5 Conclusiones y Desarrollos Futuros.

En este capítulo se expondrán las conclusiones que se desprenden del estudio realizado en los capítulos anteriores, así como se plantearán posibles desarrollos futuros a las partes que se han analizado.

5.1 Banco de Pruebas.

Respecto a los resultados obtenidos durante el banco de pruebas, podemos indicar, de forma resumida lo siguiente:

- Aritmética Flotante.
  - Para la suma se usa el 88.08 % del tiempo en los cálculos; realizándose 1,5035 *MFLOPS*.
  - Para el producto, el 87.74 %; realizándose 1,4705 *MFLOPS*.

Gracias a la similitud de los resultados se deduce que en la arquitectura *hardware* del microprocesador de *ARM* aparece una unidad aritmético – lógica especializada en productos. Puesto que estamos hablando de, aproximadamente, 1,5 MFLOPS se puede afirmar que este sistema ejecuta una instrucción bajo aritmética flotante en menos de 1 μs. Estos resultados son coherentes con los proporcionados por los manuales del fabricante del *chip*.

- Aritmética Entera.
  - Para la suma se usa el 88.08 % del tiempo en los cálculos; realizándose 1,5035 *MINTS*.
  - Para el producto, el 87.74 %; realizándose 1,4705 *MINTS*.

  Los comentarios al efecto son análogos a los que se han escrito para el caso flotante.

- Ecuaciones en diferencias: Filtros.
  - Usando un orden de (1, 1) se invierten 21 ms por iteración.
  - En el caso de orden (50, 50) se necesitan 22.7 ms por iteración.

De acuerdo con los resultados, el orden no posee una influencia de primera magnitud sobre el tiempo invertido en el cálculo de la ecuación en diferencias del filtro. De todas formas 21 ms excede, con creces, el límite impuesto por diseño de 1 ms para el bucle de control de bajo nivel. Por otra parte, sí podría ser usado este algoritmo para bucles de mayor nivel, puesto que requieren constantes de tiempo más largas.

- Cálculos Estadísticos.
  - Se invierten 20.4 ms por muestra para calcular el estadístico de segundo orden.
  - Se precisan 19.9 ms por muestra para obtener el estadístico de tercer orden.
  - Se requieren 20.7 ms por muestra para deducir el estadístico de quinto orden.
  - Y se miden 21.4 ms por muestra para hallar el estadístico de décimo orden.

Por la similitud de estos resultados con los del caso anterior, se desprenden conclusiones similares. Un ejemplo interesante de bucle de nivel medio sería el que sintoniza o ajusta automáticamente el modelo del sistema que se pretende controlar.

- Inversión de Matrices.
  - Para orden sexto el tiempo requerido queda acotado por 0.565 s.

Con los tiempos medidos resulta prohibitivo implementar los algoritmos de inversión matricial en tiempo real considerando tiempos de 1 ms para el bucle de control de bajo nivel.

- Producto de Matrices.
  - El tiempo invertido para una iteración es, aproximadamente, de 33 μs; es decir, 2 órdenes de magnitud menor que el máximo exigido por diseño para el controlador, que era de 1 ms.

Estos resultados son gratamente interesantes porque abren la posibilidad de utilizar el algoritmo de control denominado *"gain-scheduling"* para el bucle cuyo nivel de trabajo se sitúa más cercano al sistema real.

5.2 Técnicas de Prototipado Rápido.

Desde el punto de vista del ingeniero de sistemas, la herramienta *"Real Time Workshop"*[1] proporciona técnicas de diseño de prototipado rápido, las cuales permiten desarrollar aplicaciones con un coste humano, de tiempo y de otros tipos de recursos relativamente reducido.

Gracias al interfaz que proporciona se permite la depuración directa del código máquina embebido a través de su correspondiente diagrama *Simulink*, lo que facilita enormemente la visión de conjunto que hace falta en todo proceso de diseño. *RTW* proporciona un conjunto de plantillas propias que construyen el puente o interfaz entre el entorno de descripción de alto nivel –*Simulink*– y el compilador cruzado concreto; lo que, junto con la depuración directa,

[1] Por simplicidad se le denotará como *RTW*

constituye una herramienta de trabajo tal que todo el proceso interno, visible en las técnicas de desarrollo tradicionales, se torna trasparente al diseñador.

Por otro lado, *RTW* facilita la generación de las plantillas de usuario; éstas, una vez elaboradas por el diseñador, realizan el puente que comunica *Simulink* con la máquina concreta con la que se esté trabajando; además, *RTW* se ha diseñado bajo la arquitectura de código abierto, es decir, que es independiente de la plataforma en la que esté instalado. En conjunto proporciona una gran flexibilidad de cara al proceso de diseño.

Si bien es cierto que la metodología que hay que seguir para construir las plantillas propias no es sencillo, en cuanto se pretenda que comuniquen *RTW* con el compilador cruzado de trabajo. Esto ocurre de esta manera debido a que el paso de parámetros, mediante línea de comandos, es propio de cada compilador y de cada sistema operativo, obligando a la solicitud de documentación técnica concreta de cada fabricante.

Motorola, con su sistema de desarrollo basado en el microcontrolador *"MPC555"*, ha dado con un conjunto de ficheros que constituye el interfaz entre *RTW* y el compilador *"MetroWerks CodeWarrior"* para el descargador *"Phy-Core555"*. Entre estos archivos aparecen algunas funciones de gran interés que proporcionan la forma de establecer los parámetros concretos de la comunicación entre ambas partes; estas funciones se encuentran especificadas bajo el formato de *Matlab ".p"*, es decir, precompiladas, por lo tanto, no facilitan la opción de estudiar su funcionamiento interno con la intención de extender su ámbito de aplicación al caso concreto que nos compete, o sea, para el compilador *"CodeWarrior"* con el descargador *"DebugRel"*.

A raíz de este escenario, resultaría interesante estudiar la manera en la que se define el descargador, tal que se concrete *"Phy-Core555"* como *"DebugRel"*; de esta forma, la cadena de generación automática de código se extendería desde *RTW* hasta el código máquina del microprocesador de *"ARM Limited" ARM7TDMI*.

5.3 Control de Aeronaves.

Gracias a la bibliografía relativa a este apartado es posible afirmar que las técnicas de control más aptas al caso de las aeronaves se centran en los métodos *LQR*[2] y *SDRE*[3]. El primero de ellos se usa mediante la extensión a sistemas no lineales, la cual se basa en la definición de un conjunto de puntos de trabajo en los que se linealiza el modelo no lineal del helicóptero; ofrece una solución robusta cuando la referencia no cambia con demasiada rapidez. El segundo

[2] *"LQR"*: *"Linear Quadratic Regulator"*.

de ellos requiere la resolución explícita de la ecuación de *Ricatti* en función del estado; pero, si este problema se ha solventado, permite su uso en situaciones con perturbaciones acusadas y rápidos cambios de referencia manteniendo correctas características de robustez y estabilidad.

Un problema significativo, que atañe a estos controladores, estriba en que requieren de la sintonización de parámetros que ponderan la dinámica y el intervalo de aplicación de las variables que intervienen en el modelo. Esta sintonización sí posee un método claro y probado siempre que se esté trabajando sobre un sistema real modelado linealmente. Debido a que una aeronave presenta un comportamiento dinámico fuertemente no lineal, el hecho de aceptar un modelo lineal como representativo del mismo conlleva a graves errores. Esto nos obliga a utilizar modelos no lineales, para los cuales no existe un método sistemático que sintonice los parámetros de las matrices de ponderación, con el consiguiente planteamiento del esquema *"prueba y error"* –muy lento, tedioso y sin garantías de éxito–.

A causa de lo enunciado en el párrafo anterior, resultaría interesante elaborar un estudio sobre la influencia de los parámetros de peso en las variables de un modelo no lineal con el objetivo de constatar un método, que otorgue unas mínimas garantías de éxito y de cumplimiento de plazos. Un punto de comienzo sería la formulación del método *SDRE* puesto que plantea expresiones que relacionan las variables del modelo con los parámetros de peso de forma directa.

Plantear, como alternativa a los métodos de control expuestos, la posibilidad de estudio futuro de las redes neuronales resulta interesante y, a la vez, productivo. Esta afirmación se desprende de los artículos analizados, puesto que en ellos se muestran experimentos y simulaciones que avalan la estabilidad y robustez de estas redes, aunque se trabaje en entornos ruidosos, no estructurados, perturbados, no lineales y de rápidos cambios en las referencias como es el control de aeronaves en campo.

Tras el análisis de los modelos dinámicos encontrados en diversos artículos, se optó por el que ofrecía un mayor realismo, siempre y cuando su aplicación fuese relativamente directa comparada con el resto; muchos de los modelos aparecían incompletos o expresados bajo una formulación excesivamente confusa, otros contemplaban una dinámica muy reducida. En resumen, se eligió un modelo con 11 variables de estado y 4 variables de control; esto dio lugar a una matriz de realimentación del vector de estados con 44 entradas, lo que se traduce en 352 bytes para almacenar la matriz de control adecuada a cada punto de trabajo del método *LQR*, que fue el que finalmente se implantó.

---

[3] *"SDRE"*: *"State-Dependent Ricatti Equation"*.

Gracias a las medidas llevadas a cabo sobre la capacidad de computación del *ARM7TDMI*, relativas a la multiplicación matricial y a las operaciones básicas, y a la observación del código fuente para *"Controlador.mdl"*, generado automáticamente por *RTW*, se puede estimar que el tiempo necesario para llevar a cabo una iteración del algoritmo de control no excederá el límite impuesto por diseño, es decir, de un milisegundo.

La aplicación del método *LQR* con ganancia tabulada limita la evolución dinámica del modelo considerado del sistema, siempre que se busque una regulación adecuada. A mayor grado de realismo, se exige un mayor número de puntos de trabajo; lo que obliga a un mayor uso de los recursos *hardware*, muy limitados en un sistema embebido. Para el modelo concreto que se ha utilizado, con 9 puntos, se requiere del 20 % de la memoria del sistema. No son muchos puntos. Pero, en un caso real, con más puntos, el límite impuesto por la cantidad de memoria disponible del sistema puede volverse un factor crítico.

En cambio, el método *SDRE* no presenta esta limitación, al ser adaptativo explícitamente; pero sí se encuentra fuertemente restringido por la necesidad de obtener una expresión explícita de la ecuación de *Ricatti* para cualquier valor del vector de estados, dentro de su intervalo de trabajo. Gracias a que la formulación de la ecuación que modela al sistema admite múltiples representaciones, si no existe una solución explícita bajo una de ellas, es posible que sí exista bajo otra, aunque no se den garantías para ello de carácter formal –siendo éste un tema de estudio abierto–.