Capítulo 1

Contexto y estado del arte del proyecto

En este capítulo, por un lado, se va introducir el proyecto europeo *MOBY-DIC*, en el cual está enmarcado este proyecto y por el otro se va a describir el estado del arte de las técnicas que pueden utilizarse para proporcionar funciones no lineales del tipo *PWA*

1.1. Contexto. El proyecto europeo *MOBY-DIC*

Este Proyecto de Fin de Carrera está enmarcado en el proyecto europeo *MOBY-DIC* [1] en el que participa parte del profesorado del *Departamento de Electrónica y Electromagnetismo* de la *Universidad de Sevilla*. En dicho departamento he disfrutado de una beca de colaboración financiada por el *Ministerio de Educación*.

El proyecto *MOBY-DIC* está investigando y desarrollando un paradigma que proveerá de una cadena de diseño de circuitos digitales para sistemas de control empotrados. Esto se desarrollará en un marco único tanto para el modelado de procesos físicos como para el diseño e implementación en circuitos empotrados de algoritmos de control (controladores), evaluándose las propiedades de comportamiento de los sistemas en general. Los circuitos se implementarán o bien en *FPGAS* o bien en circuitos integrados dedicados.

MOBY-DIC proporcionará un flujo de diseño desarrollando una metodología basada en funciones afines a trozos que proveen una estructura de control relativamente flexible y pueden mapearse en arquitecturas digitales que ocupan poco tamaño y onsumen poca potencia.

Dependiendo de las técnicas de diseño de funciones a afines trozos (*PWA*), el proyecto *MOBY-DIC* conectará el diseño abstracto del algoritmo de control con su realización circuital, permitiendo así la síntesis de sistemas electrónicos que están directamente basados en modelos matemáticos de los entornos físicos en los cuales los circuitos serán empotrados.

Cuando se cumplan sus objetivos, *MOBY-DIC* significará un importante avance en la investigación europea en el diseño de sistemas de control empotrados. El flujo de diseño automático de *MOBY-DIC* satisfará una demanda muy requerida por la industria.

El objetivo principal de *MOBY-DIC* consiste en el desarrollo de formalismos *PWA* que sirvan de puente entre las comunidades que desarrollan controladores y las comunidades que desarrollan circuitos empotrados. Este primer objetivo conforma el fundamento teórico del proyecto *MOBY-DIC:* novedosas técnicas matemáticas para sintetizar un controlador usando funciones *PWA,* técnicas de análisis y diseño

que conecten directamente la descripción de control con la arquitectura correspondiente a su implementación digital.

El segundo objetivo consiste en la creacción de una cadena completa de herramientas que soporten el diseño. El controlador empotrado se representa, primero, a través de su modelo dinámico matemático en un entorno de simulación como la herramienta *Simulink* de *Matlab* (*The Mathworks, Inc).* La idea consiste en generar un código *RTL* (register transfer level) para el modelo matemático dinámico del controlador *PWA*, siendo éste el objetivo perseguido por muchos investigadores en el ámbito de la tecnología de síntesis de algoritmos. La herramienta de generación del código *RTL* generada específicamente para *MOBY-DIC* deberá integrarse en entornos estándar de diseño de *FPGAS* y *ASICS* como *Xilinx ISE* [2] , *Mentor Graphics* [3] o *Cadence* [4].

El tercer y último objetivo consiste en aplicar las técnicas logradas por los objetivos anteriores en tres casos de estudio. Los casos de estudio de *MOBY-DIC* han sido estrictamente seleccionandos en el campo de la automoción:

- El primer caso está relacionado con la monitorización de las baterías de vehículos híbridos.
- El segundo está relacionado con el diseño de un sensor virtual para aplicaciones dinámicas de vehículos.
- El tercer caso está relacionado con controlador de acceleración y freno de la velocidad de crucero.

1.2. Estado del arte: funciones *PWA*

1.2.1. Definiciones básicas

Se considera una función $f_{PWA} : D \rightarrow \mathbb{R}$ definida en el dominio compacto *n*-dimensional $D = \{x \in \mathbb{R}^n : -1 \le x_i \le x_+ < 1, i = 1,...,n \}$ donde x_+ está definido de la siguiente manera:

Puesto que todas las componentes del vector entrada pertenecen al intervalo $[-1,1)$, se pueden representar por números enteros de b bits codificando sólo la parte decimal. Una vez que se elije la precisión (por ejemplo b) se puede definir $x_+ = 1 - 2^{-b+1}$. El caso más general, donde $D = \{x \in \mathbb{R}^n : a_i \le x_i \le b_i, i = 1,...,n \}$, siempre se puede reportar a este caso particular.

El dominio D se particiona mediante *M* aristas e_j en *N* politopos P_i, tal que $U_{i=1}^{N} P_i = D$ y $P_k \cap P_j = \varnothing$ para $j \neq k$. Cada arista es un hiperplano (n-1)-dimensional de la forma $h_j^T x + k_j = 0$ donde $h_j \in \mathbb{R}^n$ y $k_j \in \mathbb{R}$, que divide el dominio en dos partes. Entonces, un politopo está definido por un subconjunto de aristas.

Agrupando los índices de las aristas que definen el i-ésimo politopo en el conjunto E_i, podemos decir que:

$$P_i = \{x \in D : \pm(h_j^T x + k_j) \le 0, j \in E_i\} \qquad \textbf{Ecuación 1}$$

El signo de la inecuación $\pm(h_j^T x + k_j) \le 0$ debe ser elegido para evitar politopos vacíos o duplicados.

La función f_{PWA} es afín en cada politopo P_i:

$$f_{PWA}(x) = \mathbf{f}_i^T x + g_i, \qquad x \in P_i \qquad \textbf{Ecuación 2}$$

Donde $\mathbf{f}_i \in \mathbb{R}^n$ y $g_i \in \mathbb{R}$.

La *Figura 1* muestra un dominio bidimensional $D = [0,1]^2$ particionado por 4 aristas en 5 politopos. Si se considera el politopo, P_5 se puede ver que pertenece a la zona del dominio D definida por las aristas e_1 y e_2, entonces $E_5 = \{1,2\}$. De la misma manera se obtienen $E_1 = \{1,4\}$, $E_2 = \{1,3,4\}$, $E_4 = \{1,3\}$, $E_3 = \{1,2\}$. El politopo P_5 se define como $\left\{(x_1,x_2) \in [0,1]^2 : -(h_1^T x + k_1) \le 0, +(h_2^T x + k_2) \le 0\right\}$.

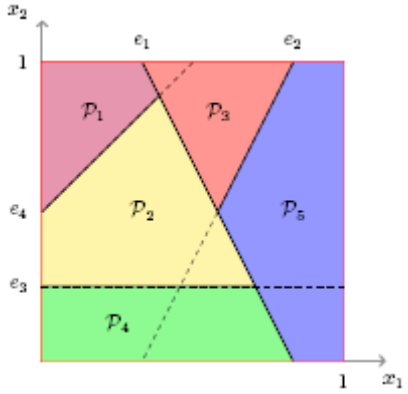

Figura 1. Ejemplo de un dominio bidimensional particionado en politopos

1.2.2. Funciones *PWA* simplificadas (*PWAS*)

La definición de una función *PWAS* está estrictamente relacionada con el dominio. En particular, las funciones PWAS $f_{PWA} : D \rightarrow \mathbb{R}$ están definidas sobre un dominio híper-rectangular $D = \{x \in \mathbb{R}^n : a \le x \le b\}$.

Dado un conjunto de $n+1$ puntos $v_o, v_1, ..., v_n$, llamados vértices, un *simplex* en $\mathbb{R}^n$ es una combinación convexa de vértices:

$$S(v_0,...,v_n) = \left\{ x : x = \sum_{i=0}^{n} \mu_i v_i, D \le \mu_i \le 1, i = 0..n, \sum \mu_i = 1 \right\} \qquad \textbf{Ecuación 3}$$

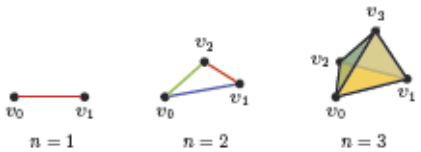

Figura 2. Ejemplo de *simplex* en 1,2 y 3 dimensiones

Un simplex $S(v_0,...,v_n)$ puede también ser representado por hiperplanos definidos por sus aristas: $S(v_0,...,v_n) = \{x : H_x \le K\}$

H y *K* se definen directamente de las inecuaciones:

$$\begin{bmatrix} 1 & \dots & 1 \\ v_0 & \dots & v_n \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ x \end{bmatrix} \ge 0 \qquad \textbf{Ecuación 4}$$

El dominio D se particiona en *simplexs* como sigue. Todas las componentes dimensionales $x_i \in [a_i,b_i]$ de D se dividen en m_i subintervalos, tomando m_{i+1} puntos $x_i^0 = a_i, x_i^1 ..., x_i^{m_i} = b_i \; i = 1..n$. Consecuentemente el dominio se dividen en $\prod_{i=1}^{n} m_i$ hiperrectángulos y contiene $N = \prod_{i=1}^{n} (m_i + 1)$ vértices reunidos en conjuntos v_x. Cada vértice tiene coordenadas $v_k = (x_1^{K_1}, x_2^{K_2}, ..., x_n^{K_n}), K_i \in \{0,...,m_i\} \; i = 1,...,n$. Las coordenadas de los vértices de cada eje se almacenan en n vectores $p_i = (x_i^0, ..., x_i^{m_i})$. Para las coordenadas de los vértices v_k, el subíndice define la componente del dominio y el superíndice establece un orden incremental de las componentes de ordenadas (*Figura 3*). Cada rectángulo se divide en $n!$ *simplexs* no solapados.

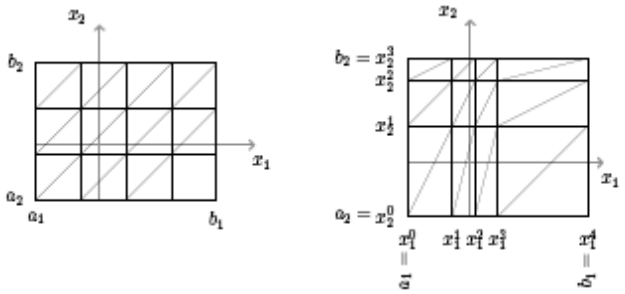

Figura 3. Ejemplo de un dominio bidimensional particionado en *simplex*

La partición resultante, que se llama partición *simplicial,* es una triangulación de D formada por una partición hiper-rectangular y una diagonal de norte a este. La posición de los vértices la delimita una única configuración de aristas H que definen la partición. Por lo tanto, la partición está completamente definida por una tripleta (a_i, b_i, m_i). Cuando la distancia entre los vértices es uniforme: $x_i^{k_{i+1}} - x_i^{k_i} = \frac{b_i - a_i}{m_i} = p_i$, y se obtiene una simplicial uniforme. Cuando la distacia entre los vértices no es la misma, se obtiene una partición simplicial no uniforme.

Una función *PWAS* f_{PWA} se define como una función que asigna una función *PWA* a cada simplex de la partición simplicial de D.

En [5] se describe cómo implementar mediante circuitos digitales funciones *PWAS* para particiones no uniformes.

Se considera un dominio $D_x = [0,1]^n$ dividido en subintervalos cuyo tamaño es una potencia de 2 negativa : $X_i^{k_i+1} - X_i^{k_i} = 2^{-q_i^{ki}}$, donde $q_i^{k_i}$ es un entero positivo que puede ser diferente para cada intervalo.

Por otro lado, se considera otro dominio, D_z, contiene N vértices $v_k^z \in V_z$ con coordenadas naturales $v_k^z = (k_1, k_2, ..., k_n)^T$, donde $K_i \in \{0,...,m_i\}, i = 1..n$, donde el tamaño de cada intervalo es unitario (y por lo tanto, todos los intervalos son iguales).

Cada componente dimensional de los dominios D_x y D_z está dividido en m_i subintervalos. Pero mientras la partición simplicial es no uniforme para D_x.

Se define uniforme para D_z y se describe una transformación $T(x)$ (*Figura 4*) para pasar del dominio D_x al D_z. De esta manera, una solución de circuito para particiones uniformes puede también emplearse en particiones no uniformes.

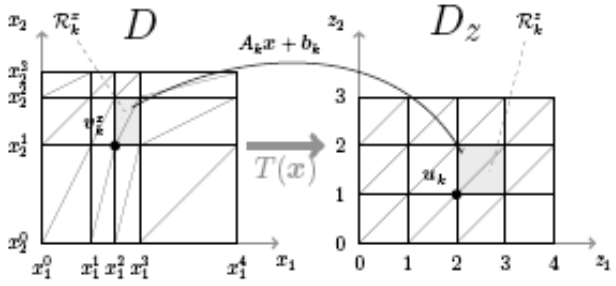

Figura 4. Ejemplo bidimensional de una partición no uniforme y su transformada uniforme

La transformación $T(x)$ se construye de la siguiente forma. Se define un hiper-cuadrado R_K^z de D_z con el siguiente producto cartesiano: $R_K^z = [K_1, K_1+1] \times [K_2, K_2+1] \times \ldots \times [K_n, K_n+1]$. Análogamente, se define un hiper-rectángulo R_K^x de D_x con $R_K^X = [X_1^{K_1}, X_1^{K_1+1}] \times \ldots \times [X_n^{K_n}, X_n^{K_n+1}]$.

La función $T(x)$ mapea cada hiper-rectángulo R_K^X en cada hiper-cuadrado R_K^z. Lo interesante es que la función $T(x)$ es también PWA sobre el dominio D_X y tiene la forma:
$T(x) = A_K x + b_K,\ x \in R_K^X$ con A_K diagonal e invertible.

1.2.3. Evaluación de funciones *PWA*

1.2.3.1. Evaluación genérica: Árbol de búsqueda binaria

Para poder evaluar $f_{PWA}(x)$ se necesita:

1) Encontrar el índice i tal que $x \in P_i$.
2) Evaluar la expresión afín $\mathbf{f}_i^T x + g_i$.

Mientras que el punto 2) es fácil de implementar (recuperar $\mathbf{f}_i$ **y** g_i de una memoria e implementar una suma y una multiplicación), el primer paso requiere una mención especial. En muchas aplicaciones, el número de politopos crece exponencialmente con el número de dimensiones y con el número de aristas [6]. Por lo tanto, será computacionalmente complejo comprobar si $x \in P_i$ para cada politopo en D comparando x con cada arista de P_i, ya que el número de comparaciones $h_j^T x + k_j \le 0$ crece exponencialmente también.

Para resolver ese problema también conocido como el problema de la localización del punto, en [7] los autores proponen construir un árbol de búsqueda binaria *off-line*, donde cada nodo que no sea una hoja represente una arista y las hojas contengan el índice i del politopo. Explorando el árbol o*n-line*, desde la raíz a la hoja, es posible localizar el politopo que contiene el vector de entrada evaluando un número relativamente pequeño de comparaciones. El árbol se construye minimizando su profundidad. En ese sentido, el tiempo necesario para resolver *on-line* el problema de la localización del punto se reduce respecto a una búsqueda combinacional.

El árbol de búsqueda binaria asociado con la partición del dominio mostrado en la *Figura 1* se muestra en la *Figura 5*. Dado el punto x, para explorar el árbol se empieza por e_1 evaluando $h_1^T x + k_1 \le 0$. Si esa condición es verdadera se selecciona la rama marcada con +, en otro caso se selecciona la rama marcada con -. Este procedimiento se repite hasta llegar a localizar la hoja. Como mucho se necesitan evaluar 3 expresiones afines para encontrar el politopo que contiene a la entrada. Por otro lado si se realiza una búsqueda combinacional, el número de expresiones afines que se deben evaluar para localizar el politopo será igual a 4, es decir, el número total de aristas. Hay que remarcar

que la ganancia, en términos del número de expresiones que hay que evaluar, será mayor cuanto mayor sea el incremento del número de aristas y de las dimensiones del dominio.

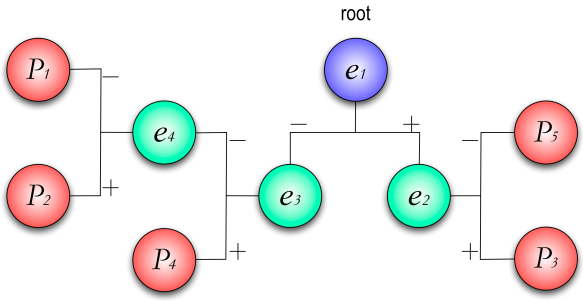

Figura 5. Árbol de búsqueda binaria construido a partir de la *Figura 1*

El problema de la localización del punto se reduce a evaluar interactivamente funciones afines cuyos coeficientes (h_j, k_j) son valores reales.

El orden de operaciones necesaria para calcular el valor de la función de un punto dado es $\Theta(n \cdot \log_2 i)$ siendo n el número de entradas e i el número de politopos.

1.2.3.2. Evaluación particular: Funciones *PWAS*

Se considera una función *PWAS* $f_{PWA} : D_z \rightarrow \mathbb{R}$, definida sobre un dominio n-dimensional apropiadamente escalado D_z (es decir, particionado de forma uniforme) y se quiere calcular $f_{PWA}(z)$. Las coordenadas del vértice del hiper-cuadrado más cercano al origen que contiene al punto $z \in D_z$ puede ser hallado extrayendo la parte entera de z. La posición exacta dentro del hiper-cuadrado depende de la parte decimal (δ) como se ve en la figura:

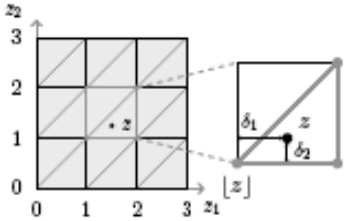

Figura 6. Ejemplo de una partición simplicial uniforme

El valor de $f_{PWA}(z)$ puede ser calculado como una interpolación lineal de $n+1$ valores de los vértices del simplex que contiene a z.

El orden de operaciones necesaria para calcular el valor de la función de un punto dado es $\Theta(n)$ para todos los m_i.

1.3. Uso de funciones *PWA* en controladores

El uso de sistemas *PWA* y controladores en el ámbito del control empezó con un seminario de Eduardo Sontag en 1981 [8]. Tuvieron que pasar casi dos décadas para que este trabajo fuera considerado relevante para el análisis de sistemas y el diseño de controladores. La llegada de los sistemas computacionales modernos y eficientes [9] estimuló el desarrollo de los análisis *PWA* y las técnicas de síntesis. Especialmente el uso de inecuaciones matriciales lineales (*LMIs*), que pueden ser resueltas eficientemente usando técnicas de programación convexa ([10], [11] y [12]) jugaron un papel crucial en el diseño de controladores y estimadores *PWA*. Los análisis de estabilidad y robustez de los

sistemas *PWA* ([13], [14]) fue lo primero que adaptó a las *LMIs* en este contexto. El diseño basado en LMI de controladores estables (y robustos) ha sido investigado en [15],[16],[17],[18] y [19], mientras que el diseño de aproximaciones para sistemas PWA ha sido investigado por [20][21] y [22]. El estudio de controladores PWA dinámicos obtenidos a través de la interconexión de aproximaciones y leyes de retroalimentación son estudiados por [23][24].

Pero todos esos no proporcionan unas aproximaciones sistemáticas para obtener el controlador deseado.

Por otro lado, una técnica de control utilizada con éxito desde los 80 ha sido el *Control Predictivo* basado en el modelo *MPC (Model Predictive Control).* Sin embargo, esta técnica se ha considerado tradicionalmente como una solución eficiente para el control de procesos lentos porque necesita resolver problemas de optimización en línea (es decir, a la vez que se está controlando el proceso). La idea que ha propuesto *A. Bemporad et als* [25] es resolver este problema de optimización fuera de línea para todos los valores de las variables de estado dentro de unos rangos y obtener la salida de control de forma explícita como una función *PWA*. De hecho existe una *toolbo*x en *Matlab (Hybrid Toolbox [26])* que permite automatizar esta solución.

1.4. Conclusiones

La implementación mediante circuitos digitales de funciones *PWA* es muy interesante para desarrollar controladores empotrados y resuelven multitud de problemas de control no lineales.

La implementación de funciones *PWAS* es más sistemática que la implementación que emplea árboles de búsqueda binaria. Sin embargo, las aproximaciones *PWAS*, tanto uniformes como no uniformes, padecen un problema que se conoce como "*curse of dimensionality*" o "*maldición de la dimensionalidad*". Este problema significa que el número de politopos en el que se divide el dominio crece exponencialmente con el número de entradas del sistema. Por lo tanto, este tipo de soluciones sólo son eficientes para sistemas con un número de entradas reducido.

Como alternativa a las funciones *PWA* evaluadas mediante árboles de búsqueda binarios y a las funciones *PWAS*, este Proyecto Fin de Carrera va a analizar otra solución: las funciones *PWA* jerárquicas, obtenidas como combinación lineal de funciones *PWA*.

Definir una jerarquía en un sistema, en particular en un controlador, es siempre difícil. Para reducir esta complejidad haremos uso de la lógica difusa que nos permite, por un lado, definir controladores *PWA* a partir de conocimiento heurístico expresado lingüísticamente y, por otro, entender y expresar lingüísticamente el comportamiento de controladores *PWA* diseñados por ingenieros de control.