ELSEVIER

Disponible en www.sciencedirect.com

ScienceDirect

Revista Iberoamericana de Automática e Informática industrial 8 (2011) 167–181

RIAI

www.elsevier.es/RIAI

Solución Explícita al Control Predictivo de Sistemas Lineales Sujetos a Restricciones No Convexas

Emilio Pérez[a,*], Carlos Ariño[a], F.Xavier Blasco[b], Miguel A. Martínez[b]

[a] *Departamento de Ingeniería de Sistemas Industriales y Diseño, Universitat Jaume I, Avenida Vicent Sos Baynat, s/n. 12071 Castelló de la Plana, España*
[b] *Instituto Universitario de Automática e Informática Industrial. Universidad Politécnica de Valencia. Camino de Vera S/N 46022 - Valencia, España*

Resumen

Este trabajo propone una solución explícita para el control predictivo de sistemas lineales sujetos a restricciones poliédricas no convexas, modeladas como la unión de un número finito de poliedros. El algoritmo se basa en el cálculo de la solución explícita de los problemas sujetos a las restricciones convexas definidas por dichos poliedros. Las regiones de las particiones así obtenidas se intersectan de forma que el nuevo conjunto de regiones tiene tantas soluciones posibles como problemas convexos se han resuelto. Mediante programación de suma de cuadrados se eliminan aquellas soluciones de cada región que no son óptimas para ningún estado. Posteriormente se realiza la unión de las regiones que compartan el mismo conjunto de soluciones. Tras la descripción de la metodología descrita, se incluye una justificación de ésta. Además, se incluye una posible solución subóptima utilizable cuando la metodología original es demasiado costosa. Por último, se muestran los resultados obtenidos en un ejemplo. *Copyright © 2011 CEA.*

Palabras Clave:
Control Predictivo, Programación multiparamétrica, Restricciones no convexas, Suma de cuadrados.

1. Introducción

El control predictivo ha demostrado su potencial y ha sido ampliamente aceptado en el control de procesos multivariables a nivel industrial, especialmente debido a su capacidad de incorporar en el diseño el tratamiento de restricciones (Camacho and Bordons, 2004). Su espectro de aplicación se ha extendido a todo tipo de procesos pudiendo considerase por un lado el control predictivo no lineal sujeto a restricciones no lineales, y por el otro el control predictivo lineal con restricciones poliédricas.

En el primer caso, es necesario recurrir a algoritmos en línea de optimización no lineal (y en general no convexa) bastante costosos computacionalmente (Camacho and Bordons, 2007) por lo que sólo podrá ser aplicado a procesos que admitan períodos de muestreo relativamente altos.

En el extremo opuesto, el control de procesos lineales sujetos a restricciones poliédricas, existen diversos algoritmos eficientes en la literatura, destacando especialmente (Bemporad et al., 2002) en el que se obtiene fuera de línea una solución explícita definida sobre regiones poliédricas. Existen además técnicas en la literatura que permiten reducir considerablemente el tiempo computacional en línea de dicha solución (Tøndel et al., 2003b), haciendo la filosofía de control predictivo factible para procesos considerablemente rápidos.

Si se desea trabajar con períodos de muestreo relativamente bajos, una generalización del control predictivo lineal que permite ampliar su aplicabilidad a diferentes problemas se consigue utilizando modelos lineales con restricciones poliédricas no convexas, definidas como la unión no convexa de un número finito de poliedros. Este tipo de restricciones aparecen de forma natural en problemas como el de evitación de obstáculos, inherentemente no convexo. La importancia de este problema se muestra en (Kurzhanski, 2005), apareciendo en muchos problemas prácticos de ingeniería de control, como la planificación de trayectorias de robots (López et al., 2006) o planificación de tráfico aéreo (Kuchar and Yang, 2000). Existen enfoques de control predictivo para este tipo de sistemas (Rakovic and Mayne, 2007), pero requieren la resolución en línea de problemas de programación mixta cuadrática-entera, que pueden requerir un tiempo en línea demasiado elevado.

Otro campo de aplicación en el que aparecen restricciones no convexas surge a partir de los sistemas no lineales con restricciones lineales. Dentro de este ámbito se encuentran los sistemas de tipo Hammerstein, definidos por una no linealidad

*Autor en correspondencia
Correos electrónicos: pereze@esid.uji.es (Emilio Pérez), arino@esid.uji.es (Carlos Ariño), xblasco@isa.upv.es (F.Xavier Blasco), mmiranzo@isa.uji.es (Miguel A. Martínez)

doi:10.1016/j.riai.2011.06.005

estática seguida de un modelo lineal. Estos modelos suelen controlarse mediante la inversión de la no linealidad, transformando el problema en uno de control de un proceso lineal, al que se le puede aplicar el control predictivo estándar, (Fruzzetti et al., 1997) y (Patwardhan et al., 1998). No obstante, cuando existen restricciones lineales sobre las entradas o variables de estado, deben ser transformadas también por la función no lineal para trasladarlas al espacio de variables en el que trabaja el controlador, pudiendo ser no convexas. En este caso, puede conseguirse una aproximación interior suficientemente precisa mediante el uso de varios poliedros cuya unión no es convexa (Bertsekas and Yu, 2009).

Otro problema del mismo tipo también pueden aparecer en sistemas no lineales con restricciones lineales controlados mediante linealización por realimentación (Slotine and Li, 1991). Dichas restricciones lineales son no convexas para el modelo linealizado (Pappas et al., 1995), por lo que de nuevo pueden ser aproximadas mediante unión no convexa de poliedros.

Una forma de tratar los sistemas lineales con restricciones poliédricas no convexas es modelándolos como sistemas afines a tramos. El control óptimo y predictivo de este tipo de modelos, que también permiten el modelado de un amplio espectro de sistemas híbridos (Heemels et al., 2001), ha sido un campo de investigación muy activo en los últimos años.

Una solución explícita completa al problema de control óptimo con horizonte de control finito e índice de coste cuadrático puede encontrarse en (Borrelli et al., 2005), donde los autores muestran que la partición del espacio de estados no es en general poliédrica, pero sí que se puede definir mediante inecuaciones lineales y cuadráticas. El procedimiento se basa en la resolución de múltiples problemas de optimización cuadrática multiparamétrica (mpQP). Sin embargo, la simplificación de la solución obtenida no se trata de forma sistemática, lo que resulta en algoritmos en línea costosos. Otra solución al problema se propuso en (Mayne and Rakovic, 2002) donde se tratan todas las posibles secuencias de control y para cada una de ellas el modelo afín definido a tramos puede tratarse y resolverse como un sistema variante en el tiempo. Esta aproximación implica la resolución de múltiples mpQP y el valor de la función de coste para cada una de las posibles secuencias debe de compararse en línea, lo que en general aumenta considerable el tiempo de cómputo en línea.

A pesar que los sistemas sujetos a restricciones poliédricas no convexas pueden modelarse como sistemas afines definidos a tramos, explotar la estructura del problema a resolver puede aportar varias ventajas significativas, especialmente en cuánto al coste computacional en línea y los requerimientos de memoria.

En la siguiente sección, se presentan algunos conceptos preliminares y herramientas necesarias. A continuación se introduce un enunciado formal del problema a resolver. En las dos secciones siguientes se desarrolla la metodología para la obtención de la solución explícita del problema propuesto. Por último, se muestran los resultados en un ejemplo y se ofrecen algunas conclusiones.

2. Preliminares

En esta sección, en primer lugar se introduce la filosofía del control predictivo basado en modelos, así como la solución explícita de éste para el caso de sistemas lineales con restricciones poliédricas. A continuación, se presenta la optimización de sumas de cuadrados como herramienta para el estudio de la factibilidad de sistemas de ecuaciones y desigualdades polinómicas.

2.1. *Control predictivo basado en modelos. Solución explícita.*

El control predictivo basado en modelos resuelve problemas de regulación definiendo una función de coste y resolviendo el problema de control óptimo de horizonte finito (1) para posteriormente aplicar una estrategia de horizonte móvil.

$$\begin{aligned}\mathcal{P}_N(x): \quad & V_N^{OPT}(x) := \min V_N(\{x_k\},\{u_k\}),\\ & \text{sujeto a:}\\ & x_{k+1} = f(x_k,u_k) \text{ para } k=0,\dots,N-1,\\ & x_0 = x,\\ & u_k \in \mathbb{U} \text{ para } k=0,\dots,M-1,\\ & u_k = 0 \text{ para } k=M,\dots,N-1,\\ & x_k \in \mathbb{X} \text{ para } k=1,\dots,N,\\ & x_N \in \mathbb{X}_f \subset \mathbb{X},\\ & V_N(\{x_k\},\{u_k\}) := F(x_N) + \textstyle\sum_{k=0}^{N-1} L(x_k,u_k)\end{aligned} \tag{1}$$

donde $L(x,u)$, $F(x)$ y $\mathbb{X}_f$ son, respectivamente, el coste de etapa, la ponderación terminal del estado y la región terminal.

La solución que minimiza el problema de optimización anterior es una secuencia de control dependiente del estado actual x

$$\mathbf{u}^{opt} := \{u_0^{\text{opt}}, u_1^{\text{opt}}, \dots, u_{N-1}^{\text{opt}}\}$$

y, por la estrategia de horizonte móvil, la acción de control aplicada a la planta es únicamente el primer elemento de la secuencia

$$\mathcal{K}_N(x) = u_0^{\text{opt}} \tag{2}$$

A continuación, se considera el sistema discreto lineal e invariante en el tiempo de la forma (3) y con restricciones poliédricas en las acciones de control y los estados:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \tag{3}$$

Usualmente, con objeto de conseguir una buena respuesta del sistema en bucle cerrado con el control predictivo, en la función de coste se elige un coste de etapa $L(x,u)$ cuadrático. Con esta elección, por razones de estabilidad, la ponderación terminal se escoge también como una función cuadrática (Mayne et al., 2000):

$$V_N(x) = \frac{1}{2}x_N^T P x_N + \frac{1}{2}\sum_{k=0}^{N-1}(x_k^T Q x_k + u_k^T R u_k) \tag{4}$$

Usando la ecuación del sistema (3), la función de coste y las restricciones sobre estados futuros pueden escribirse como función del estado actual y la secuencia de control, y por tanto

formularse en el espacio de entrada. De esta forma, el problema de optimización a resolver es:

$$\begin{aligned}\mathcal{P}_N(x) : V_N^{OPT}(x) := \min\ & \frac{1}{2}\mathbf{u}^T H\mathbf{u} + \mathbf{u}^T Fx \\ & \text{sujeto a:} \\ & \Phi\mathbf{u} \leq \Delta - \Lambda x\end{aligned}$$

Aunque el problema así planteado está en la forma de una optimización cuadrática y puede ser resuelto en línea mediante algoritmos QP, en (Bemporad et al., 2002) se muestra como el optimizador $\mathbf{u}^{opt}(x)$ tiene una estructura afín a tramos de la forma

$$\mathbf{u}^{opt}(x) = G_i x + h_i \text{ para } x \in X_i = \{x | H_i x \leq L_i\} \tag{5}$$

definida en l regiones X_i, $i = 1, \ldots l$, que forman una partición poliédrica del espacio de trabajo.

2.2. Sistemas de ecuaciones polinomiales y suma de cuadrados

Se considera el conjunto de polinomios multivariable con coeficientes reales en las variables $\{x_1, \ldots, x_n\}$, $\mathbb{R}[x]$, y un sistema de ecuaciones y desigualdades de la forma (6)

$$\begin{cases} f_i(x) = 0, & (i = 1, \ldots, m) \\ g_j(x) \geq 0, & (j = 1, \ldots, p) \end{cases} \tag{6}$$

donde $f_i(x), g_j(x) \in \mathbb{R}[x]$.

Con objeto de estudiar la factibilidad del sistema (6), se introduce la siguiente definición:

Definición 1. *Un polinomio $p(x) \in \mathbb{R}[x]$ es* suma de cuadrados *(SOS) si existe una descomposición de la forma:*

$$p(x) = \sum_i p_i^2(x)$$

donde $p_i(x) \in \mathbb{R}[x]$.

A continuación, se presenta el Positivstellensatz, teorema propuesto por (Stengle, 1973) y enunciado aquí en la forma de (Bochnak et al., 1998):

Teorema 1 (Positivstellensatz). *El sistema* (6) *no es factible si y sólo si $\exists F(x), G(x) \in \mathbb{R}[x]$ tales que:*

$$\begin{cases} F(x) + G(x) = -1 \\ F(x) \in \textbf{\textit{ideal}}(f_1, \ldots, f_m) \\ G(x) \in \textbf{\textit{cono}}(g_i, \ldots, g_p) \end{cases} \tag{7}$$

donde:

$$\textbf{\textit{ideal}}(f_1, \ldots, f_m) := \left\{ f \;\middle|\; f = \textstyle\sum_{i=1}^m t_i f_i, \quad t_i \in \mathbb{R}[x] \right\}$$

$$\textbf{\textit{cono}}(g_1, \ldots, g_p) := \left\{ g \;\middle|\; g = s_0 + \sum_i s_i g_i + \sum_{i,j} s_{ij} g_i g_j + \sum_{i,j,k} s_{ijk} g_i g_j g_k + \ldots \right\}$$

y $s_\alpha \in \mathbb{R}[x]$ son polinomios suma de cuadrados.

El teorema 1 proporciona alternativas excluyentes, es decir, si se encuentran polinomios $F(x)$ y $G(x)$ que satisfagan (7), se tiene una prueba que refuta la consistencia de (6). No obstante, el teorema en sí mismo no ofrece ninguna manera de obtener dichos polinomios. Para obtenerlos, en (Parrilo, 2000) se introduce la programación de suma de cuadrados (SOS) y se demuestra que, para un grado acotado de los polinomios t_i y s_α, pueden formularse como un problema de programación semidefinida (SDP) de la forma:

$$\begin{aligned} &\text{buscar} && s_i, t_i \\ &\text{sujeto a:} && s_i \text{ suma de cuadrados} \\ & && F(x) + G(x) = -1 \\ & && F(x) = \textstyle\sum_{i=1}^m t_i f_i(x) \\ & && G(x) = s_0 + \sum_i s_i g_i(x) + \sum_{i,j} s_{ij} g_i(x) g_j(x) + \ldots \end{aligned} \tag{8}$$

Este tipo de problemas de optimización, que han sido ampliamente usados en el ámbito de la teoría de sistemas y control (Boyd et al., 1994), son convexos (Vandenberghe and Boyd, 1996), y cuentan con *solvers* eficientes, como (Sturm, 1999).

Es importante tener en cuenta que, dado que el orden de los polinomios t_i y s_α está acotado, el teorema 1 permite únicamente obtener pruebas que refuten (6) cuando se encuentran polinomios $F(x)$ y $G(x)$, pero su factibilidad no puede ser asegurada en caso contrario (sería necesario un orden infinito de dichos polinomios).

3. Formulación del problema

Considérese el problema de regulación consistente en llevar al origen el sistema discreto lineal e invariante en el tiempo (3) con restricciones sobre las acciones de control y los estados definidas como la unión no convexa de un número finito de poliedros.

$$u_k \in \bar{\mathbb{U}} = \textstyle\bigcup_{i=0}^{\gamma_u} \mathbb{U}_i, \quad x_k \in \bar{\mathbb{X}} = \bigcup_{i=0}^{\gamma_x} \mathbb{X}_i$$

Planteando una estructura de control predictivo (1)-(2) con una función de coste cuadrática de la forma (4) puede escribirse

$$\begin{aligned}\mathcal{P}_N(x) : V_N^{OPT}(x) := \min\ & \frac{1}{2}\mathbf{u}^T H\mathbf{u} + \mathbf{u}^T Fx \\ & \text{sujeto a:} \\ & u_k \in \bar{\mathbb{U}} \text{ para } k = 0, \ldots, M-1, \\ & u_k = 0 \text{ para } k = M, \ldots, N-1, \\ & x_k = f_k(x, u_0, \ldots, u_{k-1}) \in \bar{\mathbb{X}} \text{ para } k = 1, \ldots, N, \\ & x_N = f_N(x, \mathbf{u}) \in \bar{\mathbb{X}}_f \subset \mathbb{X}\end{aligned} \tag{9}$$

donde f_k refleja la dependencia del estado en cada período de muestreo de las acciones de control previas y $\bar{\mathbb{X}}_f$ el conjunto de restricciones terminal que garantiza estabilidad. Para el sistema planteado, dicho conjunto $\bar{\mathbb{X}}_f$ es un poliedro no convexo. Un procedimiento para su cálculo puede consultarse en (Pérez et al., 2011).

Una vez que todas las restricciones están expresadas en términos de las variables de optimización u_k, el objetivo es combinarlas todas en un único poliedro no convexo ($\mathbb{T}$). Para ello,

debe tenerse en cuenta que los conjuntos de restricciones son uniones de poliedros, y todas las posibles combinaciones deben considerarse. No obstante, muchas de estas combinaciones pueden ser no factibles. Por ello, con objeto de simplificar el problema de optimización a resolver, es conveniente comprobar la factibilidad, mediante la resolución de un problema de programación lineal, de los posibles conjuntos de restricciones. A continuación, puede aplicarse un algoritmo para la unión de regiones que minimice el número de poliedros convexos como el mostrado en la sección 4.4. Una vez $\mathbb{T}$ está definido como la unión del mínimo número de poliedros, se puede formular el siguiente problema de optimización:

$$\begin{aligned} \mathcal{P}_{\mathbb{T}}(x): \quad & V_N^{OPT}(x) := \min \tfrac{1}{2}\mathbf{u}^T H\mathbf{u} + \mathbf{u}^T F x, \\ & \text{sujeto a:} \\ & (\mathbf{u}, x) \in \mathbb{T} = \textstyle\bigcup_{i=0}^{\gamma} T_i \end{aligned} \tag{10}$$

donde:

$$T_i := \left\{ (\mathbf{u}, x) \in \mathbb{R}^{Mm+n} \left| \begin{bmatrix} \Phi_i & \Lambda_i \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ x \end{bmatrix} \leq \Delta_i \right. \right\} \tag{11}$$

La manera más directa de resolver el problema (10) consiste en reescribirlo de la forma (12), resolver los problemas $\mathcal{P}_{T_i}$ y comparar en cada instante de muestreo el valor de la función de coste para cada óptimo $\mathbf{u}_{T_i}^{opt}$.

$$\begin{aligned} & \mathcal{P}_{\mathbb{T}}(x): \quad V_N^{OPT}(x) := \min \left\{ V_{iN}^{OPT}(x) \right\} \\ & \text{donde:} \\ & \mathcal{P}_{T_i}(x): \quad V_{iN}^{OPT}(x) := \min \tfrac{1}{2}\mathbf{u}^T H\mathbf{u} + \mathbf{u}^T F x, \\ & \qquad \text{sujeto a:} \\ & \qquad (\mathbf{u}, x) \in T_i \end{aligned} \tag{12}$$

Aunque este enfoque, equivalente al de (Mayne and Rakovic, 2002) para sistemas afines a tramos, es el más sencillo, tiene el importante inconveniente de ser considerablemente costoso en términos de coste computacional en línea y requerimientos de memoria, como se verá en la sección 6. Por ello, se propone el enfoque que se presenta en las siguientes secciones.

4. Solución explícita para $\gamma = 2$

En esta sección se plantea una metodología para la obtención de la solución explícita al problema (10) cuando las restricciones están formadas por la unión no convexa de dos poliedros ($\gamma = 2$). Dicha metodología está resumida en el algoritmo 1.

A continuación, se describen los procedimientos para llevar a cabo cada uno de los pasos de dicho algoritmo.

4.1. Obtención de la solución explícita de los subproblemas

Se trata de resolver, de manera explícita, los dos problemas de programación cuadrática que nos dan el posible mínimo a nuestro problema de optimización (12). Como se ha visto en la sección 2, dicha solución es una función afín a tramos sobre una partición lineal del espacio de estados (5).

Algoritmo 1 Obtención de la solución explícita para $\gamma = 2$

1. Obtener la solución explícita de los problemas de optimización con restricciones convexas.
2. Intersectar las dos particiones del espacio de estados obtenidas.
3. Comprobar si es necesario dividir las regiones resultantes en función de cual de las dos posibles soluciones es la óptima.
   a) Si es necesario, obtener las dos subregiones.
   b) Si no es necesario, comprobar si la región puede unirse con alguna de las adyacentes (en el caso de que tenga la misma solución afín).

Dependiendo de las restricciones de cada uno de los dos problemas, el conjunto de estados iniciales factibles, puede ser diferente. En ese caso, para algunos valores de x, se tendrá solución para uno de los problemas, pero no para el otro. Puesto que el algoritmo 1 requiere la intersección de las particiones se añadirán a cada solución explícita nuevas regiones sin ninguna ley de control asociada. Si no se realiza esta operación la partición final obtenida cubrirá únicamente los puntos en los que ambos problemas tienen solución. Por lo tanto, será necesario calcular el complemento de cada uno de los conjuntos de estados iniciales factibles e intersectarlo con el otro conjunto. Esto, en general, resultará en un poliedro no convexo definido por un número no mínimo de elementos, por lo que será conveniente aplicar alguno de los algoritmos de (Geyer et al., 2008), introducidos en la sección 4.4. De esta forma, el número de nuevas regiones X_{ij} que es necesario añadir, será mínimo.
Tras finalizar este procedimiento, la solución explícita para cada una de los dos problemas queda de la siguiente forma:

$$\mathcal{K}_{iN}(x) = \begin{cases} G_{ij}x + h_{ij} \mid x \in X_{ij},\ i = 1,2;\ j = 0, \ldots, N_i \\ \emptyset \mid x \in X_{ij},\ i = 1,2;\ j = N_i + 1, \ldots, N_{ri} \end{cases}$$

donde:

$$X_{ij} := \left\{ x \in \mathbb{R}^n \left| L_{ij}x \leq W_{ij} \right. \right\}$$

4.2. Intersección de las dos particiones del estado

Una vez obtenida la solución explícita para los dos controladores predictivos se tienen dos particiones del espacio de estados, cada una de ellas con una solución afín correspondiente a cada región. Para obtener una partición única, en primer lugar se deben intersectar todas las regiones poliédricas que describen ambas particiones obteniéndose:

$$X_r = X_{1i} \cap X_{2j} = \left\{ x \in \mathbb{R}^n \left| \begin{bmatrix} L_{1i} \\ L_{2j} \end{bmatrix} x \leq \begin{bmatrix} W_{1i} \\ W_{2j} \end{bmatrix} \right. \right\}, \quad r = 0 \ldots N_{r1} \cdot N_{r2}$$

En general, muchas de las $N_{r1} \cdot N_{r2}$ regiones así definidas pueden estar vacías. Puede comprobarse su consistencia resolviendo el

siguiente problema de programación lineal:

$$\begin{aligned} &\min\ s \\ &\text{sujeto a:}\quad \begin{bmatrix} L_{1i} \\ L_{2j} \end{bmatrix} x - \begin{bmatrix} W_{1i} \\ W_{2j} \end{bmatrix} \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} s \end{aligned} \tag{13}$$

siendo el espacio definido por las desigualdades de dimensión completa siempre que $s < 0$. En este caso, además, la resolución del problema nos proporciona un punto factible, x_f.

Con la resolución de estos $N_{r1} \cdot N_{r2}$ problemas de programación lineal, se eliminarán todas las regiones X_r que no sean consistentes. No obstante, las regiones restantes pueden estar definidas por ecuaciones redundantes, puesto que hasta el momento se han incluido todas las restricciones que definen a ambas regiones de origen.

Aplicar un procedimiento genérico para la eliminación de redundancias en cada una de las regiones es costoso computacionalmente. No obstante, es sencilla su eliminación si se tienen presentes algunas propiedades geométricas de las dos particiones del espacio de estados. Para ello, se introducen en primer lugar las siguientes definiciones:

Definición 2. *Dado un poliedro $X \in \mathbb{R}^n$ definido por las desigualdades lineales $Lx \leq W$ tómese uno de los hiperplanos $h := \{x | L^i x = W^i\}$. Si se cumple que $\mathcal{F} = X \cap h$ es de dimensión $n-1$, se dice que $\mathcal{F}$ es una faceta del poliedro.*

Definición 3. *Dos poliedros X_1 y X_2 son adyacentes si $dim(X_1 \cap X_2) = n - 1$.*

Para el caso particular que nos ocupa, determinar si son adyacentes dos regiones pertenecientes a una misma partición solución a un problema de control predictivo explícito puede plantearse de diversas maneras.

En primer lugar, puede demostrarse que bajo el cumplimiento de ciertas condiciones para una determinada faceta de una región, enunciadas en el teorema 2 de (Tøndel et al., 2003a), existe una única región adyacente a ésta en la partición, y difiere únicamente en una restricción activa. Por tanto, cuando estas condiciones se cumple, buscar las regiones adyacentes supone un muy bajo coste computacional.

Por contra,en aquellos casos en que las condiciones del teorema no se cumplen, para encontrar las regiones adyacentes a una dada por cada faceta se puede aplicar el algoritmo de exploración y partición de (Bemporad et al., 2002), que requiere la resolución de un problema de programación lineal para cada faceta.

En cualquiera de ambos casos, el hecho fundamental es que la mayoría de algoritmos para la obtención de la solución explícita de un problema con restricciones convexas (y en particular los dos citados anteriormente) obtienen información de la adyacencia de regiones durante el procedimiento de exploración del espacio de estados. Por tanto, simplemente con almacenar dicha información, es posible conocer qué regiones de la partición son adyacentes sin necesidad de resolver un sólo problema de optimización adicional.

Una vez se conoce la información de qué regiones de las dos particiones que se tienen son adyacentes, se puede enunciar la siguiente proposición, útil para la eliminación de ecuaciones redundantes:

Proposición 1. *Considérese una región $X_{1i} \in \mathbb{X}_1$ y el conjunto de todas las regiones de la misma partición adyacentes a X_{1i} a través de una misma faceta $\mathcal{F}$:*

$$ad_{iF} = \left\{ j \;\middle|\; \begin{array}{c} X_{1j} \in \mathbb{X}_1 \\ dim(X_{1j} \cap \mathcal{F}) = n-1 \end{array} \right\}$$

y una región X_{2i} correspondiente a otra partición diferente. Si se cumple:

$$\begin{cases} X_{1i} \cap X_{2i} \neq \emptyset \\ X_{1j} \cap X_{2i} = \emptyset \quad \forall j \in ad_{iF} \end{cases}$$

el hiperplano que define $\mathcal{F}$ será redundante en $X_{1i} \cap X_{2i}$.

Prueba 1. *La proposición puede deducirse de una propiedad que resulta evidente de la definición de regiones adyacentes: no hay ningún punto de $\mathcal{F}$ que no pertenezca a alguna de las regiones X_{1j} para $j \in ad_{iF}$. Por tanto, si se tiene que $X_{1j} \cap X_{2i} = \emptyset$ para todas las $j \in ad_{iF}$, $\mathcal{F} \cap X_{2i} = \emptyset$, y por tanto el hiperplano que define $\mathcal{F}$ es redundante en X_{1i}.* ■

La proposición 1 es de utilidad para la eliminación de algunas de las ecuaciones redundantes en las regiones de intersección si se conoce cuáles son las regiones adyacentes a cada región de las dos particiones y se ha comprobado previamente ,mediante (13), la consistencia de todas las posibles intersecciones de regiones de las dos particiones. De esta forma, sabemos qué parejas de regiones (X_{1i}, X_{2j}) tienen intersección vacía y cuales no. Haciendo uso de la proposición y buscando regiones adyacentes, es posible eliminar algunos de los hiperplanos redundantes.

No obstante, con este procedimiento todavía pueden quedar ecuaciones redundantes que no se hayan eliminado: aquellos hiperplanos correspondientes a facetas de una de las regiones intersectadas a través de las cuales hay regiones adyacentes que tienen intersección no vacía con la región de la segunda partición intersectada. En este tipo de casos, como se aprecia en el ejemplo 1, a pesar de tener información a priori de la consistencia de regiones no puede decirse nada respecto a la redundancia de alguna de sus facetas.

Ejemplo 1. *Se tiene una región X_{12} con tres regiones de su misma partición adyacentes, X_{11} y X_{13} a través de la faceta $\mathcal{F}_1$ y X_{14} a través de la faceta $\mathcal{F}_2$ (figura 1). Por otro lado, se tiene una región correspondiente a otra partición, X_{21} que cumple $X_{12} \cap X_{21} \neq \emptyset$ y se desea eliminar las redundancias de dicha intersección. Para ello, se comprueba si dicha región tiene intersección no vacía con las regiones adyacentes a X_{12}, obteniéndose $X_{11} \cap X_{21} \neq \emptyset$, $X_{13} \cap X_{21} \neq \emptyset$ y $X_{14} \cap X_{21} = \emptyset$. Por tanto, por aplicación de la proposición 1 se puede garantizar que $\mathcal{F}_2$ es redundante. Por contra, no puede sacarse conclusión alguna respecto a $\mathcal{F}_1$ que puede o no ser redundante en función de la posición de X_{21}, como muestra la figura 1.*

Por lo tanto, para comprobar si una faceta es redundante en una determinada intersección cuando no se cumplen las premisas de

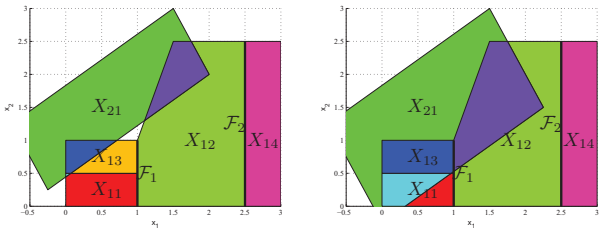

Figura 1: Eliminación de redundancias.

la proposición 1, es necesario resolver un nuevo problema de programación lineal análogo a (13). Las regiones con las que se está trabajando están definidas por:

$$X_{1i} := \{x \in \mathbb{R}^n \,|L_{1i}x \leq W_{1i}\}$$
$$X_{2i} := \{x \in \mathbb{R}^n \,|L_{2i}x \leq W_{2i}\}$$
$$\mathcal{F} := X_{1i} \cap h$$
$$h := \left\{x \in \mathbb{R}^n \,\middle|\, \left[L_h\right]_{[1\times n]} x = \left[W_h\right]_{[1\times n]}\right\}$$

y el problema de optimización a resolver es:

$$\begin{aligned} &\min\ s \\ &\text{sujeto a:} \quad \begin{bmatrix} L_{1i} \\ L_{2i} \end{bmatrix} x - \begin{bmatrix} W_{1i} \\ W_{2i} \end{bmatrix} \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} s \qquad (14) \\ &\qquad\qquad L_h x = W_h \end{aligned}$$

Si se obtiene un óptimo $s < 0$, hay algún punto del hiperplano h que se encuentra en el interior de $X_{1i} \cap X_{2i}$ y por lo tanto no será redundante en dicha intersección. Por el contrario, si $s \geq 0$, el hiperplano será redundante.

Haciendo uso de la proposición 1 es posible establecer un algoritmo que, a partir de dos particiones lineales del espacio de estados, obtenga una única partición del espacio en la que las regiones estén definidas sin redundancias (algoritmo 2).

4.3. *División de la regiones resultantes*

Tras los pasos anteriores, se dispone de una partición del espacio de estados con regiones lineales X_r definidas sin redundancias. Para cada una de estas regiones, existen dos posibles soluciones al problema de optimización, cada una de ellas con un índice de coste asociado:

$$\mathcal{K}_N(x) = \begin{cases} G_{1i}x + h_{1i} \text{ si } V_{1N}^{OPT}(x) < V_{2N}^{OPT}(x) \\ G_{2j}x + h_{2j} \text{ si } V_{2N}^{OPT}(x) < V_{1N}^{OPT}(x) \end{cases}$$

para cada $X_r := \{x \in \mathbb{R}^n \,|L_r x \leq W_r\}$, regiones que se obtienen de intersectar las dos particiones del estado y eliminar las redundancias.

Resulta interesante estudiar en el algoritmo fuera de línea si cada una de las regiones lineales está dividida en subregiones con diferente solución afín, o por el contrario la expresión de dicha solución es la misma para toda la región.

Algoritmo 2 Intersección de dos particiones del espacio de estados.

1. Para cada región de las dos particiones, X_{1i} y X_{2j} obtener información de las regiones adyacentes.
2. Para cada región de la primera partición, X_{1i}, determinar qué regiones de la segunda partición, X_{2j}, tienen una intersección no vacía (resolviendo el problema (13)).
3. Para cada par de regiones (X_{1i}, X_{2j}) que cumpla $(X_{1i} \cap X_{2j} \neq \emptyset)$:
   a) Para cada faceta $\mathcal{F}$ de X_{1i} buscar las regiones adyacentes $\{X_{1k}, X_{1l}, \ldots\}$, comprobar $(X_{1k} \cap X_{2j})$,$(X_{1l} \cap X_{2j})$,etc...:
      - Si todas las restricciones están vacías, $\mathcal{F}$ es redundante en $(X_{1i} \cap X_{2j})$ (proposición 1).
      - Si alguna no está vacía, comprobar la redundancia de $\mathcal{F}$ resolviendo (14).
   b) Para cada faceta $\mathcal{F}$ de X_{2j}, proceder análogamente.

Para ello, basta con obtener la diferencia entre los índices cuadráticos que se obtienen aplicando cada una de las dos posibles soluciones afines:

$$\begin{aligned} Q_{ij}(x) &= V_{1N}^{OPT}(x) - V_{2N}^{OPT}(x) = \\ &= \frac{1}{2}\left((\mathbf{u}_1^{opt})^T H \mathbf{u}_1^{opt} - (\mathbf{u}_2^{opt})^T H \mathbf{u}_2^{opt} + (\mathbf{u}_1^{opt} - \mathbf{u}_2^{opt})^T F x\right) = \\ &= x^T M_{ij} x + N_{ij} x + C_{ij} \qquad (15) \end{aligned}$$

donde

$$\begin{aligned} M_{ij} &= G_{1i}^T H G_{1i} - G_{2i}^T H G_{2i} + G_{1i}^T F - G_{2i}^T F \\ N_{ij} &= 2\left(h_{1i}^T H G_{1i} - h_{2i}^T H G_{2i} + h_{1i}^T F - h_{2i}^T F\right) \\ C_{ij} &= h_{1i}^T H h_{1i} - h_{2i}^T H h_{2i} \end{aligned}$$

En el caso general, $Q_{ij}(x)$ no tiene porqué pasar por el interior de la intersección de las regiones cuyas soluciones afines la definen. Por tanto, en este punto es necesario saber si el sistema de ecuaciones (16) tiene alguna posible solución, es decir, si es consistente.

$$\begin{cases} L_r x \leq W_r \\ x^T M_{ij} x + N_{ij} x + C_{ij} = 0 \end{cases} \qquad (16)$$

En el caso de que sí sea consistente, es evidente que un subconjunto de los puntos que satisfacen $L_r x \leq W_r$ cumple $Q_{ij}(x) > 0$ mientras que para el resto $Q_{ij}(x) < 0$. Para cada uno de ambos subconjuntos, la solución afín que produce un menor índice de coste es diferente, por lo tanto debemos dividir la región lineal en dos.

En el caso contrario, si (16) es inconsistente, para toda la región lineal la solución afín que produce un menor índice de coste es la misma, y no será necesario dividirla.

El sistema (16) es claramente un caso particular de (6), por lo que su factiblidad puede resolverse aplicando la metodología introducida en la sección 2.2. Como se ha visto en dicha sección, la resolución de un problema SDP de la forma (8) nos permitirá únicamente obtener garantías de la infactiblidad del

sistema (16) (para polinomios s_i y t_i de orden acotado se obtienen condiciones suficientes pero no necesarias).
Aunque sería más conveniente obtener garantías necesarias y suficientes, para el problema particular a resolver, una condición suficiente también es de gran utilidad: si al resolver (8) se obtiene solución, la curva cuadrática (15) no cruza la región X_r, y por lo tanto no será necesario dividir ésta, puesto que la expresión afín del óptimo solución del problema será la misma para toda ella. Por otro lado, si no se encuentra solución a (8), se supondrá que la curva pasa por la región lineal aunque, debido al orden acotado de los polinomios escogidos, podría no hacerlo.

Tras resolver el problema (8) es posible decidir qué regiones X_r es necesario dividir porque tiene dos soluciones afines diferentes, y cuales no. Para estas últimas, todavía es necesario averiguar a qué lado de la curva cuadrática se encuentran, y por tanto qué solución afín les corresponde. Como se acaba de comprobar que la región no está dividida por la curva Q_{ij} correspondiente, basta con comprobar a qué lado se encuentra un único punto factible de la región. En particular, se puede utilizar el punto x_f que se obtiene al resolver el problema (13) cuando se está comprobando la factibilidad de la intersección de regiones.
El algoritmo 3 resume el procedimiento para llevar a cabo las divisiones de las regiones lineales mediante curvas cuadráticas.

Algoritmo 3 División de regiones lineales

1. Para cada región X_r obtener la curva cuadrática $Q_{ij}(x)$ (15) sobre la que las dos posibles soluciones afines dan un mismo índice de coste.
2. Resolver el problema de programación SOS (8) siendo la única $f_i(x)$ la curva cuadrática y las $g_i(x)$ las desigualdades $L_r x \leq W_r$
   a) Si el optimizador encuentra solución a (8), (16) es incompatible y por tanto la curva no pasa por la región y no es necesario dividirla. La solución afín correspondiente a X_r será única:
   $$\mathcal{K}_{Nr}(x) = G_r x + h_r \tag{17}$$
   donde $G_r = G_{1i}$ y $h_r = h_{1i}$ si para un punto x_f cualquiera de la región (como el obtenido en (13)) $Q_{ij}(x_f) \leq 0$ y $G_r = G_{2i}$ y $h_r = h_{2i}$ si $Q_{ij}(x_f) > 0$.
   b) Si no la encuentra, la región X_r se divide en dos regiones por la curva (15), cada una de ellas con una solución afín diferente:
   $$\mathcal{K}_{Nr}(x) = \begin{cases} G_{1i}x + h_{1i} \text{ si } Q_{ij}(x) \leq 0 \\ G_{2j}x + h_{2j} \text{ si } Q_{ij}(x) > 0 \end{cases} \tag{18}$$

4.4. Minimización del número de regiones finales

Tras ejecutar el algoritmo anterior, se dispone de una partición del estado en la que las regiones que la forman, X_r, han sido obtenidas mediante la intersección de las regiones correspondientes a dos particiones diferentes: $X_r = X_{1i} \cap X_{2j}$. Cada una de estas intersecciones tendrá una o dos soluciones, dependiendo del resultado de la etapa de división de regiones.

Con el procedimiento seguido, se ha supuesto que en las regiones con una única solución ésta difiere del resto de regiones. No obstante, es posible que a diferentes regiones les corresponda en realidad la misma solución. Esto es así debido a que la solución de una determinada región $X_1 = X_{1i} \cap X_{2j}$, en el caso en que no haya sido necesario dividirla, es o bien la correspondiente a X_{1i} o la correspondiente a X_{2j}. Si se toma otra región $X_2 = X_{1i} \cap X_{2k}$, la solución será la misma si en ambos casos es la correspondiente a X_{1i}, por lo que en principio dichas regiones podrían unirse.
Cabe destacar que es muy importante unir dichas regiones cuando sea posible, puesto que la reducción de regiones de la partición final reduce el coste computacional del algoritmo en línea.
Teniendo esto presente, podemos definir el objetivo de la presente subsección de la siguiente forma:
Dado un conjunto X_r de poliedros con la misma solución afín, obtener un nuevo conjunto X_s que cumpla:

- Que la unión de los nuevos poliedros sea igual a la unión de los originales.
- Que el número de poliedros X_s sea mínimo.

La principal dificultad que aparece al abordar este problema se debe a que la unión de dos o más regiones no tiene por qué ser convexa, aunque las regiones originales sí lo sean.
Un procedimiento diseñado para la resolución de este problema es el propuesto en (Geyer et al., 2008). Para ello, dada una disposición A de hiperplanos $\{H_i\}_{i=1,\ldots,n}$, los autores definen el vector de signos simplificado como:

$$SV_i(x) = \begin{cases} - & \text{si } a_i^T x \leq b_i \\ + & \text{si } a_i^T x \geq b_i \end{cases} \quad \text{para } i = \{1 \ldots n\}$$

A partir de dicha definición, es posible expresar cualquier poliedro como una *celda* de la disposición de planos:

$$P_m = \{x \in \mathbb{R}^d | SV(x) = m\}$$

donde el vector m es la *marca* del poliedro P_m en la disposición de hiperplanos A.
El concepto fundamental detrás de los algoritmos propuestos en (Geyer et al., 2008) es obtener una representación mínima de los poliedros con una misma solución afín, partiendo de su envoltura, que es convexa y sencilla de calcular (Bemporad et al., 2001), y dividiéndola secuencialmente en poliedros usando los hiperplanos de A mediante uno de dos algoritmos, uno basado en ramificación y poda y el otro en lógica booleana.
El punto más costoso del algoritmo de minimización de regiones de (Geyer et al., 2008) es la obtención de la disposición de hiperplanos y las marcas de los diferentes poliedros. Tanto si se usan los algoritmos propuestos por los autores como otros basados en búsqueda inversa, (Avis and Fukuda, 1996) y (Ferrez et al., 2001), es necesaria la resolución de un determinado número de problemas LP.

A continuación se describe un procedimiento que, a partir de la información disponible del problema obtenida en las secciones anteriores, calcula la disposición de hiperplanos y las marcas de los poliedros sin necesidad de resolver ningún nuevo problema de optimización. Para ello, se parte de la siguiente observación:

Observación 1. *Dadas dos particiones del espacio de estado en las que las soluciones afines correspondientes a cada una de las regiones X_{1i} y X_{2j} sean diferentes, las únicas regiones de la partición X_r, obtenida como intersección de las dos anteriores, cuya solución afín puede ser la misma son aquellas formadas mediante la intersección de una misma región de una de las particiones de origen.*

Para diseñar el procedimiento para la unión de regiones, teniendo presente la observación 1, se partirá de una región de una de las particiones de origen, X_{1i}, y se buscarán todas las regiones X_r formadas como intersección de dicha región con alguna de las regiones de la otra partición $\{X_r | X_r = X_{1i} \cap X_{2j}, j = 1, \dots N_{r2}\}$.
Evidentemente, algunas de las regiones tendrán la misma solución afín (la correspondiente a X_{1i}), pero otras no. Además, la unión de todas las regiones X_r seleccionadas de esta manera forman un poliedro convexo, la región X_{1i}. Por tanto, es posible aplicar los algoritmos de unión de regiones descritos.
Para la obtención de la disposición de hiperplanos se seguirá el procedimiento de (Geyer et al., 2008) utilizando la información disponible de regiones adyacentes. En cuanto a la obtención de las marcas, pueden utilizarse los puntos factibles de cada región, x_{fr}, obtenidos en (13):

$$m_r = SV(x_{fr}) \tag{19}$$

El algoritmo 4 describe el procedimiento que une todas las regiones posibles, minimizando el número de regiones finales obtenidas.

La metodología descrita no es válida para unir regiones con la misma solución afín pero que no provengan de la intersección de una misma región de las particiones de origen. No obstante, ésta no es una limitación importante puesto que estos casos podrán en general evitarse con una adecuada selección de las regiones convexas de partida y, en caso contrario, puede generalizarse el algoritmo descrito de una manera relativamente sencilla.

Algoritmo 4 Minimización del número de regiones lineales

1. Para cada región X_{1i} buscar las regiones $\{X_r | X_r = X_{1i} \cap X_{2j}, j = 1, \dots N_{r2}\}$.
2. Comprobar la solución afín de cada X_r.
   - Si todas tienen la misma solución, quedarse únicamente con X_{1i}.
   - Si ninguna tiene la misma solución, pasar a la siguiente región X_{1i}.
   - Si sólo algunas tienen la misma solución, continuar con el algoritmo.
3. Obtener la disposición de planos A, eliminando desigualdades innecesarias.
4. Obtener marcas de todas las X_r según (19).
5. Aplicar algoritmo de unión de regiones:
   - Si no se desea solapamiento, aplicar algoritmo de ramificación y poda de (Geyer et al., 2008).
   - Si se permite solapamiento, aplicar algoritmo de minimización de lógica booleana de (Geyer et al., 2008).
6. Repetir procedimiento para cada región X_{2j}.

5. Solución explícita para $\gamma > 2$

En la sección anterior se ha propuesto un algoritmo para la obtención de la solución explícita de un controlador predictivo con restricciones definidas como la unión no convexa de dos poliedros. A continuación, se propondrá un procedimiento que, basado en el anterior, obtenga la solución explícita cuando las restricciones del problema estén definidas como la unión de un mayor número de poliedros.
Para ello, inicialmente se obtienen todas las soluciones explícitas con las restricciones definidas como los γ diferentes poliedros. A partir de dos de esas particiones, se aplica el algoritmo 1. Como resultado de éste se obtiene una nueva partición lineal del estado ($\mathbb{T}_2$) para la que algunas regiones (que se denotarán como $\mathbb{T}_2^1$) tienen una única solución afín, mientras que para otras (las de $\mathbb{T}_2^2$) existen dos posibles soluciones, dependiendo de a qué lado de la correspondiente curva se encuentre el vector de estados x. Sobre esta partición $\mathbb{T}_2$, se añade una nueva partición lineal, intersectando sus regiones. En este caso se obtendrán regiones con 1, 2 o hasta 3 posibles soluciones divididas por las correspondientes curvas. Añadiendo de manera sucesiva las γ particiones, se llega a la solución explícita final. El algoritmo 5 resume este procedimiento.

Puede apreciarse como, esencialmente, dicho algoritmo obtiene las γ particiones del espacio de estados y las va intersectando una a una. Cada vez que se añade una partición, por tanto, se realizan una serie de pasos que son muy similares a los que se llevan a cabo en el algoritmo presentado para dos regiones no convexas: intersectar las regiones de las dos particiones, dividirlas con curvas cuadráticas si es necesario y unir, cuando sea posible, las regiones que tienen una misma solución. No obstante, algunos de estos pasos presentan algunas particularidades que se describen a continuación.

5.1. Obtención de la solución explícita de los subproblemas

Al igual que en el caso de $\gamma = 2$, se obtiene la solución explícita a cada uno de los subproblemas de control predictivo con restricciones lineales y se amplía añadiendo las regiones que sea necesario para cubrir el espacio de estados iniciales factibles total, tal y como se ha descrito en la sección 4.1.

5.2. Intersección de las particiones de estado

En cada paso del algoritmo tendremos dos particiones lineales del espacio de estados: $\mathbb{T}_{i-1}$, cuyas regiones pueden estar

Algoritmo 5 Obtención de la solución explícita $\gamma > 2$

1. Obtener la solución explícita de los γ problemas de optimización.
2. Tomar la partición con menor número de regiones e inicializar:
   - $\mathbb{T}_1^1 = \{X_{1i}\}$, $\mathbb{T}_1^2 = \ldots = \mathbb{T}_1^\gamma = \{\emptyset\}$.
   - $\mathbb{T}_1 = \mathbb{T}_1^1$.
   - $i = 2$.
3. Tomar la siguiente partición con menor número de regiones, $\mathbb{X}_i$.
4. Intersectar con las regiones de cada una de las particiones anteriores: $\mathbb{W}_i^j = \{X_{jk} \cap X_{il} | X_{jk} \in \mathbb{T}_{i-1}^j, X_{il} \in \mathbb{X}_i\}$.
5. Para cada región de $\mathbb{W}_i^j$,comprobar cuáles de las $j+1$ posibles soluciones son aplicables a la región.
6. Actualizar las listas de regiones, $\mathbb{T}_i^j$ en función del número de soluciones aplicables.
7. Si es posible, unir las regiones de $\mathbb{T}_i^1$.
8. $i = i + 1$.
9. Si $i \leq \gamma$, volver a 3.

divididas por curvas cuadráticas, proveniente de pasos anteriores del algoritmo, y $\mathbb{X}_i$, obtenida como solución a un problema con restricciones lineales y añadida en el paso actual.

Aunque para el desarrollo del algoritmo 5 se haya dividido la partición $\mathbb{T}_{i-1}$ en diferentes subconjuntos formados por las regiones con diferente número de posibles soluciones afines, a efectos prácticos en este paso hay que intersectar todas las regiones de dos particiones lineales del espacio de estados. Por tanto, la factibilidad de estas intersecciones puede determinarse resolviendo un problema LP análogo a (13), de igual forma que para el caso de $\gamma = 2$.

En cuanto a la eliminación de redundancias en la definición de las regiones, la proposición 1 es igualmente válida. No obstante, el procedimiento en este caso sí difiere en el modo de obtener qué regiones son adyacentes. Para la partición $\mathbb{X}_i$, sigue siendo habitual tener dicha información tras obtener la solución explícita del problema convexo. Sin embargo, para la partición $\mathbb{T}_{i-1}$ esto no es posible, puesto que las regiones ya no se corresponden con regiones críticas asociadas a diferentes conjuntos de restricciones activas de un mismo problema. Por tanto, es necesario diseñar otro método. Para ello, se formulan las siguientes proposiciones:

Proposición 2. *Dadas dos regiones diferentes de una misma partición lineal* $\mathbb{X}_i$, $X_1 := \{x \in \mathbb{R}^n | L_1 x \leq W_1\}$ *y* $X_2 := \{x \in \mathbb{R}^n | L_2 x \leq W_2\}$*:*

- $dim(X_1 \cap X_2) = n - 1$ *si* X_1 *y* X_2 *son adyacentes.*
- $dim(X_1 \cap X_2) < n - 1$ *si* X_1 *y* X_2 *no son adyacentes.*

Prueba 2. *Al pertenecer ambas regiones a una misma partición, se tiene* $int(X_i) \cap int(X_j) = \emptyset$ *y por tanto* $dim(X_1 \cap X_2) < n$*. Teniendo en cuenta que, por definición, dos regiones son adyacentes si* $dim(X_1 \cap X_2) = n-1$*, la proposición queda demostrada.* ■

Proposición 3. *Dos regiones de una partición* $\mathbb{T}_i = \mathbb{T}_{i-1} \cap \mathbb{X}_i$*:*

$$T_1^i = T_1^{i-1} \cap X_1$$
$$T_2^i = T_2^{i-1} \cap X_2$$

serán adyacentes si y sólo si se cumple una de estas tres condiciones:

- $T_1^{i-1} = T_2^{i-1}$ *y* X_1 *es adyacente a* X_2 *con una intersección* $S_1 = X_1 \cap X_2$ *que cumple* $dim(S_1 \cap T_1^{i-1}) = n - 1$.
- $X_1 = X_2$ *y* T_1^{i-1} *es adyacente a* T_2^{i-1} *con una intersección* $S_2 = T_1^{i-1} \cap T_2^{i-1}$ *que cumple* $dim(S_2 \cap X_1) = n - 1$.
- T_1^{i-1} *es adyacente a* T_2^{i-1} *a través de un hiperplano h,* X_1 *es adyacente a* X_2 *por el mismo hiperplano y* $dim(S_1 \cap S_2) = n - 1$.

Prueba 3. *Si* T_1^i *y* T_2^i *son adyacentes, por definición su intersección tiene dimensión* $n - 1$*. Podemos reescribir dicha intersección como:*

$$T_1^i \cap T_2^i = (T_1^{i-1} \cap X_1) \cap (T_2^{i-1} \cap X_2) =$$
$$= (X_1 \cap X_2) \cap (T_1^{i-1} \cap T_2^{i-1}) = S_1 \cap S_2$$

Demostraremos en primer lugar la necesidad de las condiciones expuestas. Para que $T_1^i \cap T_2^i$ *sea de dimensión* $n - 1$*,* S_1 *y* S_2 *tienen que ser, como mínimo, de dimensión* $n - 1$*. Se pueden distinguir tres casos diferentes:*

- $dim(S_1) = n - 1$ *y* $dim(S_2) = n$*. Por la proposición 2,* T_1^{i-1} *y* T_2^{i-1} *tienen que ser la misma región y* X_1 *y* X_2 *adyacentes. Para que* $T_1^i \cap T_2^i$ *no esté vacía, se debe cumplir también* $S_1 \cap X_2 \neq \emptyset$*.*
- $dim(S_1) = n$ *y* $dim(S_2) = n-1$*. Por la proposición 2,* T_1^{i-1} *y* T_2^{i-1} *tienen que ser adyacentes y* $X_1 = X_2 = S_1$ *la misma región. En este caso, para que* $T_1^i \cap T_2^i$ *no esté vacía, se debe cumplir* $S_2 \cap X_1 \neq \emptyset$*.*
- $dim(S_1) = dim(S_2) = n - 1$*. Como se ha visto en la proposición 2, esto implica que* T_1^{i-1} *y* T_1^{i-1} *son adyacentes y que* X_1 *y* X_2 *también lo son. Además, la intersección de las dos intersecciones* $S_1 \cap S_2$ *sólo será de dimensión* $n - 1$ *si corresponde al mismo hiperplano.*
- $dim(S_1) = n$ *y* $dim(S_2) = n$*. En este caso, por la proposición 2* $T_1^{i-1} = T_2^{i-1}$ *y* $X_1 = X_2$*, por lo que* $T_1^i = T_2^i$ *y se trata de un caso trivial.*

En cuanto a la suficiencia, hay que probar que si se cumplen las condiciones de cada uno de los tres supuestos, T_1^i *y* T_2^i *serán adyacentes, o lo que es lo mismo que dim* $(T_1^i \cap T_2^i) = n - 1$.

- *Si* $T_1^{i-1} = T_2^{i-1}$*,* X_1 *y* X_2 *son adyacentes y* $dim(S_1 \cap T_1^{i-1}) = n - 1$*:*

  $$T_1^i \cap T_2^i = S_1 \cap S_2 = S_1 \cap T_1^{i-1}$$

  Por tanto, $dim(T_1^i \cap T_2^i) = n - 1$

- *Si* $X_1 = X_2$, T_1^{i-1} *y* T_2^{i-1} *son adyacentes y* $dim(S_2 \cap X_1) = n-1$*:*

$$T_1^i \cap T_2^i = S_1 \cap S_2 = S_2 \cap X_1$$

Por tanto, $dim(T_1^i \cap T_2^i) = n-1$

- *Si* $dim(S_1 \cap S_2) = n-1$*, es trivial que las regiones son adyacentes.*

■

Ejemplo 2. *La figura 2 muestra las regiones de dos particiones lineales,* $\mathbb{X}$ *y* $\mathbb{T}^1$*. Si se intersectan ambas particiones, se obtiene la partición* $\mathbb{T}^2$ *(figura 3). En ésta se pueden observar regiones adyacentes de los tres tipos diferentes contemplados en la proposición 3:*

- $T_1^2 = X_1 \cap T_1^1$ *es adyacente a* $T_2^2 = X_1 \cap T_2^1$*.*
- T_1^2 *es adyacente a* $T_3^2 = X_2 \cap T_1^1$*.*
- T_2^2 *es adyacente a* $T_5^2 = X_3 \cap T_3^1$ *puesto que* X_1 *es adyacente a* X_3 *a través del mismo hiperplano que* T_2^1 *es adyacente a* T_3^1*.*

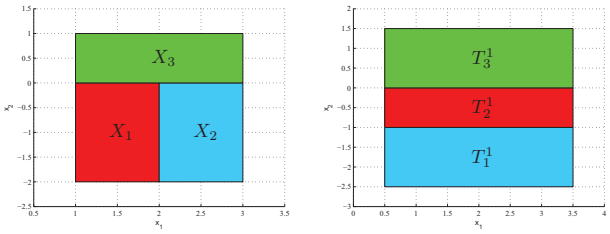

Figura 2: Regiones de las particiones $\mathbb{X}$ y $\mathbb{T}^1$.

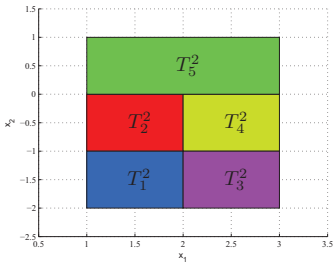

Figura 3: Regiones de la partición $\mathbb{T}^2 = \mathbb{X} \cap \mathbb{T}^1$.

Es importante notar que, para los dos primeros supuestos de la proposición 3, la comprobación de si dos regiones T_1^i y T_2^i son adyacentes se puede realizar de manera sencilla en la iteración anterior, cuando se han eliminado las redundancias de dichas regiones. Supongamos que se tienen dos particiones, $\mathbb{X}_1$ y $\mathbb{X}_2$ y que se parte de las regiones X_{11} y X_{12} que son adyacentes, y de una región X_{21} que, tras resolver el problema (13), se comprueba que tiene intersección no nula con las anteriores. Si se sigue el procedimiento que se ha descrito en el algoritmo 2, para eliminar las posibles redundancias de $X_{11} \cap X_{21}$ y de $X_{12} \cap X_{21}$ es necesario comprobar el hiperplano h correspondiente a la faceta común a ambas regiones. Es evidente que el supuesto de la proposición, $\dim(X_{11} \cap X_{12} \cap X_{21}) = n-1$ es cierto si y sólo si h no es redundante en las dos regiones.

Por tanto, cuando se comprueba la redundancia de un hiperplano en dos regiones $(X_1 \cap T_1^{i-1})$ y $(X_2 \cap T_1^{i-1})$, o bien se elimina de al menos una de ellas (en cuyo caso no es posible que sean adyacentes a través del hiperplano en cuestión), o bien las dos regiones son adyacentes.

En cuanto al tercer supuesto de la proposición, representa un tipo de regiones adyacentes que sólo se producirá en casos muy particulares, cuando haya hiperplanos coincidentes en la descripción de las diferentes zonas de restricciones. Si esto se detecta, puede formularse un LP con las desigualdades de todas las regiones que intervienen para obtener $S_1 \cap S_2$ y comprobar su dimensión.

Una vez comprobado qué regiones son adyacentes, es posible aplicar el algoritmo 2 a partir del paso 2, lo que permite obtener todas las regiones lineales sin redundancias.

Una vez se ha realizado la intersección de regiones, es conveniente aplicar de nuevo el algoritmo de unión 4 para las regiones con la misma solución, ya que si alguna se puede unir se reduce el número de problemas de suma de cuadrados a realizar.

5.3. *Eliminación de las soluciones no óptimas*

Una vez realizada la intersección de las regiones lineales y eliminadas sus redundancias, nos encontramos en el punto 5 del algoritmo 5, con una serie de conjuntos de regiones $\mathbb{W}_i^j = \{X_{jk} \cap X_{il} | X_{jk} \in \mathbb{T}_{i-1}^j, X_{il} \in \mathbb{X}_i\}$. Las regiones X_{jk}, son aquellas que en la partición anterior tenían j posibles soluciones por lo que, al intersectarlas con una región de la nueva partición, X_{il}, a priori tienen $j+1$ posibles soluciones.

Al igual que se vio en la sección 4.3, es importante, desde el punto de vista del coste computacional del algoritmo en línea, decidir cuándo la región de intersección tiene efectivamente $j+1$ posibles soluciones o cuándo son menos, porque algunas de ellas quedan descartadas al ser peor que las demás para todos los puntos de la región lineal.

Dada una determinada región lineal de $\mathbb{W}_i^j$:

$$X_r := \{x \in \mathbb{R}^n | L_r x \leq W_r\}$$

para decidir cuando una de las soluciones afines posibles, k, es una de las factibles en X_r es necesario comprobar si, en algún subconjunto de dicha región, el índice de coste asociado a la solución afín, V_k^{OPT}, es menor que todos los índices de coste asociados al resto de soluciones. Para ello, se ha de resolver un problema de consistencia de la siguiente forma:

$$\begin{cases} L_r x \leq W_r \\ V_k^{OPT} \leq V_l^{OPT} \quad l = \{1 \ldots (j+1)\} \backslash \{k\} \end{cases} \tag{20}$$

Si el problema anterior es consistente, existe algún subconjunto de X_r para el que el menor índice de coste es el asociado a la solución k, y por tanto dicha solución es una de las posibles.

El problema (20) es de nuevo un caso particular de (6), y por tanto puede resolverse como un problema de suma de cuadrados. En este caso no existen restricciones de igualdad $(f_i(x))$ y las restricciones de desigualdad $g_i(x)$ son de dos tipos:

- Desigualdades lineales que definen X_r (tantas como filas tienen las matrices L_r y W_r).
- j curvas cuadráticas de la forma $V_l^{OPT}(x) - V_k^{OPT}(x) \geq 0$.

Una vez resuelto un problema SOS de la forma (20), se sabe si una determinada solución afín, k, es una de las factibles en la región. Para comprobar todas las soluciones será por tanto necesario resolver $j + 1$ problemas.

Es interesante tratar de forma ligeramente diferente el caso particular de $j = 1$, puesto que las regiones de $\mathbb{W}_i^1$ se forman intersectando las regiones pertenecientes a $\mathbb{T}_{i-1}^1$ con las pertenecientes a $\mathbb{X}_i$. Por definición, las regiones pertenecientes a dichas particiones tienen una única solución afín en cada una de ellas. Por tanto, al igual que en la sección 4.3, puede resolverse mediante un único SOS con restricción de igualdad (16) y, en el caso de que el problema no sea consistente, comprobar después cuál de las dos soluciones es la óptima en la región.

Otra consideración a tener en cuenta que, si bien no reduce el número de SOS a resolver, sí hace que algunos de estos puedan ser más sencillos, es ir progresivamente eliminando de las comprobaciones aquellas soluciones que no son factibles. Es decir, si se resuelve un primer problema (21) y se obtiene que el problema no es factible, se tiene la certeza de que la primera solución nunca va a ser la mejor en la región X_r. Por tanto, en el siguiente SOS a resolver puede eliminarse una de las restricciones cuadráticas (22), lo que en general hará que sea menos costoso.

$$\begin{cases} L_r x \leq W_r \\ V_1^{OPT} \leq V_l^{OPT} \quad l = \{2 \ldots (j+1)\} \end{cases} \tag{21}$$

$$\begin{cases} L_r x \leq W_r \\ V_2^{OPT} \leq V_l^{OPT} \quad l = \{3 \ldots (j+1)\} \end{cases} \tag{22}$$

El algoritmo 6 resume el procedimiento para obtener qué soluciones son factibles para $j > 1$ (para $j = 1$, como se ha visto, es aplicable el algoritmo 3).

Con este procedimiento se ha resuelto el punto 5 del algoritmo 5. El punto 6 es directo, puesto que consiste simplemente en reasignar las regiones en $\mathbb{W}_i^j$ en los diferentes $\mathbb{T}_i^j$ en función de cuantas de las soluciones han resultado ser factibles.

5.4. Minimización del número de regiones

Por último, el paso 7 del algoritmo 5 consiste en unir aquellas regiones adyacentes con una misma solución afín (que por tanto debe ser única en cada una de ellas). Para ello, se aplica un algoritmo igual al 4 para las regiones correspondientes.

6. Justificación del algoritmo

Como se vio en la sección 3, el problema de control predictivo (10) cuya resolución se plantea, puede reescribirse de la forma (12) y por tanto resolverse mediante la comparación de los mínimos que se consiguen en cada uno de los γ problemas convexos, que pueden resolverse de manera explícita como se vio en la sección 2.

Algoritmo 6 Obtención de regiones factibles en X_r para $j > 1$.

1. Para cada región $X_r = T_{i-1} \cap X_i$ obtener el conjunto de $j + 1$ soluciones posibles como la unión de las soluciones factibles de las regiones que lo forman: $SP(X_r) = SF(T_{i-1}) \cup SF(X_i)$
2. Inicializar el conjunto de soluciones no factibles: $SNF(X_r) = \emptyset$.
3. Tomar k como la primera solución de $SP(X_r)$
4. Resolver el problema SOS:

$$\begin{cases} L_r x \leq W_r \\ V_k^{OPT} \leq V_l^{OPT} \quad l = \{1 \ldots (j+1)\} \backslash \{SNF(X_r) \cup k\} \end{cases}$$

   - Si el problema es factible, $SF(X_r) = SF(X_r) \cup k$.
   - Si el problema no es factible, $SNF(X_r) = SNF(X_r) \cup k$.

5. Si quedan soluciones sin explorar de $SP(X_r)$, tomar k como la siguiente y volver a 4.

Resulta sencillo de ver que, desde el punto de vista del tiempo de cómputo fuera de línea, esta alternativa es mucho más simple, puesto que sólo requiere la resolución explícita de los problemas convexos, sin necesidad de realizar la intersección de particiones ni las optimizaciones de suma de cuadrados usadas en el algoritmo propuesto. No obstante, si el objetivo fundamental es conseguir un algoritmo en línea con un coste mínimo, el algoritmo propuesto presenta ventajas respecto a la resolución y comparación de los problemas convexos.

Una solución explícita (5) puede calcularse en línea de forma muy eficiente mediante la construcción de un árbol de búsqueda binario. En este contexto, los árboles binarios fueron introducidos en (Tøndel et al., 2003b) como una alternativa mucho más eficiente que la búsqueda secuencial para la evaluación de funciones afines a tramos definidas sobre particiones poliédricas. Los árboles binarios son estructuras de datos formadas por varios nodos, cada uno de ellos con una desigualdad asociada $d(x)$ y dos nodos hijos, en el caso de los nodos no-hoja, o una expresión afín, en el caso de los nodos hoja. El árbol se recorre partiendo del nodo raíz y decidiendo a qué nodo ir a continuación comprobando el signo de la desigualdad correspondiente. El proceso se repite hasta que se alcanza un nodo hoja, y por tanto una función afín. Se define la profundidad del árbol binario D como el número de desigualdades que es necesario comprobar en el peor caso para llegar a un nodo hoja. En (Tøndel et al., 2003b) se muestra como la profundidad de un árbol es una función logarítmica del número de regiones que define la partición, N_r:

$$D = \left\lceil \frac{lnN_r}{ln\alpha} \right\rceil \tag{23}$$

donde α es un parámetro que representa el equilibrado del árbol.

Por tanto, dado un valor del vector de estados, el coste de obtener la acción de control para una partición lineal para la que se dispone de un árbol binario es comprobar el signo de un

determinado número de hiperplanos (en el peor caso la profundidad del árbol, D_i) y calcular la ley de control.

Para el problema no convexo, el coste en línea del algoritmo consistente en resolver los subproblemas convexos correspondería a comprobar $\sum^{\gamma} D_i$ hiperplanos, calcular γ índices de coste cuadráticos y buscar cuál de ellos es mínimo, y calcular la ley de control correspondiente.

Por contra, el algoritmo propuesto requiere recorrer un árbol de profundidad D, buscar el mínimo índice de coste entre un máximo de γ funciones cuadráticas y calcular la ley de control correspondiente. En este punto, es interesante notar que el número de regiones de la partición final, N, es como máximo el producto del número de regiones de cada una de las particiones de los problemas convexos, N_i, pero, para la mayoría de casos, muchas de las intersecciones son vacías y además es posible unir regiones con la misma solución, por lo que $N << \prod N_i$. Por tanto, al ser la profundidad de los árboles una función logarítmica, se tiene $D << \sum D_i$, lo que hace que el algoritmo en línea para la solución propuesta sea en general mucho menos costoso.

Otro aspecto importante para analizar los algoritmos en línea es la cantidad de memoria necesaria para su implementación. De hecho, éste es un aspecto crucial para muchas aplicaciones, puesto que el uso de memoria es en muchos casos el cuello de botella que impide el uso de un control predictivo explícito en lugar de la optimización en línea (Johansen et al., 2007).

Para el caso del algoritmo de los subproblemas, es necesario almacenar la estructura de los γ árboles binarios (todos los hiperplanos que definen las diferentes regiones y dos punteros para cada uno de los nodos del árbol), además de todas las leyes de control posibles y los índices de coste asociados a cada una de ellas. Sin embargo, para el algoritmo propuesto, se consigue una reducción de memoria considerable debido a tres motivos diferentes. En primer lugar, el número de hiperplanos a almacenar es, en el peor caso, el mismo que se necesita para almacenar las particiones solución de los problemas convexos, puesto que las regiones de la partición final están formadas por intersección de las regiones de éstas últimas. No obstante, el número de hiperplanos puede ser menor debido a los procedimientos de eliminación de redundancias y unión de regiones con la misma solución. En segundo lugar, puede que no sea necesario almacenar algunas de las leyes de control obtenidas para los problemas convexos, puesto que éstas no sean óptimas para ningún punto del espacio de estados. Por último, la reducción de memoria más significativa se produce por el hecho de que ya no es necesario almacenar todos los índices de coste, sino únicamente aquellos que corresponden a regiones con más de una solución posible.

7. Problema subóptimo simplificado

En las secciones anteriores, se ha planteado una metodología para obtener la solución explícita al problema de optimización (10). Aunque dicha metodología es adecuada cuando se trabaja con un número γ de poliedros T_i no demasiado elevado, por la propia naturaleza no convexa del problema de optimización resulta complicada de implementar a medida que γ aumenta. Esto es así fundamentalmente por dos motivos: γ es el número de particiones del espacio de estados que es necesario intersectar y, además, el número de problemas de suma de cuadrados que es necesario resolver en el peor de los casos para cada región lineal. Por tanto, resulta interesante plantear alguna alternativa para el caso en que un γ elevado no permita la obtención de la solución explícita buscada.

Si se define γ_f como el número de poliedros convexos cuya unión define $\bar{\mathbb{X}}_f$, si todos los T_i son factibles se tiene

$$\gamma = (\gamma_u)^M (\gamma_x)^{N-1} \gamma_f$$

Por tanto, existen fundamentalmente dos posibilidades directas para disminuir γ: reducir el número de regiones convexas admisibles para los conjuntos de restricciones $\bar{\mathbb{X}}$, $\bar{\mathbb{U}}$ o $\bar{\mathbb{X}}_f$ y reducir los horizontes de control M o predicción N. La primera opción implica reducir el espacio de restricciones del sistema, mientras que la segunda puede impedir la elección del ajuste del controlador que se considere óptima. Por tanto, ambas posibilidades pueden suponer una disminución no aceptable en la calidad de la respuesta o en la región de atracción que se pueden conseguir con el sistema en bucle cerrado.

Si las posibilidades anteriores no son viables, otra opción surge mediante una combinación de ambas, teniendo en cuenta la filosofía del horizonte móvil del control predictivo. La idea es, considerando que sólo se aplican las m primeras acciones de control, aplicar las restricciones exactas a dichas acciones de control pero sustituirlas por un nuevo espacio de restricciones convexo para las acciones de control y/o los estados siguientes. De una manera más general, se aplican las restricciones de manera exacta a $L < M \leq N$ acciones de control, y un nuevo conjunto de restricciones convexo al resto de acciones de control (24).

$$\begin{aligned}
\mathcal{P}^{so}_{N,M} \quad & : V^{so}_{N,M}(x) := \min \tfrac{1}{2}\mathbf{u}^T H \mathbf{u} + \mathbf{u}^T F x, \\
& \text{sujeto a:} \\
& \quad u_k \in \bar{\mathbb{U}} \mid k = 0, \ldots, L, \\
& \quad u_k \in \mathbb{U}_0 \mid k = L+1, \ldots, M-1, \\
& \quad u_k = 0 \text{ para } k = M, \ldots, N-1, \\
& \quad f_k(x, u_0, \ldots, u_{k-1}) \in \bar{\mathbb{X}} \mid k = 1, \ldots, L, \\
& \quad f_k(x, u_0, \ldots, u_{k-1}) \in \mathbb{X}_0 \mid k = L+1, \ldots, N-1, \\
& \quad f_N(x, \mathbf{u}) \in \mathbb{X}_{f0}
\end{aligned} \tag{24}$$

donde $\mathbb{U}_0$, $\mathbb{X}_0$ y $\mathbb{X}_{f0}$ son poliedros convexos que contienen el origen de sus respectivos espacios y están, a su vez, contenidos en $\bar{\mathbb{U}}$, $\bar{\mathbb{X}}$ y $\bar{\mathbb{X}}_f$ respectivamente.

Con esta simplificación, si se agrupan las acciones de control futuras en un vector como se hizo con el problema original, se obtendrá un menor número γ de conjuntos T_i, en concreto $(\gamma_u \cdot \gamma_x)^L$.

Por otra parte, el comportamiento del sistema en bucle cerrado no tiene por qué verse afectado gravemente, puesto que es de esperar que los últimos elementos de la secuencia de control óptima y la trayectoria predicha asociada a ésta se encuentren en un entorno cercano a los respectivos orígenes de ambos espacios.

8. Ejemplo

Se desea obtener un controlador explícito para el sistema de dos entradas y una salida con período de muestreo $T_s = 1s$ descrito por su representación en espacio de estados:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases}$$

$$A = \begin{bmatrix} 0{,}6065 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 0{,}7869 & 1 \end{bmatrix}$$

y sujeto a unas restricciones sobre los estados $\{-15 \leq x_1 \leq 15; -10 \leq x_2 \leq 10\}$ y sobre las acciones de control $\bar{\mathbb{U}} = \mathbb{U}_1 \cup \mathbb{U}_2$ donde:

$$\mathbb{U}_1 := \left\{ u \in \mathbb{R}^2 \left| \begin{array}{l} -1 \leq u_1 \leq 3 \\ -1 \leq u_2 \leq 1 \end{array} \right. \right\} \quad \mathbb{U}_2 := \left\{ u \in \mathbb{R}^2 \left| \begin{array}{l} 1 \leq u_1 \leq 3 \\ 1 \leq u_2 \leq 3 \end{array} \right. \right\}$$

El espacio de restricciones así definido es claramente no convexo, como se muestra en la figura 4.

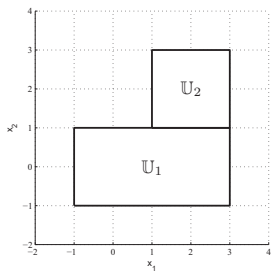

Figura 4: Restricciones sobre acciones de control.

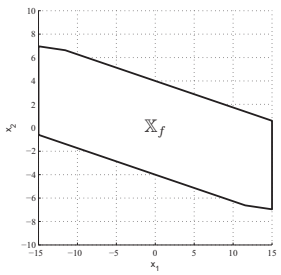

Figura 5: Restricción terminal.

Además, con objeto de garantizar la estabilidad del sistema en bucle cerrado, se añade una restricción terminal sobre el estado $x_N \in \mathbb{X}_f$ como la que se muestra en la figura 5.

Los parámetros de diseño del controlador se escogen de la siguiente manera:

$$M = N = 5, \quad Q = 0{,}1, \quad R = I$$

La matriz de ponderación del estado final, P, se obtiene resolviendo la correspondiente ecuación algebraica de Ricatti.
Inicialmente, es necesario escribir el problema de optimización a resolver de la forma (12).

Para ello, se forman las regiones de restricciones T_i. Dado que el horizonte elegido es $M = N = 5$ y que sólo existen restricciones no convexas sobre las acciones de control, se tiene $\gamma = 2^5 = 32$).
De esta forma, se tienen 32 subproblemas de optimización con restricciones lineales, sobre los que se resuelven los respectivos problemas *mpQP* obteniendo particiones del espacio de estados definidas con un número variable de regiones entre 13 y 42.

Aplicando el algoritmo propuesto, se obtiene la partición que se muestra en la figura 6. Dicha partición está formada por 210 regiones, 128 de ellas con una única solución, 70 con dos soluciones y 12 con tres.

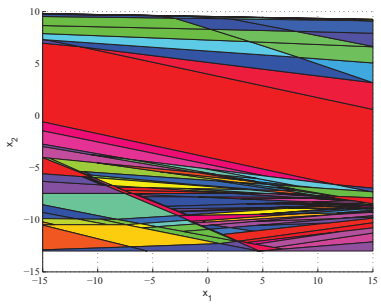

Figura 6: Solución final $N = 5$. Partición del estado.

Para la obtención de esta solución, ha sido necesaria la resolución de 6668 problemas de suma de cuadrados, y el procedimiento en su totalidad ha empleado 1686 segundos en un Intel core i5 750 a 2.66 GHz.

Es interesante analizar la solución obtenida en términos de la simplicidad del algoritmo en línea para su implementación. Para ello, según (23) y asumiendo un valor conservativo de $\alpha = 0{,}66$ (Tøndel et al., 2003b), se tiene una profundidad esperada del árbol binario de: $D = 13$. Por tanto, en el peor de los casos en algoritmo en línea deberá calcular el signo de 13 hiperplanos, calcular y buscar el mínimo entre 3 funciones cuadráticas y aplicar una ley de control.

Por contra, si no se obtiene la solución explícita final, sería necesario obtener 32 árboles binarios para los que la suma de las profundidades esperadas es de $\sum D_i = 260$. Tras la comprobación de dicho número de hiperplanos, sería necesario calcular y buscar el mínimo entre 32 funciones cuadráticas y, por último, calcular y aplicar una ley de control.

En términos de memoria, como se ha analizado, el algoritmo propuesto requiere el almacenamiento de, como máximo, el mismo número de hiperplanos que el algoritmo basado en los subproblemas convexos. El número de leyes de control se reduce de las 809 que requiere el algoritmo de los subproblemas (una para cada una de las regiones que en total definen las 32 particiones) a las 304 del algoritmo propuesto (128 para las regiones con una única solución, 140 para las 70 regiones con dos soluciones y 36 para las de tres). Por último, el número de funciones cuadráticas es significativamente inferior: de las 809 funciones del algoritmo de los subproblemas a únicamente 176 para el algoritmo propuesto (para las regiones con más de una solución).

Si la solución obtenida aún se considera demasiado compleja para su implementación, una alternativa es aplicar la simplificación propuesta en la sección 7. En concreto, se toma una $L = 2$ y unas restricciones simplificadas sobre la acción de control $\mathbb{U}_0 = \mathbb{U}_1$.

En este caso, se tiene un problema definido por 4 conjuntos T_i. Tras la aplicación de la metodología propuesta, se obtiene una partición definida por 86 regiones, 67 con una única solución, 12 con dos y 7 con tres. Para la obtención de esta solución, se han resuelto 425 problemas de suma de cuadrados y se han empleado un total de 126 segundos.

	N=5	N=2	N=5,L=2
$x_0 = [-15\ 7{,}6]^T$	8.98	9.60	8.98
$x_0 = [15\ 5{,}1]^T$	51.47	51.47	51.47
$x_0 = [15\ -12{,}3]^T$	23.58	29.89	23.66
$x_0 = [-15\ -9{,}8]^T$	93.28	94.71	93.28

Tabla 1: Índice de coste J.

Para este problema, la región de factibilidad coincide con la obtenida para el caso de $N = 5$.

A efectos de comparación, es conveniente definir un tercer controlador con $N = 2$. Para este caso, el problema tiene una complejidad similar a la del problema con $N = 5$ y $L = 2$, puesto que tiene el mismo número de conjuntos T_i, 4. La solución explícita para este controlador es la mostrada en la figura 7, y está definida por 47 regiones (31 con una solución, 11 con dos y 4 con tres) obtenidas mediante 196 problemas de suma de cuadrados con un tiempo total de 52 segundos.

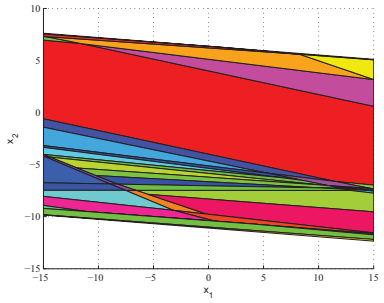

Figura 7: Solución final $N = 2$. Partición del estado.

Puede comprobarse como con este último controlador, la región de factibilidad obtenida es menor que la de los dos problemas anteriores.

Además, se compara el coste obtenido $J = \sum(y_k^T Q y_k + u_k^T R u_k)$ para las trayectorias con cada uno de los tres controladores desde varios estados iniciales incluidos en las tres regiones de factibilidad hasta alcanzar el origen. El cuadro 1 muestra el resultado obtenido. Puede comprobarse como, en el peor de los casos, la trayectoria para el controlador con $N = 2$ es hasta un 26,8 % más desfavorable que para $N = 5$. Sin embargo, el controlador $N = 5, L = 2$, que se obtiene con un coste similar, es únicamente un 0,4 % más desfavorable.

9. Conclusiones

En este trabajo, se ha planteado un procedimiento para el cálculo de la solución explícita del problema de control predictivo con restricciones poliédricas no convexas. La metodología está basada en el cálculo de la solución explícita de los problemas definidos con restricciones poliédricas y en la intersección, eliminación de soluciones redundantes (mediante programación de suma de cuadrados) y unión de las regiones de las diferentes particiones. Se ha realizado una justificación de la metodología propuesta frente a la resolución y comparación de costes de los subproblemas convexos basada en la reducción de coste del algoritmo en línea. Además, se ha introducido una solución subóptima del problema, que reduce tanto el coste en línea como el número de problemas de optimización a resolver fuera de línea. Por último,se muestra el funcionamiento de la metodología propuesta mediante un ejemplo, sobre el que se discuten todos los puntos anteriores.

English Summary

Explicit Solution to Predictive Control of Linear Systems Subject to Non-convex Constraints

Abstract

This paper presents an explicit solution to predictive control of linear systems subject to non-convex polyhedral constraints. The constraints are modeled as the union of finite number of polyhedra. The algorithm is based on the computation of the explicit solution of the problems subject to convex constraints defined by those polyhedra. The regions of the partitions obtained this way are intersected such that the new set of regions has as many possible solutions as convex problems are solved. By means of sum-of-squares programming, the non-optimal solutions are removed. Then a union of regions which share the same set of solutions is performed. Furthermore a suboptimal solution which can be used when the original methodology is too expensive is presented. Finally, some results are shown in an example.

Keywords:
Predictive Control Multiparametric Programming Non-convex Constraints Sum-of-squares.

Agradecimientos

Este trabajo ha sido realizado parcialmente gracias al apoyo de los proyectos DPI2008- 02133 y DPI2008-06731/DPI Ministerio de Ciencia e Innovación.

Referencias

Avis, D., Fukuda, K., 1996. Reverse search for enumeration. Discrete Applied Mathematics 65, 21–46.

Bemporad, A., Fukuda, K., Torrisi, F. D., 2001. Convexity recognition of the union of polyhedra. Computational Geometry 18 (3), 141–154.

Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E. N., 2002. The explicit linear quadratic regulator for constrained systems. Automatica 38 (1), 3–20.

Bertsekas, D. P., Yu, H., 2009. A unifying polyhedral approximation framework for convex optimization.

Bochnak, J., Coste, M., Roy, M. F., 1998. Real algebraic geometry. Springer.

Borrelli, F., Baotic, M., Bemporad, A., Morari, M., 2005. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. Automatica 41 (10), 1709–1721.

Boyd, S., Ghaoui, L. E., Feron, E., Balakrishnan, V., 1994. Linear matrix inequalities in system and control theory.

Camacho, E., Bordons, C., 2004. Control predictivo: Pasado, presente y futuro. Revista Iberoamericana de Automática e Informática Industrial 1 (3), 5–28.

Camacho, E., Bordons, C., 2007. Nonlinear model predictive control: An introductory review. Assessment and Future Directions of Nonlinear Model Predictive Control, 1–16.

Ferrez, J., Fukuda, K., Liebling, T. M., 2001. Cuts, zonotopes and arrangements. Preprint, Swiss Federal Institute of Technology, Lausanne.

Fruzzetti, K. P., Palazoglu, A., McDonald, K. A., 1997. Nonlinear model predictive control using Hammerstein models. Journal of Process Control 7 (1), 31–41.

Geyer, T., Torrisi, F. D., Morari, M., 2008. Optimal complexity reduction of polyhedral piecewise affine systems. Automatica 44 (7), 1728–1740.

Heemels, W., Schutter, B. D., Bemporad, A., 2001. Equivalence of hybrid dynamical models. Automatica 37 (7), 1085–1091.

Johansen, T. A., Jackson, W., Schreiber, R., Tøndel, P., 2007. Hardware synthesis of explicit model predictive controllers. IEEE Transactions on Control Systems Technology 15 (1), 191.

Kuchar, J. K., Yang, L. C., 2000. A review of conflict detection and resolution modeling methods. IEEE Transactions on Intelligent Transportation Systems 1 (4), 179–189.

Kurzhanski, A. B., 2005. Dynamic optimization for nonlinear target control synthesis. Nonlinear Control Systems 2004, 21.

López, D., Gómez-Bravo, F., Cuesta, F., Ollero, A., 2006. Planificación de trayectorias con el algoritmo RRT. Aplicación a robots no holónomos. Revista Iberoamericana de Automática e Informática Industrial 3 (3), 56–67.

Mayne, D. Q., Rakovic, S. V., 2002. Optimal control of constrained piecewise affine discrete time systems using reverse transformation. In: Decision and Control, 2002, Proceedings of the 41st IEEE Conference on. Vol. 2.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., Scokaert, P. O. M., 2000. Constrained model predictive control: Stability and optimality. Automatica 36, 789–814.

Pappas, G. J., Lygeros, J., Godbole, D. N., 1995. Stabilization and tracking of feedback linearizable systems under input constraints. In: Proceedings of the IEEE Conference on Decision and Control. Citeseer, pp. 596–601.

Parrilo, P. A., 2000. Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. Ph.D. thesis, California Institute of Technology.

Patwardhan, R. S., Lakshminarayanan, S., Shah, S. L., 1998. Constrained nonlinear MPC using Hammerstein and Wiener models: PLS framework. AICHE Journal 44 (7), 1611–1622.

Pérez, E., Ariño, C., Blasco, F. X., Martínez, M. A., 2011. Maximal closed loop admissible set for linear systems with non-convex polyhedral constraints. Journal of Process Control 21 (4), 529–537.

Rakovic, S. V., Mayne, D. Q., 2007. Robust model predictive control for obstacle avoidance: discrete time case. Lecture Notes in Control and Information Sciences 358, 617.

Slotine, J. J. E., Li, W., 1991. Applied nonlinear control. Prentice-Hall Englewood Cliffs, NJ.

Stengle, G., 1973. A Nullstellensatz and a Positivstellensatz in semialgebraic geometry. Mathematische Annalen 207 (2), 87–97.

Sturm, J. F., 1999. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Optimization Methods and Software 11 (1), 625–653.

Tøndel, P., Johansen, T. A., Bemporad, A., 2003a. An algorithm for multiparametric quadratic programming and explicit MPC solutions. Automatica 39 (3), 489–497.

Tøndel, P., Johansen, T. A., Bemporad, A., 2003b. Evaluation of piecewise affine control via binary search tree. Automatica 39 (5), 945–950.

Vandenberghe, L., Boyd, S., 1996. Semidefinite programming. SIAM Review 38 (1), 49–95.