Trabajo de Fin de Grado
Ingeniería de Tecnologías Industriales

Algoritmos de primer orden para implementación del Control Predictivo para Seguimiento

Autor: Javier Maestro Valenzuela
Tutor: Teodoro Álamo Cantarero

Sevilla, 2021

Trabajo de Fin de Grado
Ingeniería de Tecnologías Industriales

Algoritmos de primer orden para implementación del Control Predictivo para Seguimiento

Autor:
Javier Maestro Valenzuela

Tutor:
Teodoro Álamo Cantarero
Profesor Catedrático

Dpto. de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo de Fin de Grado: Algoritmos de primer orden para implementación del Control Predictivo para Seguimiento

Autor: Javier Maestro Valenzuela

Tutor: Teodoro Álamo Cantarero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

En primer lugar, me gustaría dedicar este apartado a todos y cada uno de los profesores que he tenido en esta universidad. Todos ellos ofrecieron su esfuerzo, dedicación y sabiduría para hacerme uno de los mayores regalos que una persona puede hacerle a otra, dar conocimiento.

En particular de entre la larga lista de profesores, quiero hacer un mención muy especial al tutor de este proyecto, Teodoro me ha ofrecido más esfuerzo, dedicación y sabiduría que ningún otro. Y mucha paciencia.

Por último, a mi familia, amigos y compañeros, los que me han acompañado en el día a día y sin los que este trabajo no habría sido redactado.

Resumen

En este documento voy a definir y detallar algunos de los variados algoritmos que existen para implementar un sistema de Control Predictivo, comenzando por unas nociones de convexidad y resolución de un problema de optimización de una función cuadrática con algunas restricciones en caja y de igualdad.

Para resolver los problemas de minimización de una función cuadrática con estas restricciones veremos cómo obtener su formulación dual, que no es más que la reexpresión de la función utilizando una serie de dualidades. También vamos a ver algunos algoritmos de optimización como puede ser el FISTA.

Para la resolución del problema de control predictivo se definirán una serie de funcionales que se irán acercando progresivamente a una formulación propia de MPC, finalizando con un funcional que se resolverá usando la estrategia de control basada en ADMM (Alternating Direction Method of Multipliers).

El proyecto pretende contribuir al campo del seguimiento de referencias basado en MPC, con una formulación siempre factible y en el que se aprovecha la estructura del funcional para llevar a cabo una implementación eficiente y en la que no se requiere grandes recursos de memoria.

Para la implementación de todo lo anterior se habrá utilizado MATLAB, quedando expuestos los códigos en un anexo al final de este documento.

Abstract

In this document I am going to define and detail some of the various algorithms that exist to implement a Predictive Control system, starting with a few notions of convexity and solving an optimization problem of a quadratic function including some box and equality restrictions.

To solve the minization problem of a quadratic function with these kind of restrictions we are going to get its dual formulation, which is no more than the re-expression of the original function, also called primal, using dualities. We are also going to see some optimization algorithms such as FISTA.

For the resolution of the Predictive Control problem, a series of functionalities will be defined that will progressively approach a proper formulation of MPC, ending with a functional that will be resolved using the control strategy based on ADMM (Alternating Direction Method of Multipliers).

The project aims to contribute to the field of reference tracking based on MPC, with a formulation that is always feasible and in which the structure of the functional is used to carry out an efficient implementation and in which large memory resources are not required.

Fort he implementation of all the above, MATLAB will have been used, the codes are exposed in an annex at the end of this document.

Índice

ÍNDICE DE FIGURAS

Notación

$dom(h)$	Dominio de h
$epi(h)$	Epígrafo de h
$\mathbb{R}^n$	Espacio real de n dimensiones
$\min$	Mínimo de una función
$\max$	Máximo de una función
$\arg\min$	Argumento mínimo
$\|h\|$	Norma de h
$\in$	Perteneciente a
s.a.	Sujeto a
:	Tal que
$\frac{\partial x}{\partial y}$	Derivada de x respecto de y
$\forall$	Para todo
$\leq$	Menor o igual
$\geq$	Mayor o igual
$\Leftrightarrow$	Si y sólo si

1 PROBLEMA DE OPTIMIZACIÓN CONVEXA

Comienzo definiendo el objeto de estudio de este primer capítulo: la función convexa. Para ello emplearé una serie de descripciones sobre los diferentes conceptos que han de estar implicados en la definición. Por una parte, está el concepto de convexidad, qué es y qué conlleva que un conjunto sea convexo. Por otro lado, la presentación del problema general, que será un problema de minimizar una función convexa. Todos estos conceptos son bien conocidos, ya que se profundizado mucho en ellos en trabajos como [4],[7],[9].

1.1 Definición de convexidad

La convexidad es una propiedad extremadamente utilizada y desarrollada en el campo de la optimización. Antes de pasar a la función es necesario explicar un concepto fundamental, el conjunto convexo.

Definición 1-1. Un conjunto $S \subseteq \mathbb{R}^n$ será convexo si para cualquier par de elementos de S (x, y) :

$$(1-\eta)x+\eta y \in S, \forall\, \eta \in [0,1].$$

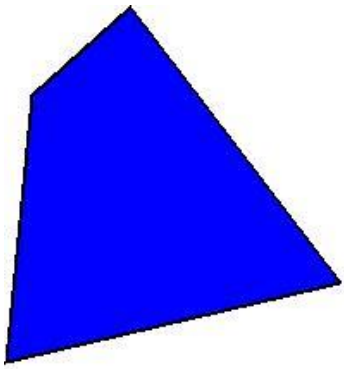


Ilustración 1-1[1]

Ilustración 1-2[2]

Se puede demostrar, esto será muy útil más adelante, que un conjunto definido por $S \equiv \{x \in \mathbb{R} : Ax \leq b\}$, es convexo. Sean dos elementos de S, $\forall x, y \in S$ por lo que cumplen la definición del conjunto

$$\alpha Ax \leq \alpha b \;\text{ y }\; \alpha \in [0,1]$$

[1] Representación gráfica de un conjunto que satisface las condiciones de convexidad.
[2] Representación gráfica de un conjunto que no satisface las condiciones de convexidad.

$$(1-\alpha)Ay \leq (1-\alpha)b.$$

Sumando ambas, tenemos

$$\alpha Ax + (1-\alpha)Ay \leq b$$
$$A(\alpha x + (1-\alpha)y) \leq b.$$

La expresión anterior es la definición de conjunto convexo. Ahora paso a definir conceptos relativos a la convexidad de una función, la que será objeto de este capítulo.

Definición 1-2. El dominio de una función lo definimos como el conjunto de puntos para los que la función toma valores finitos:

$$dom(h) = \{x \in \mathbb{R}^n : |h(x)| < +\infty\}.$$

Una función será convexa si su dominio es convexo y para todo $\alpha \in [0,1]$ se cumple que:

$$h(\alpha x + (1-\alpha)y) \leq \alpha h(x) + (1-\alpha)h(y), \forall x \in dom(h), \forall y \in dom(h).$$

El epígrafo de una función se define como el conjunto:

$$epi(h) = \{(x,t) \in dom(h) \times \mathbb{R} : h(x) \leq t\}.$$

Se concluye que una función será convexa si y sólo si su epígrafo es un conjunto convexo.

1.2 Definición del problema de optimización convexa

En segundo lugar, presento aquí el problema de optimización de una función convexa que vamos a abordar:

$$f^* = \min_{x \in X} f(x) = \min_{x \in X} h(x).$$

Trataré de encontrar el mínimo de la función objetivo $f(x)$, teniendo en cuenta una serie de restricciones que iré definiendo más adelante. Sólo se tratarán problemas de minimización y con restricciones de igualdad y restricciones en caja, debido a que cualquier problema de maximización puede convertirse en un problema de minimización, no se pierde la generalidad.

Asumiendo lo siguiente:

1. $h(x)$ es una función convexa diferenciable suave. Esto es, hay una $R \succ 0$ tal que se cumple que:

$$h(x) \leq h(y) + \langle \nabla h(y), x - y \rangle + \frac{1}{2}\|x - y\|_R^2, \quad \forall x \in \mathbb{R}^n, \quad \forall y \in \mathbb{R}^n.$$

2. El problema de minimización:

$$\min_{x \in X} f(x).$$

es resoluble. Esto es, hay una $x^* \in X \cap dom(\psi)$ tal que $f^* = f(x^*) = \inf_{x \in X} f(x)$.

Para que el problema de optimización sea convexo es necesario, por una parte, que la función $f(x)$ sea convexa, cosa que se afirma en la asunción 1. Por otra parte, que la región factible sea un conjunto convexo, esto se cumplirá cuando las restricciones de igualdad sean funciones afines y las restricciones de desigualdad sean convexas.

1.3 Reformulación del problema y resolución

Una vez formulado el problema, procedo a su resolución. Teniendo en cuenta lo anterior podemos concluir que $f(x) = h(x)$. También propongo que $h(x)$ es de la forma:

$$h(x) = \frac{1}{2} x^\top R x + q^\top x \quad .$$

Primero, tratamos de poner esta expresión en la forma original para identificar términos:

$$h(x)=\frac{1}{2}(x-y+y)^{\mathsf{T}}R(x-y+y)+q^{\mathsf{T}}(x-y+y)=$$

$$=\underbrace{\frac{1}{2}y^{\mathsf{T}}Ry+q^{\mathsf{T}}y}_{h(y)}+\frac{1}{2}(x-y)^{\mathsf{T}}R(x-y)+q^{\mathsf{T}}(x-y)+\underbrace{\frac{1}{2}(x-y)^{\mathsf{T}}Ry+\frac{1}{2}y^{\mathsf{T}}R(x-y)}_{y^{\mathsf{T}}R(x-y)}=$$

$$=h(y)+(y^{\mathsf{T}}R+q^{\mathsf{T}})(x-y)+\frac{1}{2}(x-y)^{\mathsf{T}}R(x-y)=$$

$$=h(y)+\langle q+Ry,x-y\rangle+\frac{1}{2}\|x-y\|_R^2.$$

Entonces podemos concluir que

$$\nabla h(y)=Ry+q.$$

Segundo, dado y, el x^{opt} sin restricciones se obtendrá derivando la función $h(x)$ e igualando a 0. Denotando $z=x-y$:

$$\frac{\partial h(z)}{\partial z}=\frac{\partial}{\partial z}(h(y)+\langle q+Ry,z\rangle+\frac{1}{2}\|z\|_R^2)=$$

$$=q+Ry+Rz=0\rightarrow z^{opt}=-R^{-1}(q+Ry)\rightarrow x^{opt}=z^{opt}+y.$$

Ahora introducimos las restricciones en caja, acotando χ:

$$x^-\le x\le x^+$$

$$x^--y\le z\le x^+-y$$

$$z^-\le z\le z^+.$$

Reformulamos el problema de optimización convexa en función de z:

$$f^* = \min_{x \in X} h(z) = \min_{z \in Z} h(y) + \langle q + Ry, z \rangle + \frac{1}{2} \|z\|_R^2$$

s.a.

$$z^- \leq z \leq z^+.$$

Definiendo la matriz D diagonal, definida positiva e igual al autovalor máximo de R por la matriz identidad.

$$D = \lambda_R^{\max} I = \begin{pmatrix} d_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_n \end{pmatrix} = \begin{pmatrix} \lambda_R^{\max} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_R^{\max} \end{pmatrix}.$$

Y defino: $d = \begin{bmatrix} d_1 \dots d_n \end{bmatrix} = \lambda_R^{\max} [1 \dots 1] \in \mathbb{R}^n$.

El punto óptimo de nuestra función será:

$$x^* = \arg\min_{x \in X} h(y) + \langle q + Ry, x - y \rangle + \frac{1}{2} \|x - y\|_D^2$$

s.a.

$$x^- \leq x \leq x^+.$$

Este problema es separable, es decir, cada componente del óptimo puede calcularse por separado resolviendo n problemas escalares, como veremos ahora.

En el primer programa adjunto en el Anexo I expongo el programa en MATLAB que implementa la resolución del problema de encontrar el punto óptimo de una función convexa como la que hemos estado tratando. Para ello he usado el método que he expuesto anteriormente, usar la derivada de la función e igualarla a 0 para hallar x_i^* y comparando con el resultado del comando *quadprog*. Tal como concluimos antes, para el caso sin restricciones:

$$z^{opt} = -R^{-1}(q + Ry) \rightarrow x^{opt} = z^{opt} + y.$$

Por lo tanto, en nuestra nueva formulación:

$$\hat{x}_i^* = y_i - \frac{1}{d_i}(q + Ry)_i.$$

En cada iteración se define x_i^* :

$$x_i^* = \begin{cases} x_i^- \ldots \hat{x}_i^* < x_i^- \\ \hat{x}_i^* \ldots x_i^- \leq \hat{x}_i^* \leq x_i^+ \\ x_i^+ \ldots \hat{x}_i^* > x_i^+ \end{cases}.$$

2 DUALIDAD EN OPTIMIZACIÓN CONVEXA

En la optimización convexa con restricciones hay un concepto que es de gran importancia y que se recoge en la teoría de la dualidad. Esta consiste en conectar una formulación original, también denominada primal, a través de dualidades con otra formulación llamada dual.

Conceptualmente se basa en transformar las restricciones del problema primal en variables del problema dual, así es como quedarían conectadas ambas formulaciones. Para hacer esta transformación se recurre al lagrangiano del problema original y usar una serie de propiedades que describiremos ahora. [4],[6],[7],[8].

Nuestro problema canónico incluye la minimización de una función cuadrática que es convexa y suave, con restricciones de igualdad y restricciones en caja. De esta forma:

$$J^* = \min_{z \in Z} \frac{1}{2} z^\top H z + q^\top z \quad \text{s.a.} \quad \begin{matrix} Az = b \\ Z = \left\{ z:\ z^- \le z \le z^+ \right\} \end{matrix}$$

Siendo la matriz H diagonal y definida positiva.

Así pues, este problema puede dualizarse de manera que las restricciones de igualdad en el problema primal se conviertan en una variable del problema dual, relajando las restricciones. Definiendo los teoremas de dualidad débil y dualidad fuerte sentaré las bases de lo que va a ser la dualización del problema canónico, que será un pilar fundamental en su resolución.

Este problema dual queda definido a partir de la lagrangiana del primal, basandose en la propiedad de la lagrangiana que sigue, sabiendo que λ y μ son las variables del dual:

Propiedad 2-1: Sea un problema de optimización como nuestro problema canónico, su función lagrangiana asociada $L(z, \lambda, \mu)$

$$\forall i,\ \mu_i > 0,\ z \in Z: \quad L(z, \lambda, \mu) \le f(z).$$

Se define la función dual:

$$g_0(\lambda, \mu) = \min_{z \in Z} L(z, \lambda, \mu).$$

La función dual es una cota inferior de la función objetivo evaluada en el óptimo

$$g(\lambda,\mu)\leq f(z^*).$$

Es decir, la mejor cota se da para el mayor valor posible de la función dual. Por tanto, definimos el problema dual como

$$\max_{\lambda,\mu} g(\lambda,\mu)$$

$$\forall i:\ \mu_i>0.$$

De esto podemos adoptar una serie de conclusiones, la función dual, basada en el lagrangiano de la primal, se define como el mínimo de este lagrangiano y es un problema cóncavo que obtiene el óptimo del problema primal en su máxima cota. Por lo tanto, un problema de minimización de función convexa se transformaría en un problema de maximización a través de su formulación dual.

2.1 Dualidad aplicada al problema de optimización

Ahora bien, sabiendo qué es la formulación dual de un problema de optimización convexa y teniendo en cuenta el teorema de la dualidad fuerte, y recordando el problema canónico:

$$J^*=\min_{z\in Z}\frac{1}{2}z^{\mathsf{T}}Hz+q^{\mathsf{T}}z \quad \text{s.a.} \quad \begin{array}{l} Az=b \\ Z=\left\{z:\ z^-\leq z\leq z^+\right\}\end{array}$$

Defino el funcional dual del problema anterior

$$\varphi(\lambda)=\min_{z\in Z}\frac{1}{2}z^{\mathsf{T}}Hz+q^{\mathsf{T}}z+\lambda^{\mathsf{T}}(Az-b).$$

Por lo tanto

$$z(\lambda)=\arg\min_{z\in Z}\frac{1}{2}z^{\mathsf{T}}Hz+\lambda^{\mathsf{T}}(Az-b).$$

Como el óptimo del problema canónico cumple que $Az^* = b$ y $z^* \in Z$ tenemos que:

$$\varphi(\lambda) = \min_{z \in Z} \frac{1}{2} z^{\mathsf{T}} H z + \lambda^{\mathsf{T}} (Az - b) \leq \frac{1}{2} z^{*\mathsf{T}} H z^* + \lambda^{\mathsf{T}} (Az^* - b) =$$

$$= \frac{1}{2} z^{*\mathsf{T}} H z^* = J^*.$$

Por tanto:

$$J^* \geq \varphi(\lambda), \forall \lambda.$$

Para el cálculo de $z(\lambda)$, es decir, la solución al problema dual, denotamos:

- c_i a la i-ésima componente del vector $A^{\mathsf{T}} \lambda$.
- h_i al elemento i-ésimo de la diagonal de H.
- z_i a la i-ésima componente del vector z.

Con esto, y dado que H es diagonal, se tiene

$$z(\lambda) = \arg\min_{z \in Z} \frac{1}{2} z^{\mathsf{T}} H z + (A^{\mathsf{T}} \lambda)^{\mathsf{T}} z =$$

$$= \arg\min_{z \in Z} \sum_{i=1}^{n} h_i \frac{z_i^2}{2} + c_i z_i.$$

Teniendo en cuenta las restricciones en caja, y con la afirmación anterior, tenemos que $z(\lambda)$ se calcula resolviendo n problemas de optimización escalares.

$$z_i(\lambda) = \begin{cases} -\frac{c_i}{h_i} \ldots z_i^- \leq z_i \leq z_i^+ \\ z_i^- \ldots -\frac{c_i}{h_i} < z_i^- \\ z_i^+ \ldots -\frac{c_i}{h_i} > z_i^+ \end{cases}$$

Una vez caracterizada la forma de hallar $z(\lambda)$, denotando

$$z_\lambda = \arg\min_{z\in Z} \frac{1}{2} z^\top Hz + \lambda^\top (Az - b).$$

Obtenemos

(2.1)

$$\varphi(\lambda+\Delta\lambda) = \min_{z\in Z} \frac{1}{2} z^\top Hz + (\lambda+\Delta\lambda)^\top (Az-b) \le \frac{1}{2} z_\lambda^\top Hz_\lambda + (\lambda+\Delta\lambda)^\top (Az_\lambda - b) =$$

$$= \varphi(\lambda) + \Delta\lambda^\top (Az_\lambda - b).$$

$\varphi(\lambda)$ es una función cóncava en cuyo óptimo se encuentra también la solución al problema de optimización convexa. Dado que $\varphi(\lambda)$ es una cota inferior de J^*, el objetivo es calcular el máximo de $\varphi(\lambda)$ respecto de λ.

Para ello, utilizaré el algoritmo gradencial FISTA (Fast Iterative Shrinking-Threshole Algorithm), cuya explicación y descripción es el objetivo del siguiente capítulo.

3 ALGORITMO FISTA

En el capítulo anterior quedaba definido el concepto de reformular el problema de optimización convexa (primal) a través de una serie de dualidades que hacían que el óptimo del problema dual fuera también solución del primal.

Ahora, voy a describir el algoritmo que utilizo para encontrar esa solución dual. Este algoritmo se llama FISTA, por las siglas de 'Fast Iterative Shrinking-Threshole Algorithm'. Se trata de un algoritmo de descenso de gradiente rápido que se puede aplicar a problemas de optimización convexa. Es sencillo de implementar y tiene una baja complejidad computacional por iteración.[3],[4],[6].

Este algoritmo requiere conocimiento previo del valor óptimo de la función objetivo, cosa que calculamos con el procedimiento descrito en el capítulo anterior.

$$x^* = FISTA(x_0, k_{\min}, E_c)$$

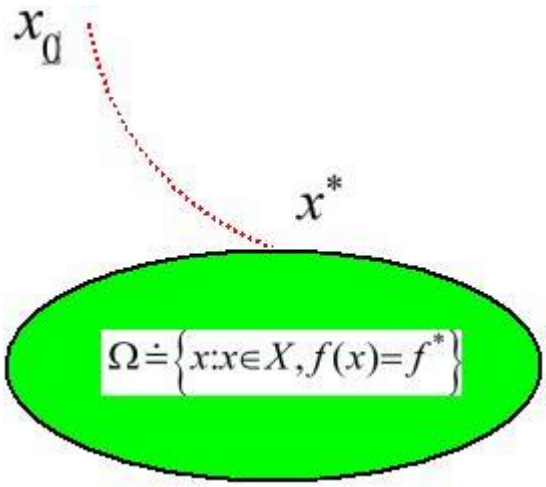


Ilustración 3-1[3]

El algoritmo requiere también el cálculo del mapeo de gradiente compuesto, que defino a continuación.

Dado $y \in \mathbb{R}^n$, el mapeo de gradiente compuesto $g(y)$ de la función $f(y) = h(y)$ se define como

[3] Representación gráfica simplificada de un problema de optimización con restricciones de igualdad y en caja.

$$g(y)=R(y-y^{+}).$$

Donde

$$y^{+}=\arg\min_{x\in X}\psi(y)+\langle q+Ry,x-y\rangle+\frac{1}{2}\|x-y\|_R^2.$$

Este problema es fácilmente resoluble. El algoritmo tiene un antecesor, un algoritmo gradencial llamado ISTA, que tiene también como protagonista a la formulación dual y que tiene la forma

---

1) **ELEGIMOS** ε (parámetro de tolerancia)

2) $k=0,\lambda_k=0$

3) **REPETIR**

4) $z_k=\arg\min_{z\in Z}\ \frac{1}{2}z^{\mathsf{T}}Hz+\lambda_k^{\mathsf{T}}(Az-b)$

5) $\lambda_{k+1}=\lambda_k+(AH^{-1}A^{\mathsf{T}})^{-1}(Az_\lambda-b)$

6) **CONDICIÓN DE SALIDA** $\|Az_k-b\|_2\le\varepsilon$

7) $k=k+1$ **and GOTO(3)**

Este algoritmo se implementará en el capítulo siguiente y justificaremos la utilización del algoritmo FISTA sobre él por términos de eficiencia. Ahora sí, el FISTA.

---

1) **ELEGIMOS** $z\in\mathbb{R}^n,k_{\min}\ge 0,E_c.$

2) $y_0=x_0=z^{+},t_0=1,k=0$

3) **REPETIR**

4) $k=k+1$

5) $x_k=y_{k-1}^{+}$

6) $t_k = \frac{1}{2}(1+\sqrt{1+4t_{k-1}^2})$

7) $y_k = x_k + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$

8) **CONDICIÓN DE SALIDA** E_c

9) **HASTA** E_c **and** $k \geq k_{\min}$

La salida de la función sería la variable $r = x_k$. Este bucle habría que repetirlo en un programa principal que responde al siguiente esquema:

---

1) **REQUIERE** $r_0 \in X, k_{\min} \geq 0, \varepsilon > 0, E_c$.

2) $j = 0$

3) **REPETIR**

4) $j = j+1$

5) $r_j = FISTA(r_{j-1}, k_{\min}, E_c)$

9) **HASTA** $\left\| g(r_j) \right\|_* \leq \varepsilon$

La salida de la función sería la variable $x^* = r_j$.

4 INTRODUCCIÓN A LA FORMULACIÓN MPC

El control predictivo es un conjunto de métodos de control que fue conceptualmente concebido en la industria petroquímica a finales de los 70. Está desarrollado sobre ciertos principios comunes tales como el uso de un modelo de proceso para pronosticar la salida del sistema en momentos futuros de la dinámica y el cálculo de una acción de control óptima basado en la minimización de una función de coste y con la posibilidad de añadir restricciones sobre las variables de ese proceso.

Este modelo de la planta se denomina modelo de predicción, y su comportamiento futuro se predice dentro de un intervalo de tiempo llamado horizonte de predición. Una vez aplicada la estrategia para encontrar la acción de control óptima se recalculará ésta desplazando el horizonte de predicción ajustándolo a las condiciones iniciales. Esto se llama horizonte deslizante.[1],[2],[3],[4],[5].

El control predictivo presenta una serie de ventajas respecto de otros métodos de control, entre ellas están.

- Permite controlar procesos con comportamientos dinámicos tales como procesos de fase no-mínima o procesos altamente oscilatorios.
- Aunque se basa en principios básicos, es una metodología abierta que permite contribuciones y mejoras en el futuro.
- Su carácter predictivo hace que compense intrínsecamente los tiempos muertos.
- Posibilidad de incorporar restriccionesa a las variables de control.

Por otra parte, también tiene sus desventajas, como el alto coste computacional que requiere la complejidad del algoritmo. Aunque lo cierto es que esta desventaja se está paliando con los avances tecnológicos en los últimos años, por lo que podría no considerarse una gran desventaja actualmente.

4.1 Fundamento del MPC

El objetivo del MPC consiste en el hallazgo de una trayectoria futura de la variable manipulada u. Para lograr este objetivo, se sigue la siguiente metodología:

Se fija un horizonte de predicción N, para el cual se van a predecir las salidas futuras en cada instante. Esto se logra utilizando un modelo de predicción que describe el comportamiento de las variables del proceso a controlar y el cual reside en el controlador. Las predicciones dependen de los valores conocidos hasta el instante k_i y de las señales de control futuras.

Para calcular las señales de control futuras se utiliza una función de coste que en el caso estándar es cuadrática y penaliza los errores entre la salida predicha y la trayectoria de referencia. En el caso de este trabajo, iré exponiendo gradualmente diferentes funciones de coste y sus resultados, concluyendo en una función más rica y propia de este tipo de estrategias.

Este problema, en ausencia de restricciones, puede ser resuelto analíticamente. En caso contrario es necesario un algoritmo iterativo de optimización. Para ambas soluciones, las ecuaciones devuelven un vector que

contiene las acciones futuras, cuya dimension depende del horizonte de control. Sin embargo, solamente el primer elemento de este vector tiene interés para nuestra estrategia y se calculará uno nuevo en cada instante de la dinámica. Esto se refleja en la siguiente figura.

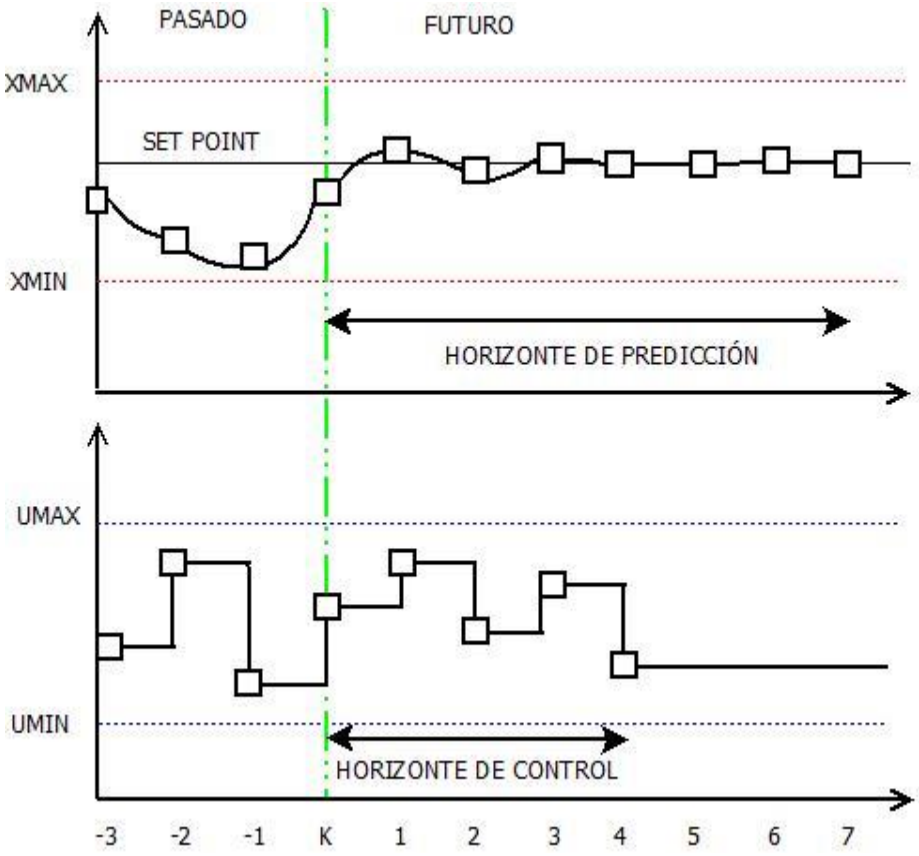


Ilustración 4-1[4]

4.2 Formulaciones del problema MPC

Vamos a considerar 3 formulaciones diferentes para plantear la función de coste que queremos minimizar para nuestro problema MPC, desde una más básica hasta otra más elaborada. El procedimiento consistirá en transformar la formulación inicial en un problema cuadrático QP que dualizaremos para poder resolver.

Para la resolución dual utilizaré el algoritmo FISTA en las tres formulaciones, comparándolo con el resultado del commando *quadprog*. En el caso de la tercera formulación, utilizaré también el algoritmo ADMM (Alternating Direction Method of Multipliers), que será objeto del siguiente capítulo.

[4] Esquema de explicación de la dinámica típica de un sistema controlado con MPC.

4.2.1 Formulación 1

Vamos a partir de una formulación simplificada de un problema MPC, en la que tendremos una serie de restricciones y una simplificación inicial que dice que nuestro punto de equilibrio estará en cero. El objetivo será no circunscribirnos a la restricción de esta formulación y poder hacer algo con condiciones más libres, lo que veremos en la siguiente formulación (2).

$$J = \min_{\substack{u_0,\dots,u_N \\ x_0,\dots,x_N}} \sum_{i=0}^{N} (x_i - x_e)^\top Q(x_i - x_e) + (u_i - u_e)^\top R(u_i - u_e) =$$

$$= \min_{\substack{u_0,\dots,u_N \\ x_0,\dots,x_N}} \frac{1}{2} z^\top H_1 z + q_1^\top z .$$

Sabiendo que $Q > 0, R > 0$ que son diagonales y denominando

$$z = \begin{bmatrix} x_0, u_0, x_1, u_1, x_2, u_2, \dots, x_{N-1}, u_{N-1}, x_N, u_N \end{bmatrix}$$

$$H_1 = 2 \begin{bmatrix} Q & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & R & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & Q & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & R \end{bmatrix}$$

$$q_1^\top = [-2x_e^\top Q \quad -2u_e^\top R \quad -2x_e^\top Q \quad -2u_e^\top R \quad \cdots \quad -2x_e^\top Q \quad -2u_e^\top R].$$

El problema está sujeto a las siguientes restricciones de igualdad y en caja.

$x_0 = x$, condición inicial que es dato.

$$x_{i+1} = Ax_i + Bu_i, \quad i = 0,\ldots,N-1$$
$$x_N = x_e$$
$$u_N = u_e$$
$$\underline{x} \le x_i \le \overline{x}, \quad i = 0,\ldots,N$$
$$\underline{u} \le u_i \le \overline{u}, \quad i = 0,\ldots,N.$$

Que podemos reescribir definiendo las matrices

$$D = \begin{bmatrix} A & B & -I & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & A & B & -I & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & A & B & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & A & B & -I & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ x_e \\ u_e \end{bmatrix}.$$

Como simplificación inicial, tenemos que $x_e = 0, u_e = 0$. Este es el punto de equilibrio y ha de cumplir que

$Ax_e + Bu_e = x_e$. Reescribiendo el problema, tenemos:

$$\min_{\substack{u_0,\dots,u_N \\ x_0,\dots,x_N}} \quad \frac{1}{2} z^\top H_1 z + q_1^\top z$$

s.a.

$$Dz = b$$
$$\underline{z} \le z \le \overline{z}\,.$$

Sabiendo que

$$\underline{z} = \begin{bmatrix} \underline{x} \\ \underline{u} \\ \underline{x} \\ \underline{u} \\ \vdots \\ \underline{x} \\ \underline{u} \end{bmatrix}, \quad \overline{z} = \begin{bmatrix} \overline{x} \\ \overline{u} \\ \overline{x} \\ \overline{u} \\ \vdots \\ \overline{x} \\ \overline{u} \end{bmatrix}.$$

Tal como vimos en capítulos anteriores, vamos a aplicar el algoritmo FISTA para resolver el problema de optimización del dual a este, pero de la forma que nos explica el método MPC descrito anteriormente. Para ello voy a comenzar por dualizar el problema y formularlo de manera que sea sencillo de resolver en función de las variables duales.

El funcional dual sería

$$\varphi(\lambda) = \min_{z \in Z} \sum_{i=0}^{N} \frac{1}{2} z^\top H_1 z + q_1^\top z + \lambda^\top (Dz - b)$$

$$= \arg\min_{z \in Z} \sum_{i=0}^{n} h_{1(i)} \frac{z_{(i)}^2}{2} + q_{1(i)} + c_{(i)} z_{(i)}\,.$$

Ya vimos que este problema es separable, por lo que hallamos el óptimo resolviendo un problema escalar por

cada componente. Teniendo en cuenta que $h \rightarrow$diagonal de H y que $c = D^{\top}\lambda$. Por lo tanto, para calcular el óptimo del problema dual:

$$y_{(i)}(\lambda) = \begin{cases} -\frac{c_{(i)}}{h_{1(i)}} \cdots \underline{z}_{(i)} \leq -\frac{c_{(i)}}{h_{1(i)}} \leq \overline{z}_{(i)} \\ \underline{z}_{(i)} \cdots -\frac{c_{(i)}}{h_{1(i)}} < \underline{z}_{(i)} \\ \overline{z}_{(i)} \cdots -\frac{c_{(i)}}{h_{1(i)}} > \overline{z}_{(i)} \end{cases}$$

Como ya se explica en el capítulo 2.

Este proceso se hará para todas las formulaciones que presentaré aquí, para así poder proceder a su resolución con el agortimo FISTA, que trabaja con la formulación dualizada del problema de optimización. Este algoritmo va a ser utilizado de cara a resolver el problema basado en MPC, que nos devolverá las acciones de control que minimizan la función de coste en función de predicciones futuras del sistema.

Para ello seguimos el siguiente procedimiento:

1) Elegir el horizonte de predicción y el número de componentes del vector de control: N y n respectivamente.
2) Definir el punto de equilibrio: $x_e = 0, u_e = 0$ en este primer caso.
3) Inicializar vectores de salida $x(n, N)$ y de actuación $u(n, N)$. Así como sus respectivos límites para las restricciones en caja: $\underline{x}, \overline{x}, \underline{u}, \overline{u}$.
4) Definir la condición inicial, que será dato para el problema, por ejemplo, elijo el valor medio entre los límites inferior y superior de la salida.

$$x_0 = \frac{\underline{x} + \overline{x}}{2}.$$

5) Construimos las matrices Q, R, A, B, H_1, D.
6) Elegimos el parámetro de tolerancia ε, 10^{-5} por ejemplo.
7) REPETIR mientras la norma de la diferencia entre la primera componente del vector que contiene la salida del sistema y la acción de control (z), y el vector de referencias l sea mayor que ε.
8) FOR $i = 1 : N - 1$

9) Ejecutar algoritmo FISTA con condición que cumple las restricciones de igualdad, $norm(Dz(:,i)-b)>\varepsilon$.

4.2.2 Formulación 2

Abordamos ahora una formulación más propia de MPC, aún con sus limitaciones. Siendo x_s, u_s un punto de equilibrio que cumple estrictamente las restricciones. Esta formulación garantiza que el problema va a ser factible, asumiendo que existe un punto de equilibrio que cumple las restricciones,cosa que tenemos en x_s, u_s .

Usaremos una estrategia de control que se fundamenta en el uso de un objetivo artificial de control x_s . Dicha estrategia se basa en ponderar tanto una distancia del estado k-ésimo y de la actuación k-ésima al estado y actuación objetivo artificial con respecto a la referencia. Si esta última ponderación es elevada, la salida final será cercana a la referencia, mientras que si es baja la salida no tiene por qué acercarse a la referencia requerida. [4][5].

$$J=\min_{\substack{u_0,\dots,u_{N-1}\\ x_0,\dots,x_N\\ x_s,u_s}} \sum_{i=0}^{N-1}(x_i-x_s)^{\top}Q(x_i-x_s)+(u_i-u_s)^{\top}R(u_i-u_s)+$$

$$+(x_e-x_s)^{\top}T(x_e-x_s)+(u_s-u_e)^{\top}S(u_s-u_e)+\tau\left\|x_0-x\right\|_1+(x_N-x_e)^{\top}T(x_N-x_e)=$$

$$=\min_{\substack{u_0,\dots,u_{N-1}\\ x_0,\dots,x_N\\ x_s,u_s}} \frac{1}{2}z^{\top}H_2z+q_2^{\top}z\cdot$$

Sabiendo que:

- x (condición inicial) es dato.
- (x_e, u_e), punto de equilibrio que también viene dado.
- τ es un escalar >0 .

[5] Esta estrategia fue desarrollada por el Departamento de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Esta publicada en el artículo ''MPC for tracking piecewise constant references for constrained linear systems'', escrito por D. Limón, I. Alvarado, T. Álamo y E. F. Camacho.

La función está sujeta a una serie de restricciones

$$x_{i+1} = Ax_i + Bu_i, \quad i = 0,\ldots,N-1$$
$$x_s = Ax_s + Bu_s$$
$$x_N = x_e$$
$$\underline{x} \le x_i \le \overline{x}, \quad i = 0,\ldots,N$$
$$\underline{u} \le u_i \le \overline{u}, \quad i = 0,\ldots,N-1$$
$$\underline{x} + \varepsilon \le x_s \le \overline{x} - \varepsilon$$
$$\underline{u} + \varepsilon \le u_s \le \overline{u} - \varepsilon$$

La formulación número 2 tan sólo tiene una intención de acercamiento a un función más propia de MPC, el cálculo del óptimo en cada iteración se realizará tanto con el algoritmo ISTA como con el FISTA.

4.2.3 Formulación 3

Para este bloque se va a trabajar con una formulación más compleja y adaptada a lo que es una expresión típica basada en MPC. Tras definirla, se calculará su expresión dual para proceder a su resolución en el siguiente capítulo. El capítulo 5 estará basado en el uso de método ADMM a través de la expresión dual de esta formulación.

- Definimos 2 vectores que incluirán las variables de decisión que aparecen en la formulación 2 (v) y las variables que están restringidas (y). Estos vectores pueden colocarse según 2 disposiciones diferentes: separada e intercalada. Aquí usaremos la ordenación intercalada de manera que H nos quede como una matriz en banda, esto lo veremos luego.

$$v = \left[x_0, u_0, x_1, u_1, x_2, u_2, \ldots, x_{N-1}, u_{N-1}, x_N, x_s, u_s \right].$$

Teniendo que

$$x_s = Ax_s + Bu_s$$
$$x_{i+1} = Ax_i + Bu_i, i = 0,\ldots,N-1$$
$$x_N = x_s$$

Estas restricciones pueden reescribirse de modo que queden de la forma

$$Dv = 0.$$

Donde D es una matriz de la forma

$$D = \begin{bmatrix} A & B & -I & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A & B & -I & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A & B & -I & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A & B & -I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & I & -I & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & (A-I) & B \end{bmatrix}.$$

El segundo vector es

$$y = \left[\bar{x}_0, \bar{u}_0, \bar{x}_1, \bar{u}_1, \bar{x}_2, \bar{u}_2, \ldots, \bar{x}_{N-1}, \bar{u}_{N-1}, \bar{x}_N, \bar{x}_s, \bar{u}_s \right].$$

Cuyas restricciones son las siguientes

$$\begin{aligned} &\underline{x} \le x_i \le \bar{x}, \quad i = 0, \ldots, N \\ &\underline{u} \le u_i \le \bar{u}, \quad i = 0, \ldots, N-1 \\ &\underline{x} + \varepsilon \le x_s \le \bar{x} - \varepsilon \\ &\underline{u} + \varepsilon \le u_s \le \bar{u} - \varepsilon \end{aligned}$$

Al igual que antes, reescribimos las restricciones de manera que quedan de la forma.

$$\underline{y}=\begin{bmatrix}\underline{x}\\ \underline{u}\\ \underline{x}\\ \vdots\\ \underline{x}+\varepsilon\\ \underline{u}+\varepsilon\end{bmatrix}\leq y\leq \overline{y}=\begin{bmatrix}\overline{x}\\ \overline{u}\\ \overline{x}\\ \vdots\\ \overline{x}-\varepsilon\\ \overline{u}-\varepsilon\end{bmatrix}.$$

Teniendo definidos ambos vectores, anuncio la restricción que vamos a dualizar:

$$v=y.$$

Ahora definiremos el funcional en términos de v e y. Sabiendo que $(\tau_Q>0,\tau_Q\in\mathbb{R})$ tiene que ser tal que $Q-\tau_Q I>0$, se calcula

$$\tau_Q=\frac{\lambda_{\min}(Q)}{2}$$

i) $$\left\|x_i-x_s\right\|_Q^2=\left\|x_i-x_s\right\|_Q^2-\left\|x_i-x_e\right\|_{\tau_Q I}^2+\left\|x_i-x_e\right\|_{\tau_Q I}^2=$$

$$=\underbrace{\left\|x_i-x_s\right\|_Q^2-\left\|x_i-x_e\right\|_{\tau_Q I}^2}_{\text{estrictamente convexo en } x_i}+\underbrace{\left\|\overline{x}_i-x_e\right\|_{\tau_Q I}^2}_{\substack{\text{estrictamente}\\ \text{convexo en } \overline{x}_i}}.$$

Lo anterior es análogo para las expresiones que siguen

$$\tau_R=\frac{\lambda_{\min}(R)}{2}$$

ii) $$\left\|u_i-u_s\right\|_R^2=\left\|u_i-u_s\right\|_R^2-\left\|u_i-u_e\right\|_{\tau_R I}^2+\left\|\overline{u}_i-u_e\right\|_{\tau_R I}^2$$

$$\tau_T = \frac{\lambda_{\min}(T)}{2}$$

iii) $$\left\|x_s - x_e\right\|_T^2 = \left\|x_s - x_e\right\|_T^2 - \left\|x_s - x_e\right\|_{\tau_T I}^2 + \left\|\overline{x}_s - x_e\right\|_{\tau_T I}^2$$

iv) $$\left\|x_N - x_e\right\|_T^2 = \left\|x_N - x_e\right\|_T^2 - \left\|x_N - x_e\right\|_{\tau_T I}^2 + \left\|\overline{x}_N - x_e\right\|_{\tau_T I}^2$$

$$\tau_S = \frac{\lambda_{\min}(S)}{2}$$

v) $$\left\|u_s - u_e\right\|_S^2 = \left\|u_s - u_e\right\|_S^2 - \left\|u_s - u_e\right\|_{\tau_S I}^2 + \left\|\overline{u}_s - u_e\right\|_{\tau_S I}^2$$

- Sumando todos estos términos, obtenemos la expresión del funcional de esta tercera formulación como suma de dos términos:

$$J_v = \sum_{i=0}^{N-1} \left(\left\|x_i - x_s\right\|_Q^2 - \left\|x_i - x_e\right\|_{\tau_Q I}^2 + \left\|u_i - u_s\right\|_R^2 - \left\|u_i - u_e\right\|_{\tau_R I}^2\right) +$$

$$+ \left\|x_s - x_e\right\|_{T-\tau_T I}^2 + \left\|x_N - x_e\right\|_{T-\tau_T I}^2 + \left\|u_s - u_e\right\|_{S-\tau_S I}^2 = \frac{1}{2} v^{\mathsf{T}} H_v v + q_v^{\mathsf{T}} v + c\,.$$

Las matrices anteriores tienen esta forma

$$H_v = 2\begin{bmatrix} Q-\tau_Q I & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & -Q & 0 \\ 0 & R-\tau_R I & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & -R \\ 0 & 0 & Q-\tau_Q I & 0 & \cdots & 0 & 0 & 0 & -Q & 0 \\ 0 & 0 & 0 & R-\tau_R I & \cdots & 0 & 0 & 0 & 0 & -R \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & Q-\tau_Q I & 0 & 0 & -Q & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & R-\tau_R I & 0 & 0 & -R \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & T-\tau_T I & 0 & 0 \\ -Q & 0 & -Q & 0 & \cdots & -Q & 0 & 0 & NQ+(T-\tau_T I) & 0 \\ 0 & -R & 0 & -R & \cdots & 0 & -R & 0 & 0 & NR+(S-\tau_S I) \end{bmatrix}$$

Para garantizar la viabilidad del problema, es necesario que las matrices S y T sean lo suficientemente grandes como para que se afirme que H_v es definida positiva. De este modo, se demuestra que el problema es convexo y que se encontrará un mínimo de manera factible.

$$q_v^\top = [\overbrace{2x_e^\top \tau_Q I}^{x_0} \quad \overbrace{2u_e^\top \tau_R I}^{u_0} \quad \overbrace{2x_e^\top \tau_Q I}^{x_1} \quad \overbrace{2u_e^\top \tau_R I}^{u_1} \quad \cdots \quad \overbrace{2u_e^\top \tau_R I}^{x_{N-1}} \quad \overbrace{2u_e^\top \tau_R I}^{u_{N-1}} \quad \overbrace{-2x_e^\top (T-\tau_T I)}^{x_N} \quad \overbrace{-2x_e^\top (T-\tau_T I)}^{x_S} \quad \overbrace{-2u_e^\top (S-\tau_S I)}^{u_S}]$$

Y la constante

$$c = -\tau_Q N x_e^\top x_e - \tau_R N u_e^\top u_e + 2x_e^\top (T-\tau_T I) x_e + u_e^\top (S-\tau_S I) u_e .$$

La constante anterior no influirá en la optimización, por lo que no la tendremos en cuenta a partir de ahora.

Para el segundo término

$$J_y = \sum_{i=0}^{N-1} (\left\| \overline{x}_i - x_e \right\|_{\tau_Q I}^2 + \left\| \overline{u}_i - u_e \right\|_{\tau_R I}^2) + \left\| \overline{x}_s - x_e \right\|_{\tau_T I}^2 +$$
$$+ \left\| \overline{x}_N - x_e \right\|_{\tau_T I}^2 + \left\| \overline{u}_s - u_e \right\|_{\tau_S I}^2 + \tau \left\| \overline{x}_0 - x \right\|_1 .$$

Para calcular el término lineal hacemos lo siguiente

$$\tau\left\|\bar{x}_0 - x\right\|_1 = \tau\sum_{j=1}^{n_x}\left|\bar{x}_{0,j} - x_j\right|.$$

De esta forma, el funcional que vamos a analizar de aquí en adelante es de la forma

$$J = J_v + J_y.$$

El problema de optimización queda entonces definido como

$$\min_{v,y} J \quad \text{s.a.} \begin{cases} Dv = 0 \\ \underline{y} \le y \le \bar{y} \\ v = y \end{cases}.$$

Para llevar a cabo la resolución presento aquí su formulación dual, en la que $v = y$ representa la restricción dualizada. La función varphy:

$$\min_{v,y} \; J_v + J_y + \lambda^{\mathsf{T}}(v - y) \quad \text{s.a.} \begin{cases} Dv = 0 \\ \underline{y} \le y \le \bar{y} \end{cases}.$$

Este problema tiene la gran ventaja de ser separable, es decir, se pueden calcular por separado las variables duales dada una cierta λ de esta forma

$$v_\lambda = \arg\min_v \; J_\lambda(v) = J_v + \lambda^{\mathsf{T}} v \quad , \text{ s.a. } Dv = 0$$

$$y_\lambda = \arg\min_y \; J_\lambda(y) = J_y - \lambda^{\mathsf{T}} \mathrm{y} \quad , \text{ s.a. } \underline{\mathrm{y}} \le y \le \bar{y} \; .$$

La primera de ellas se calcula resolviendo el siguiente sistema de ecuaciones, según lo descrito en [1],[3],[4].

$$J_\lambda(v)=\frac{1}{2}v^\mathsf{T}H_v v+q_v^\mathsf{T}v+\lambda^\mathsf{T}v$$

$$\begin{bmatrix} H_v & D^\mathsf{T} \\ D & 0 \end{bmatrix}\begin{bmatrix} v_\lambda \\ \eta_\lambda \end{bmatrix}=\begin{bmatrix} -q_v-\lambda \\ 0 \end{bmatrix}.$$

Despejando:

$$H_v v_\lambda+D^\mathsf{T}\eta_\lambda=-q_v-\lambda \;\rightarrow\; v_\lambda=H_v^{-1}(-q_v-\lambda-D^\mathsf{T}\eta_\lambda)$$
$$Dv_\lambda=0.$$

La segunda se resuelve componente a componente de y_λ, es un problema separable. Para cada componente z el problema es de la forma

$$\min_{z\in\mathbb{R}} \;\frac{r}{2}(z-d)^2+a\left|z-s\right|+gz \;\text{ s.t. }\; \underline{z}\le z\le\overline{z}\,.$$

Para hacerlo, recordemos la forma del funcional original

$$J_y=\sum_{i=0}^{N-1}(\left\|\overline{x}_i-x_e\right\|_{\tau_Q I}^2+\left\|\overline{u}_i-u_e\right\|_{\tau_R I}^2)+\left\|\overline{x}_s-x_e\right\|_{\tau_T I}^2+$$
$$+\left\|\overline{x}_N-x_e\right\|_{\tau_T I}^2+\left\|\overline{u}_s-u_e\right\|_{\tau_S I}^2+\tau\left\|\overline{x}_0-x\right\|_1.$$

Si analizamos el funcional componente a componente de y nos queda

$$\sum_{i=0}^{N-1}(\tau_Q\sum_{j=1}^{n_x}(\overline{x}_{i,j}-x_{e,j})^2+\tau_R\sum_{j=1}^{n_x}(\overline{u}_{i,j}-u_{e,j})^2)+\tau_T\sum_{j=1}^{n_x}(\overline{x}_{s,j}-x_{e,j})^2+$$
$$+\tau_T\sum_{j=1}^{n_x}(\overline{x}_{N,j}-x_{e,j})^2+\tau_S\sum_{j=1}^{n_x}(\overline{u}_{s,j}-u_{e,j})^2+\tau\sum_{j=1}^{n_x}\left|\overline{x}_{0,j}-x_j\right|.$$

Si particularizo la expresión para, por ejemplo, la primera componente y le añado el término dual

$$\tau_Q(\overline{x}_{0,1}-x_{e,1})^2+\tau_R\sum_{j=1}^{n_x}(\overline{u}_{i,j}-u_{e,j})^2+\tau_T\sum_{j=1}^{n_x}(\overline{x}_{s,j}-x_{e,j})^2+$$

$$+\tau_T\sum_{j=1}^{n_x}(\overline{x}_{N,j}-x_{e,j})^2+\tau_S\sum_{j=1}^{n_x}(\overline{u}_{s,j}-u_{e,j})^2+\tau\left|\overline{x}_{0,j}-x_j\right|-\lambda_1\overline{x}_{0,1}.$$

Reordenando

$$2\tau_Q\frac{1}{2}(\overline{x}_{0,1}-x_{e,1})^2+\tau\left|\overline{x}_{0,1}-x_1\right|-\lambda_1\overline{x}_{0,1}+\tau_R\sum_{j=1}^{n_x}(\overline{u}_{i,j}-u_{e,j})^2+\tau_T\sum_{j=1}^{n_x}(\overline{x}_{s,j}-x_{e,j})^2+$$

$$+\tau_T\sum_{j=1}^{n_x}(\overline{x}_{N,j}-x_{e,j})^2+\tau_S\sum_{j=1}^{n_x}(\overline{u}_{s,j}-u_{e,j})^2.$$

Si ahora denomino:

- $r=2\tau_Q$
- $a=\tau$
- $g=-\lambda_1$
- $d=x_{e,1}$
- $s=x_j$
- $z=\overline{x}_{0,1}$

Puedo ponerlo de la forma que se indicaba

$$\frac{r}{2}(z-d)^2+a\left|z-s\right|+gz+c.$$

La constante representa al resto de términos del funcional que no dependen de la componente sobre la que he particularizado en este caso, esto pasará para cada componente de y. Nótese que cuando z no sea una de las componentes de $\overline{x}_0$, el término que multiplica a pasará a ser constante y será inocuo para la optimización.

Ahora debemos analizar una casuística, ya que la incógnita a puede ser igual o distinto de cero. Sólo será distinto de cero para $\overline{x}_0$.

Caso 1: $a = 0$

$$\min_{z\in\mathbb{R}} \ \frac{r}{2}(z-d)^2 + gz \ \xrightarrow{\frac{\partial}{\partial z}=0} \ \mathrm{r}(\hat{z}^* - d) + g = 0$$

$$\hat{z}^* = d - \frac{g}{r} = \begin{cases} d - \frac{g}{r} \ \ \ldots \ \ d - \frac{g}{r} \in [\underline{z}, \overline{z}] \\ \underline{z} \ \ \ldots \ \ d - \frac{g}{r} < \underline{z} \\ \overline{z} \ \ \ldots \ \ d - \frac{g}{r} > \overline{z} \end{cases}.$$

Dicho de otra forma

$$\hat{z}^* = \max\left\{ \underline{z} \ , \ \min\left\{ d - \frac{g}{r} \ , \ \overline{z} \right\} \right\}.$$

Caso 2: $a \neq 0$

Se puede resolver el problema 2 veces en función del signo de $\hat{z}^* - s$.

1. Asumimos que $\hat{z}^* - s > 0$

$$\min_{z} \ \frac{r}{2}(z-d)^2 + az - as + gz \ \xrightarrow{\frac{\partial}{\partial z}=0} \ \mathrm{r}(\hat{z}^* - d) + a + g = 0.$$

Despejando y denotando $\overline{g} = g + a$

$$\hat{z}^* = d - \frac{\bar{g}}{r} = \begin{cases} d - \frac{\bar{g}}{r} \ \ \ldots \ \ d - \frac{\bar{g}}{r} \in [\underline{z}, \bar{z}] \\ \underline{z} \ \ \ldots \ \ d - \frac{\bar{g}}{r} < \underline{z} \\ \bar{z} \ \ \ldots \ \ d - \frac{\bar{g}}{r} > \bar{z} \end{cases} = \max\left\{ \underline{z} \ , \ \min\left\{ d - \frac{\bar{g}}{r} \ , \ \bar{z} \right\} \right\} \ .$$

1. Asumimos que $\hat{z}^* - s < 0$

$$\min_z \ \ \frac{r}{2}(z-d)^2 - az + as + gz \ \xrightarrow{\frac{\partial}{\partial z} = 0} \ \mathrm{r}(\hat{z}^* - d) - a + g = 0.$$

Despejando y denotando $\bar{g} = a - g$

$$\hat{z}^* = d + \frac{\bar{g}}{r} = \begin{cases} d + \frac{\bar{g}}{r} \ \ \ldots \ \ d + \frac{\bar{g}}{r} \in [\underline{z}, \bar{z}] \\ \underline{z} \ \ \ldots \ \ d + \frac{\bar{g}}{r} < \underline{z} \\ \bar{z} \ \ \ldots \ \ d + \frac{\bar{g}}{r} > \bar{z} \end{cases} = \max\left\{ \underline{z} \ , \ \min\left\{ d + \bar{g} \ , \ \bar{z} \right\} \right\}.$$

Para las siguientes resoluciones hará falta poner el funcional $J_\lambda(y)$ de la forma QP. Las matrices H_y y q_y quedarían así

$$H_y = 2 \begin{bmatrix} \tau_Q I & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & \tau_R I & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \tau_Q I & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \tau_R I & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \tau_Q I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \tau_R I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \tau_T I & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \tau_T I & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \tau_S I \end{bmatrix}$$

$$q_y^\top = [\overbrace{-2x_e^\top \tau_Q I + \tau\alpha}^{\bar{x}_0} \quad \overbrace{-2u_e^\top \tau_R I}^{\bar{u}_0} \quad \overbrace{-2x_e^\top \tau_Q I}^{\bar{x}_1} \quad \overbrace{-2u_e^\top \tau_R I}^{\bar{u}_1} \quad \cdots \quad \overbrace{-2u_e^\top \tau_R I}^{\bar{x}_{N-1}} \quad \overbrace{-2u_e^\top \tau_R I}^{\bar{u}_{N-1}} \quad \overbrace{-2x_e^\top (\tau_T I)}^{\bar{x}_N} \quad \overbrace{-2x_e^\top (\tau_T I)}^{\bar{x}_S} \quad \overbrace{-2u_e^\top (\tau_S I)}^{\bar{u}_S}]$$

Siendo α un vector cuyas componentes pueden ser 1 o -1, en función del valor de $\left|\bar{x}_{0,1} - x_1\right|$.

$$c = -\tau_Q N x_e^\top x_e - \tau_R N u_e^\top u_e + 2\tau_T x_e^\top x_e + \tau_S u_e^\top u_e - \tau \sum_{j=1}^{n_x} x_j .$$

Una vez determinada la componente óptima en ambas suposiciones, vemos si $\hat{z}^* - s$ es mayor o menor que cero en los 2 casos. Cuando tengamos el óptimo que cumple su hipótesis, descartamos la otra opción. Hecho esto con todas las componentes, ya tendremos determinada la y_λ.

Como voy a resolver el problema con el algoritmo FISTA, el próximo paso será calcular la $\Delta\lambda$ utilizando la siguiente propiedad:

Propiedad 4-1. Denotanto

$$z_\lambda = \arg\min_{z \in Z} \ \frac{1}{2} z^\top H z + q^\top z + \lambda^\top (Az - b).$$

Entonces, si H es definida positiva, según lo expresado en [7],[8].

$$\varphi(\lambda + \Delta\lambda) \geq \varphi(\lambda) + \Delta\lambda^\top (Az_\lambda - b) - \frac{1}{2} \Delta\lambda^\top A H^{-1} A^\top \Delta\lambda .$$

Al ser nuestro problema de la forma

$$\min_{v,y} \ J = J_v + J_y + \lambda^\top (v - y) = \frac{1}{2} v^\top H_v v + \frac{1}{2} y^\top H_y y + q_v^\top v + q_y^\top y + \lambda^\top (v_\lambda - y_\lambda)$$

Definiendo las matrices

$$A = [I \quad -I]$$
$$b = 0.$$

De esta forma

$$Az=b \;\rightarrow\; [I \quad -I]\begin{bmatrix} v \\ y \end{bmatrix}=0 \;\rightarrow\; v-y=0.$$

Tal como ya hemos comentado, dado un λ podemos calcular tanto v_λ como y_λ. Ahora definimos

$$\varphi(\lambda)=J_v(v_\lambda)+J_y(y_\lambda)+\lambda^{\mathsf{T}}(v_\lambda-y_\lambda).$$

Utilizando la propiedad anterior y las matrices de la dinámica

$$\varphi(\lambda+\Delta\lambda)\geq\varphi(\lambda)+\Delta\lambda^{\mathsf{T}}(v_\lambda-y_\lambda)-\frac{1}{2}\Delta\lambda^{\mathsf{T}}AH^{-1}A^{\mathsf{T}}\Delta\lambda.$$

Con $H=\begin{bmatrix} H_v & 0 \\ 0 & H_y \end{bmatrix}$, tenemos que

$$AH^{-1}A^{\mathsf{T}}=\begin{bmatrix} I & -I \end{bmatrix}\begin{bmatrix} H_v & 0 \\ 0 & H_y \end{bmatrix}^{-1}\begin{bmatrix} I \\ -I \end{bmatrix}=H_v+H_y.$$

Reformulando

$$\varphi(\lambda+\Delta\lambda)\geq\varphi(\lambda)+\Delta\lambda^{\mathsf{T}}(v_\lambda-y_\lambda)-\frac{1}{2}\Delta\lambda^{\mathsf{T}}(H_v+H_y)\Delta\lambda.$$

Derivando respecto de $\Delta\lambda$ podemos concluir que el valor que maximiza la cota inferior de $\varphi(\lambda+\Delta\lambda)$ es

$$\frac{\partial}{\partial\Delta\lambda}(\varphi(\lambda)+\Delta\lambda^{\mathsf{T}}(v_\lambda-y_\lambda)-\frac{1}{2}\Delta\lambda^{\mathsf{T}}(H_v+H_y)\Delta\lambda)=0 \;\rightarrow$$
$$\rightarrow\; v_\lambda-y_\lambda-(H_v-H_y)\Delta\lambda=0.$$

Despejando obtenemos

$$\Delta\lambda^* = (H_v + H_y)^{-1}(v_\lambda - y_\lambda).$$

Planteo aquí el algoritmo a implementar para resolver esta última formulación con FISTA

---

1) **ELEGIMOS PARÁMETRO DE TOLERANCIA** ε.

2) $\lambda_0 = \varphi_0 = \varphi_1 = 0, t_0 = 1, k = 1$

3) **REPETIR**

4) $t_k = \frac{1}{2}(1+\sqrt{1+4t_{k-1}^2})$

5) $\lambda_k = \varphi_k + \frac{t_{k-1}-1}{t_k}(\varphi_k - \varphi_{k-1})$

6) $v_k = \arg\min \ \frac{1}{2} v^\top H_v v + q_v{}^\top v + \lambda_k^\top v$

7) $y_k = \arg\min \ \frac{1}{2} y^\top H_y y + q_y{}^\top y - \lambda_k^\top y$

8) $\varphi_{k+1} = \lambda_k + (H_v + H_y)^{-1}(v_k - y_k)$

9) **SI** $\|v_k - y_k\|_2 \leq \varepsilon$ **ENTONCES EXIT**

10) $k = k+1$

Cuya implementación se encuentra en el anexo de códigos al final del documento.

Con esta formulación MPC que se expone, se garantiza que el problema es siempre factible, ya que si la condición inicial no puede llevar el sistema al punto de equilibrio sin violar las restricciones, se va a calcular la condición inicial más próxima que sí es capaz de hacerlo.

Para trabajos futuros, se podría aprovechar la estructura en semi-banda de problemas intermedios de optimización para poder implementar algoritmos más eficientes, cuyo coste sea menor.

5 INTRODUCCIÓN AL MÉTODO ADMM

El método de multiplicadores de dirección alterna usa actualizaciones parciales para las variables duales. Este método se aplica a menudo para resolver problemas como el nuestro, es decir, del tipo

$$\min_{x,y} \; f(x)+g(y) \;\; , \;\; \text{s.a.} \;\; x=y \, .$$

Este problema puede atacarse con métodos de optimización restringida, cuya función objetivo es separable en x e y. [2],[3]. La actualización dual requiere resolver una función de proximidad en x e y al mismo tiempo. La técnica ADMM permite resolver este problema aproximadamente resolviendo primero x con y fijo, y luego resolviendo y con x fijo.

En lugar de iterar hasta la convergencia, el algoritmo procede a actualizar la variable dual y luego repite el proceso. Esto no es equivalente a la minimización exacta, pero se puede demostrar que este método converge a la respuesta correcta. [2].

Si particularizo a nuestro caso, tendríamos

$$\min_{v,y} J = J_v + J_y \quad \text{s.a.} \begin{cases} Dv=0 \\ \underline{y} \le y \le \overline{y} \\ v=y \end{cases} .$$

Sabiendo que J_v y J_y se corresponden con las expresiones del capítulo anterior. Ahora, dualizando el problema nos quedará la expresión que pasaremos a la formulación ADMM.

$$\min_{v,y} \; J_v + J_y + \lambda^{\mathsf{T}}(v-y) \quad \text{s.a.} \begin{cases} Dv=0 \\ \underline{y} \le y \le \overline{y} \end{cases} .$$

El problema ADMM será de la forma

$$\varphi_\rho(\lambda) = \min_{v,y} \; J_v + J_y + \lambda^{\mathsf{T}}(v-y) + \frac{\rho}{2}\left\| v-y \right\|_2^2 .$$

El algoritmo se basa en dejar fija una de las variables y calcular la otra (fijo y $\rightarrow v_{new}$), para después volver a calcular la anterior ($v_{new} \rightarrow y_{new}$). A continuación, actualizamos λ . Tendrá la siguiente estructura

---

1) **ELEGIMOS** $\varepsilon > 0$ (parámetro de tolerancia $\sim 10^{-5}$).

2) **ELEGIMOS** $\rho > 0$ y $y_0 \in Y$

3) $k=0,\ \lambda_0 = 0$

4) $v_{k+1} = \arg\min_{v} \ J_v + \lambda_k^{\mathsf{T}} v + \frac{\rho}{2}\left\| v - y_k \right\|_2^2$

5) $y_{k+1} = \arg\min_{y} \ J_y - \lambda_k^{\mathsf{T}} y + \frac{\rho}{2}\left\| v_{k+1} - y \right\|_2^2$

6) $\lambda_{k+1} = \lambda_k + \rho(v_{k+1} - y_{k+1})$

7) **SI** $\max\left\{ \left\| v_{k+1} - y_{k+1} \right\|_2 \ , \left\| y_{k+1} - y_k \right\|_2 \right\} \leq \varepsilon \rightarrow$ **EXIT**

8) $k = k+1$ y $GOTO(4)$

A continuación, desarrollo el procedimiento para resolver los puntos 4 y 5 del algoritmo.[2],[3].

4)

$$J_v = \frac{1}{2} v^{\mathsf{T}} H_v v + q_v^{\mathsf{T}} v + c$$

$$\Gamma_k(v) = J_v + \lambda_k^{\mathsf{T}} v + \frac{\rho}{2}\left\| v - y_k \right\|_2^2 =$$

$$= \frac{1}{2} v^{\mathsf{T}} H_v v + q_v^{\mathsf{T}} v + c + \lambda_k^{\mathsf{T}} v + \frac{\rho}{2} v^{\mathsf{T}} v - \rho y_k^{\mathsf{T}} v + \frac{\rho}{2} y_k^{\mathsf{T}} y_k \rightarrow$$

$$\left.\begin{array}{l}\xrightarrow{\frac{\partial}{\partial v}=0}(H_v+\rho I)v+(q_v+\lambda_k-\rho y_k)=0\\ Dv=0\end{array}\right\}=$$

$$=\begin{bmatrix}H_v & D^{\mathsf{T}}\\ D & 0\end{bmatrix}\begin{bmatrix}v_{k+1}\\ \eta\end{bmatrix}=\begin{bmatrix}-\lambda_k+\rho y_k-q_v\\ 0\end{bmatrix}.$$

Desarrollando el sistema de ecuaciones

$$(H_v+\rho I)v_{k+1}+D^{\mathsf{T}}\eta=-\lambda_k+\rho y_k-q_v$$
$$Dv_{k+1}=0.$$

Despejando

$$v_{k+1}=(H_v+\rho I)^{-1}(-\lambda_k+\rho y_k-q_v-D^{\mathsf{T}}\eta)\rightarrow Dv_{k+1}=0\rightarrow$$

$$\rightarrow D(H_v+\rho I)^{-1}(-\lambda_k+\rho y_k-q_v-D^{\mathsf{T}}\eta)=$$

$$=-D(H_v+\rho I)^{-1}D^{\mathsf{T}}\eta+D(H_v+\rho I)^{-1}(-\lambda_k+\rho y_k-q_v)=0\rightarrow$$

$$\rightarrow D(H_v+\rho I)^{-1}D^{\mathsf{T}}\eta=D(H_v+\rho I)^{-1}(-\lambda_k+\rho y_k-q_v).$$

Simplificando

$$\eta=(D(H_v+\rho I)^{-1}D^{\mathsf{T}})^{-1}D(H_v+\rho I)^{-1}(-\lambda_k+\rho y_k-q_v)$$

$$v_{k+1}=(H_v+\rho I)^{-1}(-\lambda_k+\rho y_k-q_v-D^{\mathsf{T}}(D(H_v+\rho I)^{-1}D^{\mathsf{T}})^{-1}D(H_v+\rho I)^{-1}(-\lambda_k+\rho y_k-q_v))$$

Ahora, vamos a ver como desarrollar el punto 5.

5)

$$J_y = \frac{1}{2} y^\top H_y y + q_y^\top y + c$$

$$\gamma_k(y) = J_y - \lambda_k^\top y + \frac{\rho}{2}\left\| v_{k+1} - y \right\|_2^2 =$$

$$= \frac{1}{2} y^\top H_y y + q_y^\top y + c - \lambda_k^\top y + \frac{\rho}{2} y^\top y - \rho y^\top v_{k+1} + \frac{\rho}{2} v_{k+1}^\top v_{k+1}$$

Denotando $f = \rho v_{k+1} - \lambda_k$

$$y_{k+1} = \arg\min_{y \in Y} \quad \frac{\rho}{2} \overbrace{y^\top y}^{\sum y_i^2} - \overbrace{f^\top y}^{\sum f_i y_i}$$

La componente i-ésima de y_{k+1} será

$$y_{k+1} = \begin{cases} \frac{f_i}{\rho} \ldots \frac{f_i}{\rho} \in [\underline{y}, \overline{y}] \\ \underline{y} \ldots \frac{f_i}{\rho} < \underline{y} \\ \overline{y} \ldots \frac{f_i}{\rho} > \overline{y} \end{cases}$$

En MATLAB

$$y_{k+1} = \max\left\{ \underline{y}, \min\left\{ \frac{f_i}{\rho}, \overline{y} \right\} \right\}$$

Para mostrar resultados del algoritmo anterior, expongo una serie de representaciones gráficas que muestran el comportamiento del algoritmo en función de una serie de parámetros. Para caracterizar el problema que se ha solucionado para obtenerlas, defino aquí los siguientes parámetros:

Horizonte de predicción: N=50

Dimensión del problema: n=25

Para la construcción de las matrices Q, R, T y S :

$$Q = randn(n) \qquad R = randn(n) \qquad T = cI \qquad S = cI$$
$$Q = QQ^{\mathsf{T}} + 2I \qquad R = RR^{\mathsf{T}} + 2I \qquad T = TT^{\mathsf{T}} + 2I \qquad S = SS^{\mathsf{T}} + 2I$$
$$Q = \max(\lambda_Q)I \qquad R = \max(\lambda_R)I \qquad T = \max(\lambda_T)I \qquad S = \max(\lambda_S)I$$
$$\tau_Q = \frac{\min(\lambda_Q)}{2} \qquad \tau_R = \frac{\min(\lambda_R)}{2} \qquad \tau_T = \frac{\min(\lambda_T)}{2} \qquad \tau_S = \frac{\min(\lambda_S)}{2}$$

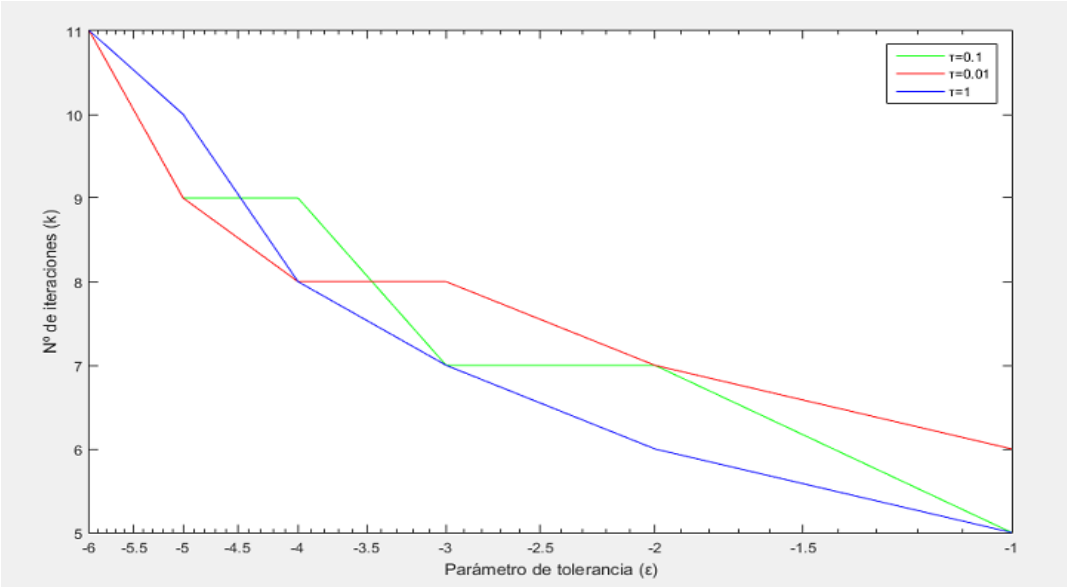
$\rho = 100$

COND. INICIAL	TIEMPO DE EJECUCIÓN (sg)
-0.333	92.4552
0.3273	99.2307
2.5777	113.2307
0.7436	113.2016
-1.3344	94.2417
-4.4671	99.7304

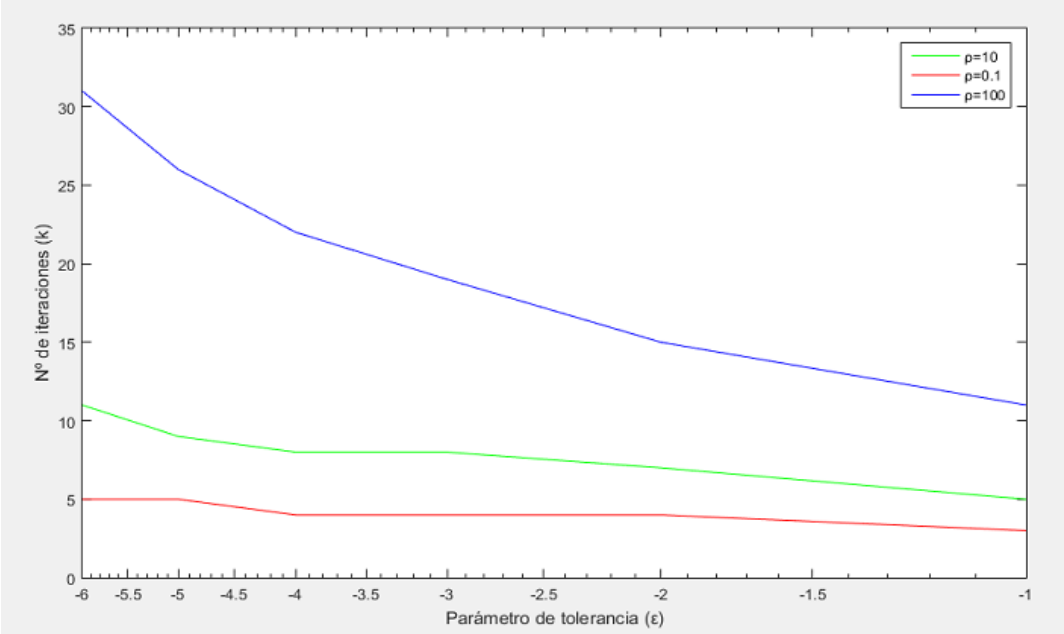
$\rho = 0.1$

COND. INICIAL	TIEMPO DE EJECUCIÓN (sg)
-1.0265	15.7848
2.3128	16.6988
4.8788	20.9689
-7.1629	16.2011
14.5903	16.0353
-21.4026	15.8792

Nº de iteraciones en función del parámetro de tolerancia y del valor de τ .

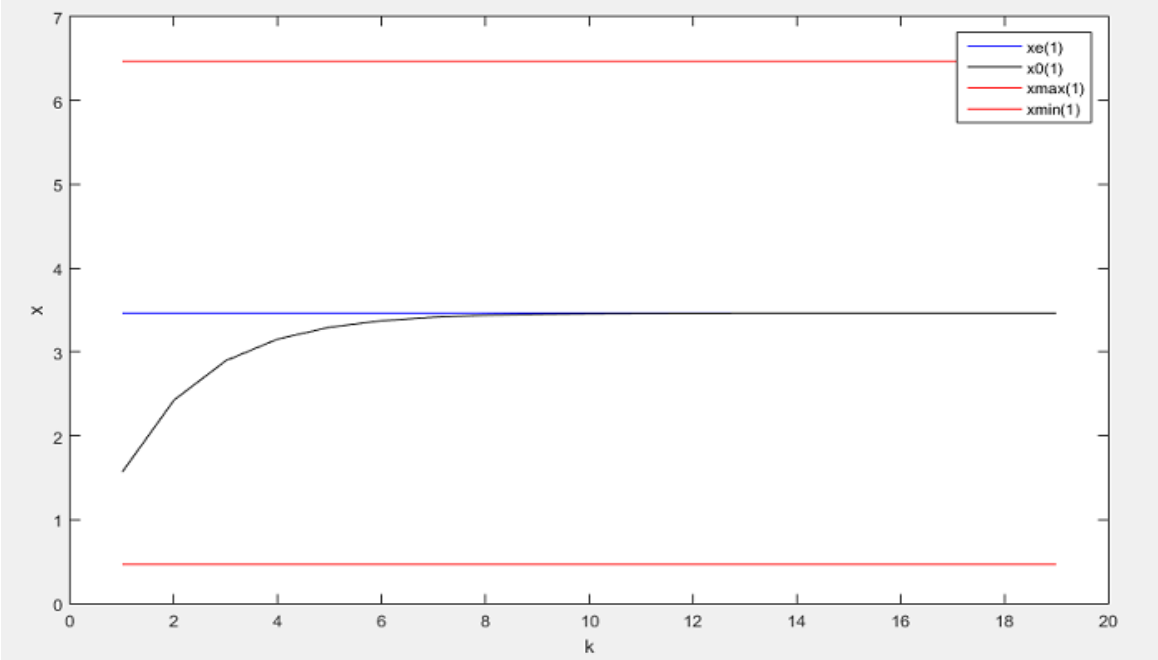


Nº de iteraciones en función del parámetro de tolerancia y del valor de ρ .

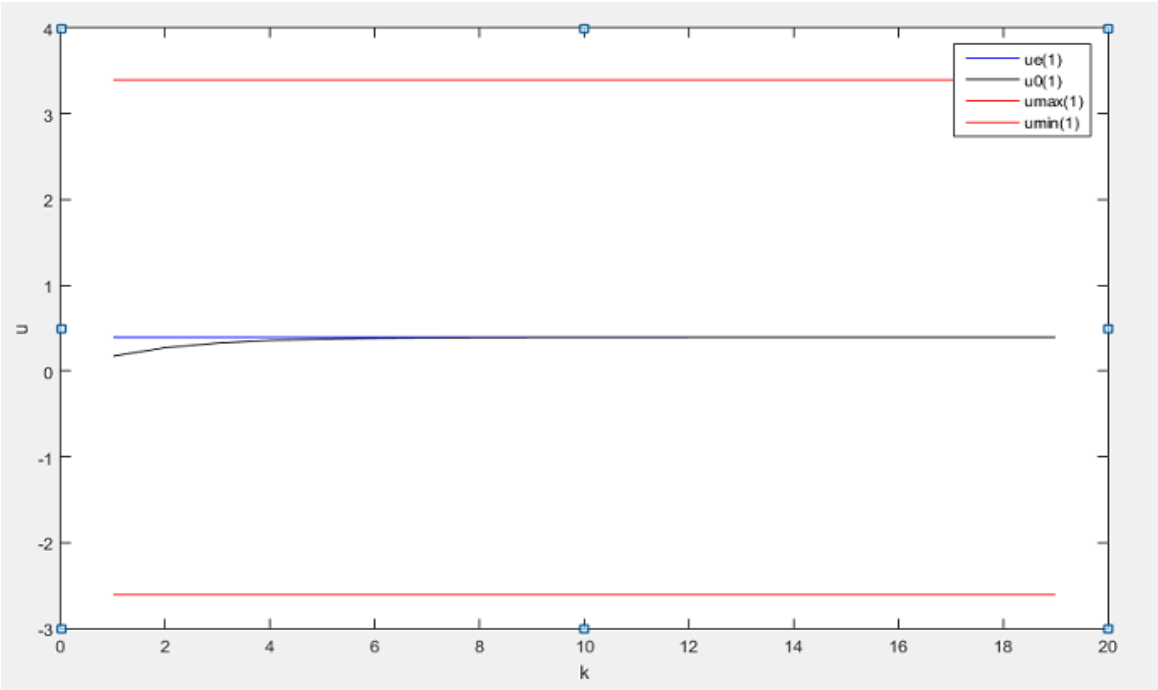


Representación de la evolución de la primera componente de x (x0(1)), viendo que cumple las restricciones y

que converge al punto de equilibrio.



Representación de la evolución de la primera componente de u (u0(1)), viendo que cumple las restricciones y que converge al punto de equilibrio.



ANEXO: CÓDIGOS DE MATLAB

En este anexo expondré el código escrito en MATLAB que he elaborado para llevar a cabo las pruebas numéricas que describo teóricamente durante el trabajo y desarrollar los resultados mostrados en los capítulos anterior.

Algoritmo ISTA con restricciones en caja

Main Function

```
n=50;
xsup=5*ones(n,1);
xinf=-3*ones(n,1);
xoptplus=(xsup+xinf)/2;
xopt=zeros(n,1);
%=========================================================================
%Formación matriz R:
R=randn(n);
R=R*R'+2*eye(n);
%=========================================================================
D=max(eig(R))*eye(n);
q=randn(n,1);
j=0; %Contador para número de iteraciones
epsylon=10^(-8); %Parámetro de tolerancia
%=========================================================================
%Función quadprog:
xopt_qp=quadprog(D,q,[],[],[],[],xinf,xsup);
%=========================================================================
%ALGORITMO ISTA:
while norm(xoptplus-xopt)>epsylon
   j=j+1;
   xopt=xoptplus;
   xoptplus=FUNCTION_ISTA(xopt,D,q,n,xsup,xinf);
end
%Comparación algoritmo ISTA implementado con quadprog:
I(:,:)=[xopt_qp xoptplus];
```

FUNCTION_ISTA

```
function [ xoptplus ] = FUNCTION_ISTA( y,R,q,n,xsup,xinf )
xopt_gorro=zeros(n,1);
xoptplus=zeros(n,1);
d=diag(R);
for i=1:n
   aux=q+(R*y);
   xopt_gorro(i)=y(i)-(1/d(i))*aux(i);
   xoptplus(i)=max(xinf(i),min(xopt_gorro(i),xsup(i)));

end
end
```

Algoritmo FISTA con restricciones en caja y de igualdad

FISTA

```
n=50;
%=========================================================================
%Formación matriz H:

H=randn(n*2);
H=H*H'+2*eye(n*2);
H=max(eig(H))*eye(n*2);
%=========================================================================
A=randn(n,n*2);
b=randn(n,1);
z0=A'*inv(A*A')*b; alpha=0.75; zplus=z0+alpha; zmin=z0-alpha;
q=zeros(n*2,1);
%=========================================================================
%Resultado quadprog:
zopt_qp=quadprog(H,q,[],[],A,b,zmin,zplus);
%=========================================================================
%Inicialización de variables
epsilon=10^(-10);%Parámetro de tolerancia
k=0;
tant=0;
t=tant;
x=zeros(n,1);
xpost=x;
lambda=zeros(n,1);
```

```
z=zeros(n*2,1);

%=========================================================================
%Implementacion algoritmo:
while norm(A*z-b)>epsilon
   tant=t;
   xant=x;
   x=xpost;
    t=0.5*(1+sqrt(1+4*(tant^2)));
    lambda=x+((tant-1)/t)*(x-xant);
    z=ARGMIN(A,lambda,H,zplus,zmin,n);
    xpost=lambda+inv(A*inv(H)*A')*(A*z-b);
    k=k+1;
end
zopt=z;
%=========================================================================

%Comparación algoritmo FISTA implementado con quadprog:
F(:,:)=[zopt_qp zopt];
```

ARGMIN

```
function [ z ] = ARGMIN( A,lambda,H,zplus,zmin,n )
c=A'*lambda;
h=diag(H);
for i=1:(n*2)
z(i)=max(zmin(i),min(-c(i)/h(i),zplus(i)));
end
z=z';
end
```

Formulaciones MPC

Formulación 1

Main program

```
N=100;
n=50;
%Condiciones iniciales:
xplus=200*ones(n,1);
xmin=-65*ones(n,1);
x=zeros(n,N);
x(:,1)=(xplus+xmin)/2;
```

```
uplus=7*ones(n,1);
umin=-8*ones(n,1);
u=zeros(n,N);
%=========================================================================
%Construcción de Q:
Q=randn(n);
Q=Q*Q'+2*eye(n);
Q=max(eig(Q))*eye(n);

%Construcción de R:
R=randn(n);
R=R*R'+2*eye(n);
R=max(eig(R))*eye(n);
%=========================================================================
A=randn(n);
B=randn(n);
C=randn(n);
K=randn(n);

%Punto de equilibrio(xe,ue):

ye=randn(n,1);

s=[(eye(n)-A) -B; C K]\[zeros(n,1);ye];

xe=s(1:n);

ue=s(n+1:2*n);

%Matrices H y D:
I=zeros(n);
H(:,:)=[Q I; I R];
q=zeros(n*2,1);
D(:,:)=[A B];
sum_qp=0;
sum=0;
%Parámetros de tolerancia y contador de iteraciones:
epsilon1=10^(-5);
epsilon2=10^(-5);
k=0;
 z(:,:)=[x;u];
l(:,:)=[xe;ue];
%=========================================================================
%Bucle de control:
while norm(z(:,1)-l)>epsilon2
for i=1:N-1

zplus=[xplus-xe ; uplus-ue];
zmin=[xmin-xe ; umin-ue];
b=x(:,i+1)-xe;

zopt_qp(:,i)=quadprog(H,q,[],[],D,b,zmin,zplus);

f=zeros(n,1);
fpost=f;
lambda=zeros(n,1);
tant=0;
t=tant;
```

```
z(:,i)=[x(:,i)-xe;u(:,i)-ue];
lambda_1=0;
g=0;
%Algoritmo ISTA:
while norm(D*z(:,i)-b)>epsilon1
    z(:,i)=ARGMIN2(D,lambda,H,zplus,zmin,n);
    lambda_1=lambda_1+inv(D*inv(H)*D')*(D*z(:,i)-b);
    g=g+1;
end
%Algoritmo FISTA:
w=0;
while norm(D*z(:,i)-b)>epsilon1
   tant=t;
   fant=f;
   f=fpost;

    t=0.5*(1+sqrt(1+4*(tant^2)));
    lambda=f+((tant-1)/t)*(f-fant);
    z(:,i)=ARGMIN2(D,lambda,H,zplus,zmin,n);
    fpost=lambda+inv(D*inv(H)*D')*(D*z(:,i)-b);
    w=w+1;
end
zopt(:,i)=z(:,i);

xopt_qp(:,i)=zopt_qp(1:n,i)+xe;
uopt_qp(:,i)=zopt_qp(n+1:2*n,i)+ue;

xopt(:,i)=zopt(1:n,i)+xe;
x(:,i)=xopt(:,i);
uopt(:,i)=zopt(n+1:n*2,i)+ue;
u(:,i)=uopt(:,i);

 sum_qp=sum_qp+zopt_qp(:,i)'*H*zopt_qp(:,i);
 sum=sum+zopt(:,i)'*H*zopt(:,i);
end
x(:,1)=x(:,2);
k=k+1;
end
%Comparación algoritmo FISTA implementado con quadprog:
S(:,:)=[uopt_qp(:,1) uopt(:,1)];
F(:,:)=[xopt_qp(:,1) xopt(:,1)];
SUM(:,:)=[sum_qp sum];
```

ARGMIN2

```
function [ z ] = ARGMIN2( D,lambda,H,zplus,zmin,n )
c=D'*lambda;
h=diag(H);
for i=1:(n*2)
z(i)=max(zmin(i),min(-c(i)/h(i),zplus(i)));
end
z=z';
end
```

Formulación 3

Main program

```
N=50;
n=25;
tau=10000000;
epsilon=10^(-3);

%A,B,C y K:
A=randn(n);
B=randn(n);
C=randn(n);
K=randn(n);

%Punto de equilibrio(xe,ue):
ye=randn(n,1);
s=[(eye(n)-A) -B; C K]\[zeros(n,1);ye];
xe=s(1:n);
ue=s(n+1:2*n);

%Valores para restricciones en caja:
xmax=xe+epsilon+3*ones(n,1);
xmin=xe-epsilon-3*ones(n,1);

x_ini=xmax-1; %x dato inicial.

umax=ue+epsilon+3*ones(n,1);
umin=ue-epsilon-3*ones(n,1);

xsmin=xmin+epsilon;
xsmax=xmax-epsilon;

usmin=umin+epsilon;
usmax=umax-epsilon;
```

```
%Restricciones en caja de y:

ymin=zeros((2*N+3)*n,1);
for i=1:2*n:(2*N+2)*n+1
   if i<(2*N+1)*n-n+1
      ymin(i:i+n-1)=xmin;
      ymin(i+n:i+2*n-1)=umin;
   end
   if i>(2*N+1)*n-n && i<(2*N+1)*n+1
      ymin(i:i+n-1)=xmin;
   end
end
for i=(2*N+1)*n+1:n:(2*N+3)*n
   if i>(2*N+1)*n && i<(2*N+2)*n+1
       ymin(i:i+n-1)=xsmin;
   end
   if i>(2*N+2)*n
       ymin(i:i+n-1)=usmin;
   end
end

ymax=zeros((2*N+3)*n,1);
for i=1:2*n:(2*N+2)*n+1
   if i<(2*N+1)*n-n+1
      ymax(i:i+n-1)=xmax;
      ymax(i+n:i+2*n-1)=umax;
   end
   if i>(2*N+1)*n-n && i<(2*N+1)*n+1
      ymax(i:i+n-1)=xmax;
   end
end
for i=(2*N+1)*n+1:n:(2*N+3)*n
   if i>(2*N+1)*n && i<(2*N+2)*n+1
       ymax(i:i+n-1)=xsmax;
   end
   if i>(2*N+2)*n
       ymax(i:i+n-1)=usmax;
```

```
   end
end


%Construcción de Q y de tauQ_eye:
Q=randn(n);
Q=Q*Q'+2*eye(n);
Q=max(eig(Q))*eye(n);
tauQ=min(eig(Q))/2;
tauQ_eye=tauQ*eye(n);

%Construcción de R y de tauR_eye:
R=randn(n);
R=R*R'+2*eye(n);
R=max(eig(R))*eye(n);
tauR=min(eig(R))/2;
tauR_eye=tauR*eye(n);

%Construcción de T y de tauT_eye:
T=350*eye(n); %T=cte*I, la constante la elegí hasta que Hv fuera positiva.
T=T*T'+2*eye(n);
T=max(eig(T))*eye(n);
tauT=min(eig(T))/2;
tauT_eye=tauT*eye(n);

%Construcción de S y de tauS_eye:
S=350*eye(n); %S=cte*I, la hice igual que  T.
S=S*S'+2*eye(n);
S=max(eig(S))*eye(n);
tauS=min(eig(S))/2;
tauS_eye=tauS*eye(n);

%=========================================================================
%Construcción de las matrices H_v, q_v y D:
%H_v:
H_v=zeros((2*N+3)*n);
%H_v:
for i=1:2*n:(2*N+1)*n+1
```

```
   if i<(2*N)*n-n+1
   H_v(i:i+n-1,i:i+n-1)=Q-tauQ_eye;
   H_v((2*N+1)*n+1:(2*N+2)*n,i:i+n-1)=-Q;
   H_v(i+n:i+2*n-1,i+n:i+2*n-1)=R-tauR_eye;
   H_v((2*N+2)*n+1:(2*N+3)*n,i+n:i+2*n-1)=-R;
   end
   if i>(2*N)*n-n && i<(2*N+1)*n+1
      H_v(i:i+n-1,i:i+n-1)=T-tauT_eye;
   end
end
for i=(2*N+1)*n+1:n:(2*N+3)*n
   if i>(2*N+1)*n && i<(2*N+2)*n+1
      for j=1:2*n:(2*N)*n-n
     H_v(j:j+n-1,(2*N+1)*n+1:(2*N+2)*n)=-Q;
      end
      H_v((2*N+1)*n+1:(2*N+2)*n,i:i+n-1)=N*Q+(T-tauT_eye);
   end
   if i>(2*N+2)*n
      for j=1:2*n:(2*N)*n-n
      H_v(j+n:j+2*n-1,(2*N+2)*n+1:(2*N+3)*n)=-R;
      end
      H_v((2*N+2)*n+1:(2*N+3)*n,i:i+n-1)=N*R+(S-tauS_eye);
   end
end
H_v=2*H_v;

%q_v:
q_v=zeros(1,(2*N+3)*n);
for i=1:2*n:(2*N+1)*n+1
   if i<(2*N)*n-n+1
     q_v(i:i+n-1)=2*xe'*tauQ_eye;
     q_v(i+n:i+2*n-1)=2*ue'*tauR_eye;
   end
   if i>(2*N)*n-n && i<(2*N+1)*n+1
     q_v(i:i+n-1)=-2*xe'*(T-tauT_eye);
   end
end
for i=(2*N+1)*n+1:n:(2*N+3)*n
```

```
   if i>(2*N+1)*n && i<(2*N+2)*n+1
      q_v(i:i+n-1)=-2*xe'*(T-tauT_eye);
   end
   if i>(2*N+2)*n
      q_v(i:i+n-1)=-2*ue'*(S-tauS_eye);
   end
end
q_v=q_v';

%D:
gg=-eye(n);
j=0;
D=zeros((N+2)*n,(2*N+3)*n);
   for i=1:n:(N+1)*n+1
      if i<N*n+1
   D(i:i+n-1,i+j*n:i+j*n+n-1)=A;
   D(i:i+n-1,i+j*n+n:i+j*n+2*n-1)=B;
   D(i:i+n-1,i+j*n+2*n:i+j*n+3*n-1)=gg;
   j=j+1;
      end
      if i>N*n && i<(N+1)*n+1
         D(i:i+n-1,i+N*n:i+(N+1)*n-1)=eye(n);
         D(i:i+n-1,i+(N+1)*n:i+(N+2)*n-1)=gg;
      end
      if i>(N+1)*n
        D(i:i+n-1,(2*N+1)*n+1:(2*N+2)*n)=A-eye(n);
        D(i:i+n-1,(2*N+2)*n+1:(2*N+3)*n)=B;
      end
   end



%Construcción de H_y, q_y:
%H_y:
H_y=zeros((2*N+3)*n);
for i=1:2*n:(2*N+1)*n+1
   if i<(2*N+1)*n-n+1
   H_y(i:i+n-1,i:i+n-1)=tauQ_eye;
```

```
   H_y(i+n:i+2*n-1,i+n:i+2*n-1)=tauR_eye;
   end
   if i>(2*N+1)*n-n && i<(2*N+1)*n+1
      H_y(i:i+n-1,i:i+n-1)=tauT_eye;
   end
end
for i=(2*N+1)*n+1:n:(2*N+3)*n
   if i>(2*N+1)*n && i<(2*N+2)*n+1
      H_y((2*N+1)*n+1:(2*N+2)*n,i:i+n-1)=tauT_eye;
   end
   if i>(2*N+2)*n
      H_y((2*N+2)*n+1:(2*N+3)*n,i:i+n-1)=tauS_eye;
   end
end
H_y=2*H_y;


%q_y:
q_y=zeros(1,(2*N+3)*n);
for i=1:2*n:(2*N+1)*n+1
   if i<(2*N+1)*n-n+1
      if i==1
      q_y(i:i+n-1)=-2*xe'*tauQ_eye+tau;
      else
      q_y(i:i+n-1)=-2*xe'*tauQ_eye;
      end
      q_y(i+n:i+2*n-1)=-2*ue'*tauR_eye;
   end
   if i>(2*N+1)*n-n && i<(2*N+1)*n+1
      q_y(i:i+n-1)=-2*xe'*tauT_eye;
   end
end
for i=(2*N+1)*n+1:n:(2*N+3)*n
   if i>(2*N+1)*n && i<(2*N+2)*n+1
      q_y(i:i+n-1)=-2*xe'*tauT_eye;
   end
   if i>(2*N+2)*n
      q_y(i:i+n-1)=-2*ue'*tauS_eye;
```

```
   end
end
q_y=q_y';
```

FISTA

```
% FISTA:

%Inicialización de variables algoritmo:
lambda=0*ones((2*N+3)*n,1);
phi2=0*ones((2*N+3)*n,1);
phi3=0*ones((2*N+3)*n,1);
t1=0; t2=0; k=1;
v_opt=7*ones((2*N+3)*n,1);
y_opt=5*ones((2*N+3)*n,1);
b=1;
while (norm(v_opt-y_opt)>epsilon) && (b==1)
  t1=t2;
  phi1=phi2;
  phi2=phi3;
  t2=0.5*(1+sqrt(1+4*t1^2));
  lambda=phi2+((t1-1)/t2)*(phi2-phi1);
  v_opt=calculo_v(N,n,lambda,q_v,H_v,D);
  [y_opt,q_y]=calculo_y(N,n,q_y,lambda,xe,ue,x_ini,ymin,ymax,tauQ,tauR,tauT,tauS,tau);
  phi3=lambda+(inv(inv(H_v)+inv(H_y))*(v_opt-y_opt));
  [Jv,b]=varphy(H_v,q_v,H_y,q_y,lambda,v_opt,y_opt);
  k=k+1;
end
L=[v_opt y_opt];
```

Calculo_v

```
function [ v_opt ] = calculo_v( N,n,lambda,q_v,H_v,D )
%Matrices de ceros para el sistema de ecuaciones:
m=zeros((N+2)*n);
w=zeros((N+2)*n,1);
```

```
%Matriz de coeficientes:
F=[H_v D';D m];

%Matriz solución:
e=[-q_v-lambda;w];

%Resultado:
o=F\e;
v_opt=o(1:(2*N+3)*n);
end
```

Calculo_y

```
function [ y_opt,q_y ] = calculo_y( N,n,q_y,lambda,xe,ue,x,ymin,ymax,tauQ,tauR,tauT,tauS,tau )
%x -> dato inicial.
%ymin,ymax -> las restricciones en caja, están construidas en el main.
for i=1:(2*N+3)*n  %bucle que recorre el vector y.
   %El siguiente if es para ir haciendo que j vaya de 1 a n cada vez que
   %llegamos a una nueva variable de decisión:
if (i<n+1) %Si el índice es menor que n+1, entonces estoy en x(0).
   j=i;
else
   j=mod(i,n); %Si el índice es superior, miro el resto de dividir i entre n
   %y la j será igual a ese resto, excepto si es cero.
   if (j==0) %Si el resto es 0, j=n.
      j=n;
   end
end

if (mod(i,2*n)<=n) && (mod(i,2*n)~=0) %Para ver la variable de decisión que estoy optimizando es
x(1,2,...,N-1).
  %Los parámetros son los que acompañan a los términos de x en el
  %funcional.
d=xe(j);
r=2*tauQ;
if (i>2*N*n) && (i<(2*N+1)*n+1) %Si es x(N), r es otra.
  r=2*tauT;
end
else %Si estoy optimizando una de las u(1,2,...,N-1).
d=ue(j);
r=2*tauR;
end
%Lo que sigue es para las 2 últimas variables, xs e us, que por el orden en
%el que están puestas en y, no corresponden al lugar que se le asignaría de
%la forma anterior.
if (i>(2*N+1)*n) && (i<(2*N+2)*n+1)
```

```
   d=xe(j);
   r=2*tauT;
end
if (i>(2*N+2)*n) && (i<(2*N+3)*n+1)
   d=ue(j);
   r=2*tauS;
end

%Los otros parámetros tal como vimos.
g=-lambda(j);%Este signo es negativo por la forma en que definimos el varphy.
a=tau;
s=x(j);

%Así, ya quedaría definido el funcional como 0.5(z-d)^2+a|z-s|+gz.

   if (i<n+1)  %Si estamos en x(0) -> a es distinto de 0.
      y_=max(ymin(i),min(((-a-g)/r)+d,ymax(i))); %Suponemos x0(j)-x(j)>0
      y__=max(ymin(i),min(((a-g)/r)+d,ymax(i))); %Suponemos x0(j)-x(j)<0

     if ((y_-s)<0) %Si ambas son negativas, cogemos la que se calculó suponiendo eso:
       y_opt(i)=y__;
       alpha(i)=-1;
     end
     if ((y__-s)>0) %Si ambas son positivas, lo mismo:
       y_opt(i)=y_;
       alpha(i)=1;
     end
  else  %Si no estamos en x(0)-> a=0.
     y_opt(i)=max(ymin(i),min((d-g/r),ymax(i)));
  end
end

q_y(1:n)=q_y(1:n)+tau*alpha';

y_opt=y_opt';
end
```

ADMM

```
%Algortimo ADMM:

lambda=0*ones((2*N+3)*n,1);
rho=100;
y_ant=(ymax+ymin)/2;
k=1;

v_post=7*ones((2*N+3)*n,1);
y_post=0*ones((2*N+3)*n,1);
```

```
while max(norm(v_post-y_post),norm(y_post-y_ant))>epsilon
  y_ant=y_post;
  v_post=inv(H_v+rho*eye((2*N+3)*n))*(-lambda+rho*y_ant-q_v-D'*...
  inv(D*inv(H_v+rho*eye((2*N+3)*n))*D')*D*inv(H_v+rho*eye((2*N+3)*n))*...
  (-lambda+rho*y_ant-q_v));
  f=rho*v_post-lambda;
  y_post=max(ymin,min(f/rho,ymax));
  lambda=lambda+rho*(v_post-y_post);
  k=k+1;
end
```

REFERENCIAS

[1] Pablo Krupa, Daniel Limon and Teodoro Alamo, "Implementation of Model Predictive Control in Programmable Logic Controllers ", IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, pp. 1-12, 2020.

[2] Pablo Krupa, Ignacio Alvarado, Daniel Limon and Teodoro Alamo, "Implementation of model predictive control for tracking in embedded system using a sparse extended ADMM algorithm", pp. 2-8, 2020.

[3] Pablo Krupa García, "Implementation of MPC in embedded systems using first order methods", *Tesis Doctoral,* pp. 9-135, 2021.

[4] Adrián Ayastuy Rodríguez, "Optimización convexa aplicada a MPC", *Proyecto Fin de Carrera*, pp. 3-57, 2014.

[5] Stefan Richter, Colin Neil Jones and Manfred Morari, "Computational Complexity Certification for Real-Time MPC With Input Constraints Based on the Fast Gradient Method ", IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 57, NO. 6, pp. 1391-1401, 2012.

[6] Teodoro Alamo, Pablo Krupa and Daniel Limon, "Gradient Based Restart FISTA ", *Department of Systems Engineering and Automation, University of Seville,* 2019.

[7] Teodoro Álamo, "Optimización Convexa", *Escuela Técnica Superior de Ingeniería, Universidad de Sevilla.*

[8] Teodoro Álamo, "FISTA", *Escuela Técnica Superior de Ingeniería, Universidad de Sevilla.*

[9] Teodoro Alamo, Pablo Krupa and Daniel Limon, "Restart FISTA with Global Linear Convergence", *Department of Systems Engineering and Automation, University of Seville*, 2019.