, x_n ou $-x_n$, où x_n est la $n$ième variable de décision (scalaire) de la LMI.

Pour chacun de ces types, `struct` indique la structure particulière de `X` :

- Pour les matrices de type 1 (symétrique diagonale par bloc), `struct` est une matrice de 2 colonnes, et avec autant de lignes que de blocs diagonaux. La première indique la taille du bloc diagonale, et la seconde la nature du bloc :

-1	$\rightarrow$	bloc nul
0	$\rightarrow$	bloc scalaire
1	$\rightarrow$	bloc symétrique plein

- Pour les matrices de type 2 (matrices rectangulaires pleines), `struct` est un vecteur à 2 entiers, représentant la taille (nombre de lignes et nombre de colonnes) de la matrice
- Pour les matrices de type 3 (matrice particulière), `struct` est une matrice de même taille que le bloc à décrire, et chacun de ses coefficient le numéro d'une variable de décision scalaire. Par exemple, les matrices 3×3 de Toeplizt :

  $$Y = \begin{pmatrix} y_1 & y_2 & y_3 \\ y_2 & y_1 & y_2 \\ y_3 & y_2 & y_1 \end{pmatrix}$$

  sont définies par
  `Y=lmivar(3,n+[1 2 3;2 1 2;3 2 1]);`
  où n est le nombre de variables de décisions déjà déclarées dans cette LMI.

5.3.3 Déclaration des LMI

Nous allons maintenant déclarer chaque terme de chaque inégalité matricielle affine (nous ne spécifierons que les termes de la diagonale supérieure, ou inférieure ; la matrice F étant symétrique, cela est suffisant).
Une LMI est composée de :

- termes constants
- variables, c'est à dire de la forme PXQ ou $PX^\top Q$, où P et Q sont des matrices données constantes, et X une variable matricielle, déclarée au préalable avec `lmivar`
- facteurs extérieurs (constants)

Chaque terme d'une LMI est spécifié un à un, en utilisant `lmiterm`, en spécifiant à quel LMI et à quel bloc de la LMI il se réfère. Pour spécifier un bloc de LMI composé de sommes de plusieurs termes, on indique tout simplement un à un les termes, et le numéro de bloc ; les termes se référant au même bloc sont alors additionnés.

Pour commencer, il faut indiquer que l'on va rentrer les coefficients d'une nouvelle LMI avec `newlmi`. Cette fonction renvoie le numéro de la LMI créée,

qui nous servira pour remplir ses coefficients.

```
LMI1=newlmi;
```

Puis on indique un par un les termes de cette LMI avec `lmiterm` :

- pour un terme constant :

  ```
  lmiterm([N i j 0],P) ;
  ```

  avec :

  * `N` : numéro de la LMI (donné par `newlmi` par exemple, s'il s'agit du terme à gauche de l'inégalité $<$, l'opposé de ce numéro, s'il s'agit d'un terme à droite de l'inégalité
  * `i j` : coordonnées du bloc de la LMI
  * `P` : terme constant (une matrice, ou un scalaire dans le cas d'une matrice proportionnelle à la matrice identité (dans ce cas, la taille de la matrice est donnée par le contexte)

- pour des termes dépendants d'une inconnue, de la forme PXQ :

  ```
  lmiterm([N i j X],P,Q) ;
  ```

  avec :

  * `N` : numéro de la LMI
  * `i j` : coordonnées du bloc
  * `X` : indice représentant l'inconnue matricielle X. Cet indice est renvoyé par la fonction `lmivar` lors de la création de l'inconnue X
  * `P` et `Q` : matrices constantes. Si le paramètre `Q` est omis, cela correspond à $Q = I$

- pour les termes dépendants d'une inconnue, de la forme $PX^\top Q$:

  ```
  lmiterm([N i j -X],P,Q) ;
  ```

  * `X` : indice de l'inconnue X

- pour les termes dépendants d'une inconnue, de la forme $PXQ+Q^\top X^\top P^\top$:

  ```
  lmiterm([N i j X],P,Q,'s') ;
  ```

5.3.4 Déclaration de la fonction objectif

La fonction objectif doit être déclarée sous la forme d'un objectif linéaire : $C^\top x$ où $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ est le vecteur d'inconnues (variables de décisions, scalaires).
Les fonctions `mincx`, `feasp` et `gevp`, permettant de résoudre les trois principaux problèmes LMI, admettent comme paramètre le vecteur C.
Comme souvent les inconnues des LMI sont matricielles, il est utile de faire

appel à la fonction `mat2dec` qui permet de passer d'une représentation à une autre. Ainsi, à partir d'une LMI et de valeurs pour les inconnues matricielles, on peut retrouver un vecteur d'inconnues scalaires.
Par exemple, si l'on veut minimiser la trace d'inconnue matricielle X, C devra sélectrionner la diagonale de X, et cela s'obtient en faisant $X = I$. Le vecteur d'inconnues scalaires représentant la fonction objectif et servant aux fonctions `mincx`, `feasp` et `gevp` s'obtient ainsi :

```
c=mat2dec(LMI,eye(3));
```

De plus, la fonction `defcx` permet tout de même d'exprimer la fonction objectif en utilisant les inconnues matricielles (cf [12]).

5.3.5 Les *solvers*

Trois *solvers* sont disponibles dans la LMI Control Toolbox, correspondant aux trois problèmes principaux rencontrés (cf chapitre 2.4)

- Problème de faisabilité : le solver correspondant est `feasp`
- Minimisation d'un objectif linéaire : le solver adéquat est `mincx`
- Problème de valeurs propres généralisées : `gevp`

Pour chacune de ces fonctions, il est possible de donner les options de résolutions, telles que le nombre maximum d'itérations, le rayon de faisabilité (une valeur R telle que la solution x vérifie $x^\top x < R^2$, ce qui permet d'empêcher d'avoir des solutions avec une norme très grande), la précision relative sur la valeur optimale de l'objectif, etc.

5.4 Deuxième exemple

5.4.1 Problème à résoudre

On cherche deux matrices symétriques $X \in \mathbb{R}^{6\times 6}$ et $S \in \mathbb{R}^{4\times 4}$, avec S de la forme :

$$S = \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_2 & s_3 \\ 0 & 0 & s_3 & s_4 \end{pmatrix}$$

tels que

$$\begin{aligned} \begin{pmatrix} A^\top X + XA + C^\top SC & XB \\ B^\top X & -S \end{pmatrix} &< 0 \\ X &> 0 \\ S &> I \end{aligned}$$

5.4.2 Source

Ce problème peut se formuler de la façon suivante :

```
% initialisation des matrices
A=...
B=...
C=...

% mise en place de la contrainte LMI
setlmis([]);

% déclaration des inconnues
X=lmivar(1,[6 1]);              % X est une matrice symétrique 6x6 quelconque
S=lmivar(1,[2 0;2 1]);          % S est une matrice avec 2 blocs diagonaux
                                % le 1er est scalaire, le 2eme symétrique plein

% 1ere LMI
IQ1=newlmi;
lmiterm([IQ1 1 1 X],1,A,'s');   % A'X+XA
lmiterm([IQ1 1 1 S],C',C);      % C'SC
lmiterm([IQ1 1 2 X],1,B);       % XB
lmiterm([IQ1 2 2 S],-1,1);      % -S

% 2eme LMI
IQ2=newlmi;
lmiterm([-IQ2 1 1 X],1,1);      % 0<X (d'où le -IQ2)

% 3eme LMI
IQ3=newlmi;
lmiterm([-IQ3 1 1 S],1,1);      % S
lmiterm([IQ3 1 1 0],1);         % I

% fin de la paramétrisation
LMI=getlmis;
```

5.5 L'éditeur de LMI

Il existe une autre façon d'écrire un système d'équations LMI avec MATLAB Control Toolbox. Il s'agit d'utiliser l'éditeur de LMI (disponible grâce à la commande `lmiedit`). Celui-ci est constitué d'une interface graphique

qui permet de spécifier les LMI comme des expressions matricielles symboliques. Cela permet une saisie de manière plus intuitive et efficace qu'avec les commandes `lmiterm` et `lmivar`, mais ces derniers sont plus puissants et flexibles.

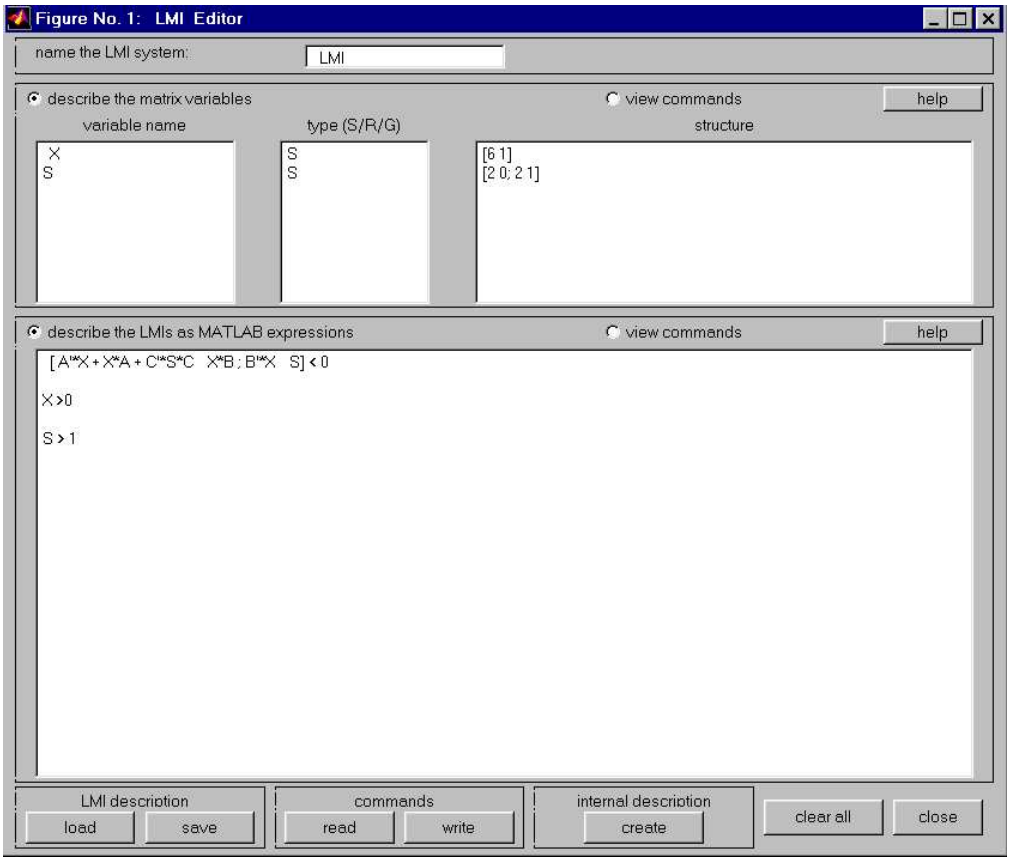

FIG. 5.1 – Interface graphique `lmiedit`

Chapitre 6

Conclusion

Dans la théorie de la commande, les LMI jouent, depuis quelques années, un rôle important : elles permettent une formulation unifiée de bon nombre de problèmes de commande des systèmes linéaires, en les transformant en problème d'optimisation sous contraintes convexes. Elles constituent à ce titre un outil central en automatique, et ce, depuis l'avènement d'algorithmes performants permettant leur résolution numérique.
Il existe cependant des problèmes de commande tout à fait importants d'un point de vue pratique, tels que les problèmes de commande H_2 ou H_∞ à ordre et à structure fixés, la commande multimodèle, le dimensionnement optimisé de systèmes, ...
Ces problèmes font intervenir une classe plus générale d'inégalités matricielles : les inégalités matricielles bilinéaires :

Définition 9 (Bilinear Matrix Inequality)
Une BMI (ou Inégalité Matricielle Biaffine) est une contrainte de la forme :

$$F(x,y) \triangleq F_{0,0} + \sum_{i=1}^{p} x_i F_{i,0} + \sum_{j=1}^{q} y_j F_{0,j} + \sum_{i=1}^{p}\sum_{j=1}^{q} x_i y_j F_{i,j} \; > \; 0 \qquad (6.1)$$

où

* $x = (x_1, \ldots, x_p) \in \mathbb{R}^p$, $y = (y_1, \ldots, y_q) \in \mathbb{R}^q$ *sont les variables de décisions*
* *les* $F_{i,j}$ *pour* $1 \leq i \leq p$ *et* $1 \leq j \leq q$ *sont des matrices symétriques de* $\mathbb{R}^{n \times n}$

On distingue là aussi deux types de problèmes :

- **Problème de faisabilité**
  Trouver une solution $(x, y) \in \mathbb{R}^p \times \mathbb{R}^q$ à la BMI $F(x, y) > 0$

- **Problème de minimisation d'un objectif linéaire sous contraine BMI**

$$\max_{(x,y)\in\Omega} C_x^\top x + C_y^\top y \text{ avec } \Omega = \{(x,y) \in \mathbb{R}^p \times \mathbb{R}^q \mid F(x,y) > 0\}$$

Notons qu'une LMI est une BMI particulière (avec $F_{i,j} = 0$, pour $1 \leq i \leq p$ et $1 \leq j \leq q$). De plus, certaines BMI particulières peuvent être transformées en LMI par changement d'inconnues ou par élimination de variables.

Dans le cas général, un problème de faisabilité BMI correspond à un problème de minimisation d'une fonction biconvexe sur $\mathbb{R}^p \times \mathbb{R}^q$, pour lequel il n'existe pas de méthode générale (les problèmes BMI appartiennent à la classe des systèmes NP-difficile, donc il n'existe pas d'algorithme polynomial de résolution des BMI), mais il est possible d'obtenir des solutions approchées [10].

Annexe A

Exemples de problème LMI ou BMI

A.1 1erexemple

Le problème est le suivant :
On dispose d'un système G de 2ème ordre

$$G(s) = \frac{g\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2}$$

Et d'un régulateur K de structure PID :

$$K(s) = \frac{1}{T_i s} + K_p$$

On supposera connu les coefficients T_i et K_p de ce régulateur (par placement de pôle du système Modèle-Régulateur).
On dispose d'un modèle de référence M_{ref} et l'on désire rajouter un préfiltre à ce processus régulé afin d'obtenir un modèle le plus proche (au sens de la norme H_∞) du modèle de référence).

On impose un préfiltre P de la forme :

$$P(s) = \frac{b}{s} + d$$

On cherche donc à résoudre le problème suivant :

$$\min_{b,d} \|M(s) - M_{ref}(s)\|_\infty$$

On calcule $M(s)$:

$$\begin{aligned} M(s) &= \frac{G(s)P(s)}{1+K(s)G(s)} \\ &= \frac{g\omega_n(ds+b)}{s^3+2\zeta\omega_n s^2+(1+gK_p)\omega_n^2 s+\frac{g\omega_n^2}{T_i}} \end{aligned}$$

On peut voir d'ailleurs, que, si l'on dispose de pôles x, y et z, vérifiant $x+y+z=-2\zeta\omega_n$, alors on peut identifier T_i et K_p :

$$T_i = \frac{-g\omega_n^2}{xyz} \quad ; \quad K_p = \frac{xy+xz+yz}{\omega_n^2 g} - \frac{1}{g}$$

On peut alors écrire notre système Préfiltre-Processus-Régulateur sous une forme d'état :

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{g\omega_n^2}{T_i} & -(1+gK_p)\omega_n^2 & -2\zeta\omega_n \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$
$$C = \begin{pmatrix} g\omega_n^2 b & g\omega_n^2 d & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 0 \end{pmatrix}$$

De la même manière, on peut définir les matrices d'état A_{ref}, B_{ref}, C_{ref} et D_{ref} du modèle de référence.

On peut donc en déduire les matrices d'états du modèle différence

$$\Delta M(s) = M(s) - M_{ref}(s)$$

de représentation d'état ΔA, ΔB, ΔC et ΔD. On a :

$$\Delta A = \begin{pmatrix} A & 0 \\ 0 & A_{ref} \end{pmatrix} \qquad \Delta B = \begin{pmatrix} B \\ B_{ref} \end{pmatrix}$$
$$\Delta C = \begin{pmatrix} C & -C_{ref} \end{pmatrix} \quad \Delta D = \begin{pmatrix} D & -D_{ref} \end{pmatrix}$$

Puisque le calcul de la norme H_∞ du modèle différence peut se traduire sous forme d'un problème convexe sous contrainte LMI, écrivons-le :

$$\|\Delta M(s)\|_\infty = \min_{\gamma, X>0, X=X^\top}$$

sous la contrainte

$$\begin{pmatrix} \Delta A^\top X + X\Delta A & X\Delta B & \Delta C^\top \\ \Delta B^\top X & -\gamma I & \Delta D^\top \\ \Delta C & \Delta D & -\gamma I \end{pmatrix} < 0$$

On décompose ΔC de la manière suivante :

$$\Delta C = C_1 + Y C_2$$

où

$$C_1 = \begin{pmatrix} 0 & 0 & 0 & -C_{ref} \end{pmatrix}$$

$$C_2 = \begin{pmatrix} g\omega_n^2 & 0 & 0 & 0_{C_{ref}} \\ 0 & g\omega_n^2 & 0 & 0_{C_{ref}} \end{pmatrix}$$

et où Y est une nouvelle inconnue matricielle de notre LMI
(Y s'écrit $Y = \begin{pmatrix} b & d \end{pmatrix}$).

Ainsi, pour trouver les paramètres b et d du préfiltre tels que la norme H_∞ du modèle différence soit la plus faible, il faut résoudre le problème sous contrainte LMI suivant :

$$\min_{\gamma, X=X^\top, X>0, Y} \gamma$$

$$\begin{pmatrix} \Delta A^\top X + X\Delta A & X\Delta B & C_1^\top + C_2^\top Y^\top \\ \Delta B^\top X & -\gamma I & \Delta D^\top \\ \Delta D & C_1 + YC_2 & -\gamma I \end{pmatrix} < 0$$

A.1.1 Cas 1

On prendra, dans les trois cas suivants, les valeurs numériques suivantes :

$$g = 1 \quad \omega_n = 1 \quad \zeta = 1$$

On considère ici le modèle de référence :

$$M_{ref} = \frac{1}{(s+0.5)(s+1)}$$

On règle alors K_p et T_i pour avoir un pôle en -0.5 et un en -1. On obtient donc

$$M(s) = \frac{ds + b}{(s+1)(s+0.5)^2}$$

On peut donc s'attendre ici, après optimisation du problème LMI, à optenir $b = 0.5$ et $d = 1$. Et c'est évidemment ce que l'on trouve (en 0.4s avec une précision de 10^{-8} sur un ordinateur de bureau)

A.1.2 Cas 2

De la même manière, si l'on considère le modèle de référence

$$M_{ref} = \frac{20}{(s+1)(s+0.5)^2}$$

, on retrouve, après optimisation LMI, des valeurs de b et d (ici $d = 0$ et $b = 20$) telles que le modèle obtenu est identique au modèle de référence, puisque cela est ici possible.

A.1.3 Cas 3

Un cas plus interessant est le cas suivant :

$$M_{ref} = \frac{0.1s^2 + 10s + 0.05}{(s+1)(s+0.5)^2}$$

En effet, l'ordre de ce modèle est trop grand pour que l'on puisse un réglage de b et d nous donnant un modèle identique à notre modèle de référence. Il s'agit donc bien, ici, de trouver le préfiltre permettant à notre modèle de se rapprocher le plus (au sens de la norme H_∞).
Nous trouvons les valeurs suivantes : $b = 0.0375$ et $d = 10$ Il est interessant de tracer les diagrammes de bode de ces deux modèles (référence et le modèle qui s'en approche) (figure A.5)

A.2 2^{e}exemple

On dispose d'un système G du 2^{e}ordre défini par

$$G(s) = \frac{g\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

On impose un régulateur K de type Proportionnel Intégral (figure A.2) :

$$K(s) = K_p + \frac{K_i}{s}$$

De plus, on considère une fonction de pondération W_τ paramétrée par τ :

$$W_\tau(s) = \frac{1 + \tau s}{\tau s}$$

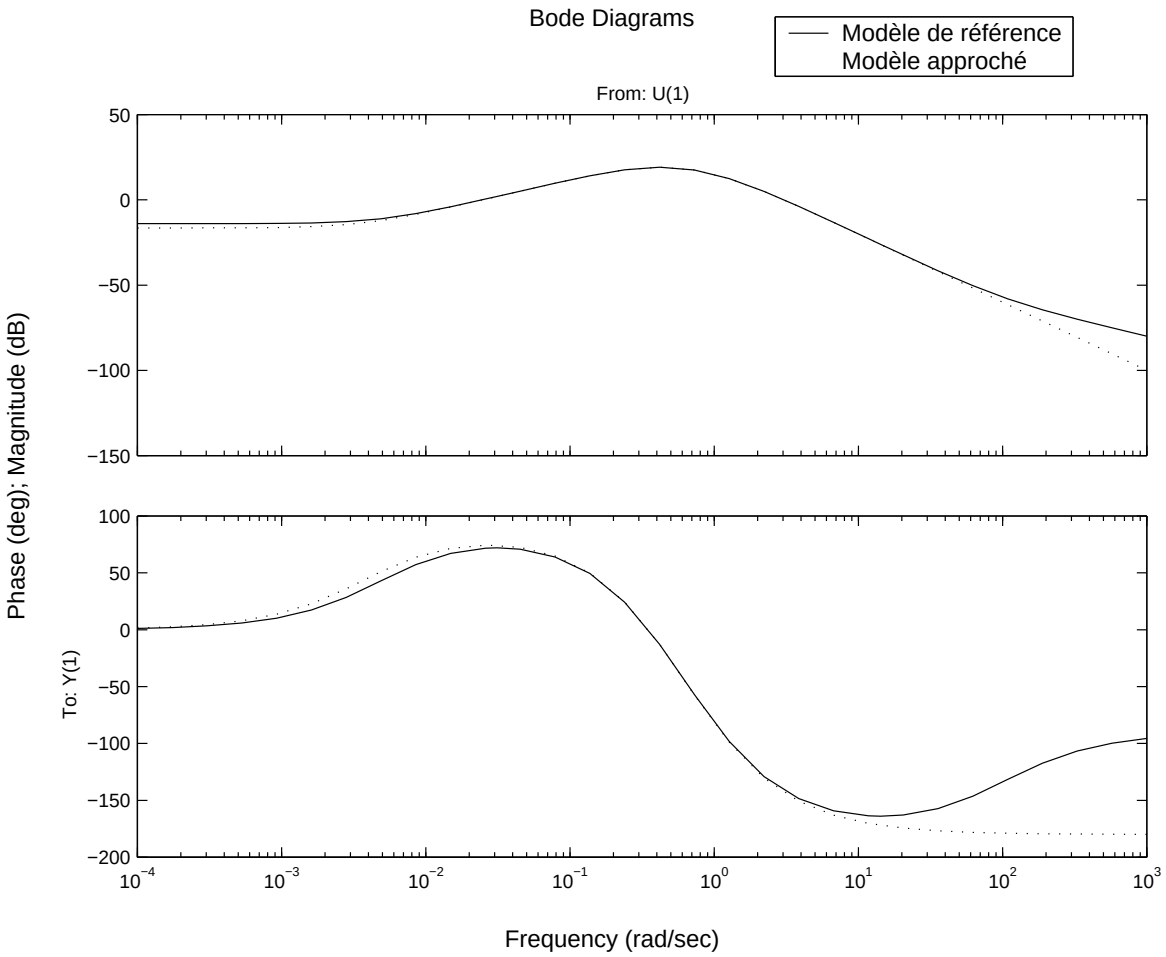

FIG. A.1 – Diagramme de Bode des deux modèles

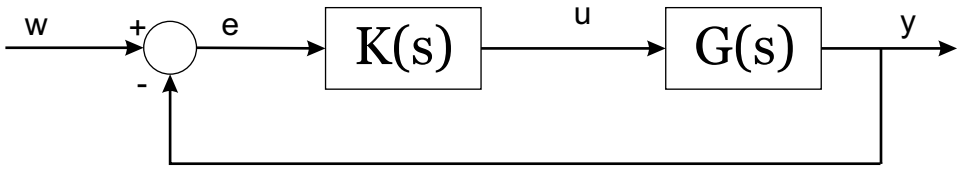

FIG. A.2 – Système régulé

On cherche alors le régulateur K stabilisant le système et qui minimise τ sous la contrainte

$$\|S(s)W(s)\|_\infty < a$$

($S(s)$ est la sensibilité ; il s'agit ici d'une contrainte sur la sensibilité pondérée).
On se donne les valeurs numériques suivantes :

$$g = 1,\ \omega_n = 1,\ \zeta = 0.1,\ a = 1 \text{ ou } a = 0.8$$

Pour pouvoir obtenir un problème se mettant sous forme LMI ou BMI, où K_p, K_i et τ sont des variables, il faut mettre le problème sous la forme suivante (figure A.3) :

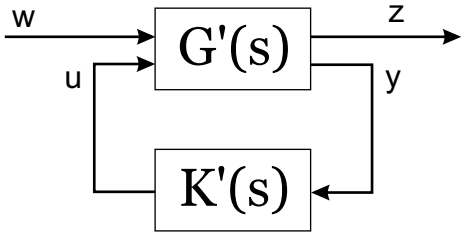

FIG. A.3 – Mise sous forme standard

où K' est seul fonction de K_p, K_i et τ. G' est le système G **augmenté**, pour y incorporer la sensibilité.
Malheureusement, comme celle-ci est pondérée par une fonction $W(s)$ qui dépend d'un paramètre que l'on doit optimiser, on doit directement utiliser son expression $W(s) = 1 + \chi\frac{1}{s}$, avec $\chi = \frac{1}{\tau}$.

On se ramène tout d'abord à un modèle augmenté dans lequel on a incorporé la fonction de pondération de la sensibilité :
Ensuite (figure **??**) :

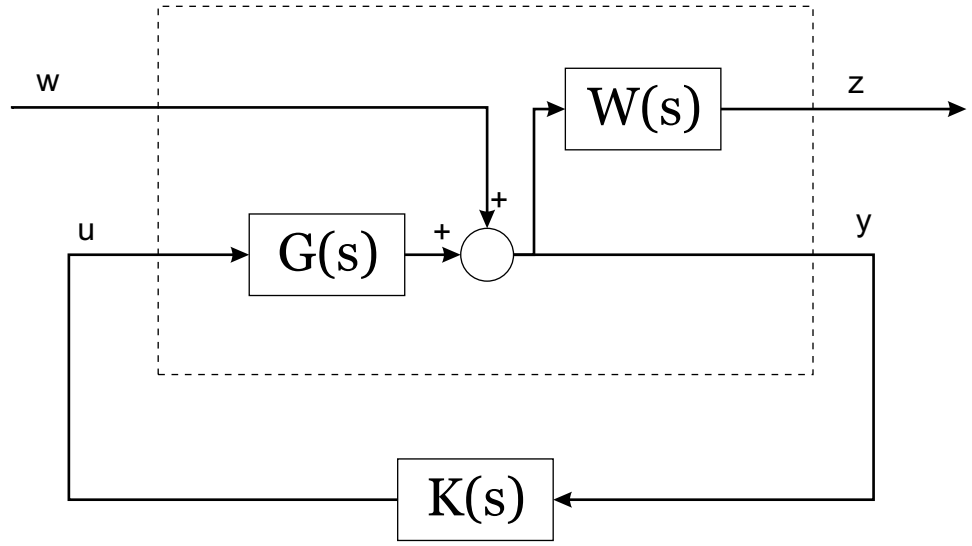

FIG. A.4 – Modèle augmenté, avec la sensibilité pondérée

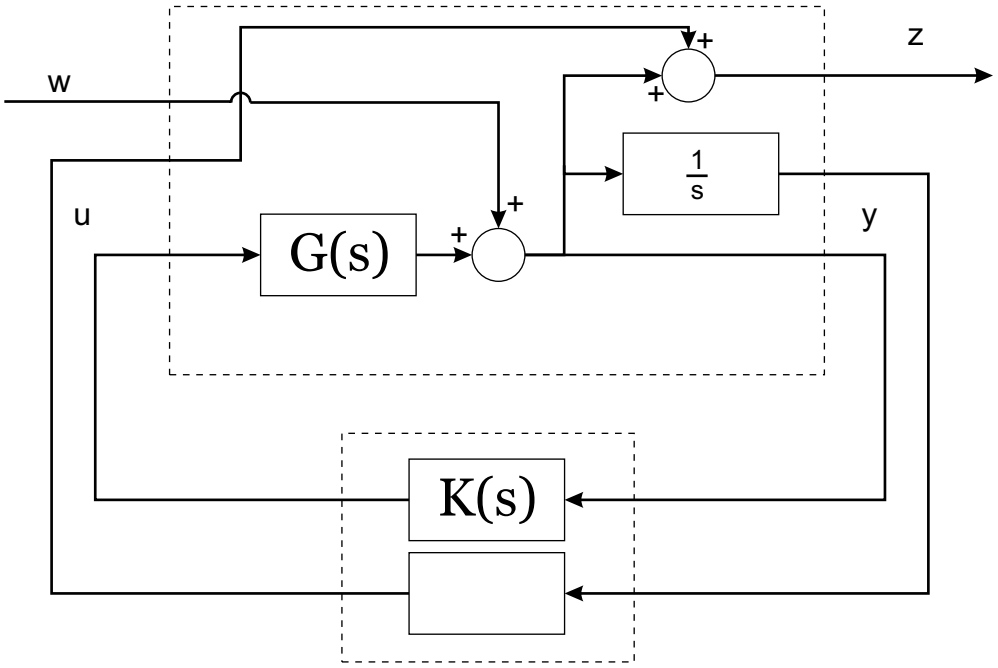

FIG. A.5 – Nouvelle formulation du problème

Bibliographie

[1] G. Scorletti. *Approche unifiée de l'analyse et de la commande des systèmes par la formulation LMI.* PhD thesis, Paris XI et Supélec, 1997.

[2] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[3] J.C. Willems. Least squares stationary optimal control and the algebraic riccati equation. In *IEEE Trans. Aut. Control*, pages 621–624, 1971.

[4] S. Boyd, V. Balakrishnan, E. Feron, and L. El Ghaoui. Control system analysis and synthesis via linear matrix inequalities. In *American Control Conference - San Franscisco*, pages 2147–2154, Juin 1993.

[5] D. Henrion. Robust analysis with linear matrix inequalities and polynomial matrices. Graduate course on polynomial Methods for robust control - Part IV.1, Novembre 2001.

[6] C. Scherer and S. Weiland. Linear matrix inequalities in control. Dutch Institute of Systems and Control - Graduate course, Octobre 2002.

[7] F. Delebecque. *LMITOOL : a package for LMI Optimization in Scilab.* INRIA Rocquencourt, 1995.

[8] Z. Hurák. Semidefinite programming and linear matrix inequalities in control. Faculty of Electrical Engineering, Czech Technical University, Décembre 2001.

[9] J.C. Geromel, J. Bernussou, and M.C. Oliveira. h_2 norm optimization with constrained dynamic output feedback controllers : Decentralized and reliable control. In *36th Conference on Decision and Control, San Diego*, 1997.

[10] M. Yagoubi. *Commande Robuste Structurée et Optimisation Convexe.* PhD thesis, Ecole Centrale de Nantes - Université de Nantes, 2003.

[11] L. Vandenberghe and V. Balakrishnan. Algorithms and software for LMI problems in control. In *IEEE CACSD Symposium in Dearborn*, 1996.

[12] P. Gahinet, A. Nemirovski, A.J. Laub, and M. Chiali. *LMI Control Toolbox for use with MATLAB©*. The MATHWORKS Inc.