

Filtrado Óptimo y Adaptativo
Curso de Doctorado en Tecnología de las
Comunicaciones
Universidad Carlos III de Madrid

**Implementación del algoritmo QRD-RLS en un
array sistólico:
estudio e implementación**

Pablo Barrera González
<*pbarrera@tsc.uc3m.es*>

20 de junio de 2003

1. Introducción

En el presente trabajo se va a realizar una implementación del algoritmo *QRD-RLS* en un *array sistólico*, viendo que operaciones que hay que realizar, paso a paso, y relacionándolas con las expresiones generales del *QRD-RLS*.

El algoritmo *RLS* es uno de los algoritmos tradicionales de descenso según gradiente. Presenta ventajas frente al *LMS* dado que obtiene una menor cantidad de ruido una vez que el algoritmo ha convergido. En su contra tiene una velocidad de convergencia menor y una mayor carga computacional. Con el *array sistólico* lo que se intenta es realizar los cálculos necesarios para este algoritmo de forma paralela para que una implementación *hardware* sea más eficiente que un *RLS* tradicional sobre un *DSP*.

2. QRD-RLS

Para poder implementar un *RLS* en un *array sistólico* es necesario, en primer lugar, modificar el desarrollo del *RLS*. En el *RLS* básico se obtiene que el vector de pesos del filtro puede obtenerse a partir de la matriz de datos de la siguiente forma:

$$R\mathbf{w} = \mathbf{p} \quad (1)$$

$$\mathbf{w} = R^{-1}\mathbf{p} \quad (2)$$

Esto implica ir actualizando continuamente la matriz R y a su vez calcular su inversa en cada paso. Dado el elevado coste computacional de esta aproximación se tiene a emplear una variante por la que no se almacena R sino R^{-1} , actualizándola con los valores de entrada paso a paso.

Otra posible simplificación consiste en emplear una descomposición *QR* por la que se obtenga una R diagonal superior, simplificando la obtención de \mathbf{w} en la expresión (2). En el *array sistólico* esta descomposición *QR* se realiza mediante la utilización de rotaciones de Givens, encargadas de diagonalizar la matriz cada vez que un nuevo dato entra con un coste computacional muy bajo.

3. Implementación en un array sistólico

Para realizar una implementación paralela del *RLS*, se puede emplear un *array sistólico*. Viendo las expresiones del último apartado podemos dividir el cálculo en dos partes claramente diferenciadas:

- Por un lado tenemos el cálculo de la matriz R empleando las rotaciones de Givens.
- Un vez obtenida la matriz R en un instante determinado, podemos, mediante *retro-sustitución*, calcular los valores del vector de pesos \mathbf{w} .

En las figuras 1 y 2 pueden verse los esquemas de estas dos partes, separadas. Ambas están formadas por elementos sencillos que funcionan con el mismo reloj, de tal forma que todos

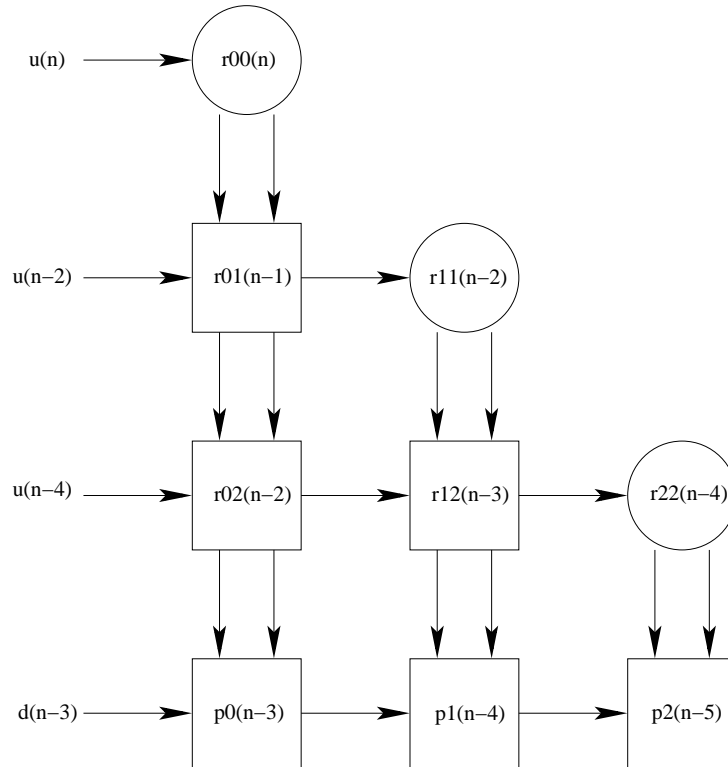


Figura 1: Parte triangular del array sistólico. Se encarga de calcular los valores de la matriz R , efectuando las rotaciones de *Givens*. También es la encargada de obtener el vector \mathbf{p}

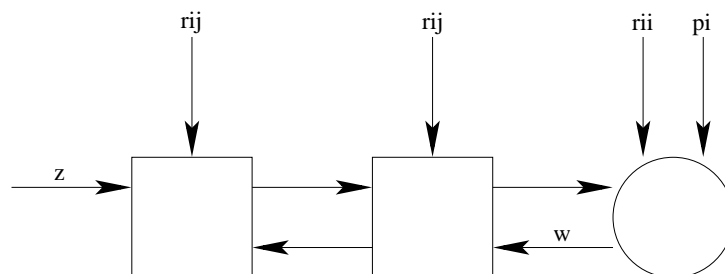


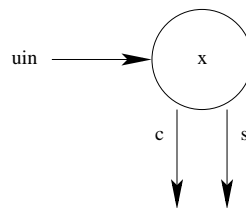
Figura 2: Parte lineal del array sistólico. Partiendo de los valores, en un instante, de la matriz R y el vector \mathbf{p} , calcular el vector de pesos \mathbf{w} mediante retro-sustitución.

toman las entradas en el mismo instante de tiempo. Ahora bien, el reloj de la parte triangular y de la parte lineal no es, en principio, el mismo. Aunque se entrará en un explicación más profunda en la parte dedicada a la sección lineal.

A continuación pasaremos a ver en más detalle cada una de estas dos partes que componen el array sistólico.

3.1. Sección triangular

En la figura 1 puede verse la sección triangular del array sistólico, formada por dos tipos de celdas: las de la frontera y las del interior. El comportamiento de cada uno de estos tipos puede verse en las figuras 3 y 4. El parámetro λ es el mismo que el del algoritmo RLS. Se emplea para proporcionar más o menos importancia a la memoria del RLS, lo que se reflejará en la velocidad de convergencia y en las prestaciones de ruido.



Si $u_{in} = 0$ entonces:

$$\begin{aligned} c &\leftarrow 1 \\ s &\leftarrow 0 \\ x &\leftarrow \lambda^{1/2}x \end{aligned}$$

en caso contrario:

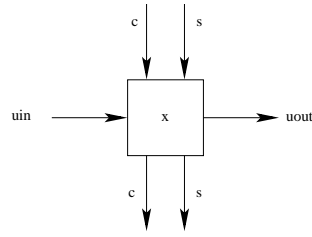
$$\begin{aligned} x' &\leftarrow \sqrt{\lambda x^2 + |u_{in}|} \\ c &\leftarrow \frac{\lambda^{1/2}x}{x'} \\ s &\leftarrow \frac{u_{in}}{x'} \\ x &\leftarrow x' \end{aligned}$$

Figura 3: Funcionamiento de las celdas de frontera de la zona triangular

La inicialización de las celdas se realiza poniendo todos sus valores a cero, excepto la variable c , que se fija a 1. De esta forma tenemos que para cualquier instante de tiempo inferior a 0:

$$\begin{aligned} x &= 0 \\ c &= 1 \\ s &= 0 \end{aligned}$$

Esta parte del array es la encargada de calcular la matriz R , que al emplear la descomposición QR es diagonal superior. Según van entrando los nuevos elementos se van actualizando



$$\begin{aligned} u_{out} &\leftarrow cu_{in} - s^* \lambda^{1/2} x \\ x &\leftarrow su_{in} + c \lambda^{1/2} x \end{aligned}$$

Figura 4: Funcionamiento de las celdas interiores de la zona triangular

las posiciones de la matriz R en diferentes instantes, empleando las rotaciones de Givens para mantener la forma triangular de la matriz. Son necesarios $2N - 1$ ciclos de reloj para terminar de calcular todos los elementos de la matriz en un instante determinado, donde N es el tamaño del vector w . Esto es así porque es necesario que la información que entra por la celda de frontera más alta llegué hasta la celda frontera más baja.

Por otra parte, en la última fila de celdas se calcula el valor del vector p . Para completar su cálculo son necesarios $2N$ ciclos de reloj. Una vez que están todos estos datos almacenados en memoria, transcurridos los ciclos de reloj necesarios, puede calcularse el vector de pesos resultantes sin más que extraer la matriz R y el vector p , para un instante determinado, y realizar, como ya se ha visto, la siguiente operación:

$$w = R^{-1}p \quad (3)$$

Los contenidos de las celdas en cada instante de tiempo puede verse en el cuadro 1.

Entradas	Contenidos		
$u(n)$	$r_{00}(n)$		
$u(n-2)$	$r_{01}(n-1)$	$r_{11}(n-2)$	
$u(n-4)$	$r_{02}(n-2)$	$r_{12}(n-3)$	$r_{22}(n-4)$
$d(n-3)$	$p_0(n-3)$	$p_1(n-4)$	$p_2(n-5)$

 Cuadro 1: Contenido de las celdas en el instante n

3.2. Sección lineal

Para no tener que realizar la inversión de la matriz de la expresión (3) puede implementarse un sistema de retro-sustitución en el array sistólico. Esto puede hacer porque la matriz R es siempre triangular. Para realizar esta retro-sustitución empleamos la segunda parte del array sistólico, la que hemos llamado sección lineal. En ella vamos calculando el valor de cada uno de los pesos, empezando por el último. De la celda de frontera de esta zona sale un peso cada 2 ciclos de reloj, lo que revela, sin lugar a dudas, la necesidad de emplear un reloj diferente en esta sección que en el resto del array. El tiempo necesario para obtener el vector w completo serán $3N$ ciclos de reloj, contando el tiempo necesario para recorrer todas las celdas de esta

sección. Otro punto que se debe tener en cuenta es como hacer llegar los datos desde las celdas de la parte triangular, con la información acerca de los diferentes valores de las componentes de R y p en los instantes adecuados. Este punto se tratará de manera más extensa en la sección 4. Por ahora supondremos que los valores necesarios llegan hasta las celdas de manera correcta. El funcionamiento de las celdas puede verse en las figuras 5 y 6. Para despejar correctamente \hat{w} será necesario hacer llegar en los instantes adecuados r_{ii} y p_i a la celda de la frontera y r_{ij} a las celdas del interior, siguiendo la secuencia del cuadro 2. En el mismo también puede verse el orden y los instantes de salida de las componentes del vector de pesos \hat{w} .

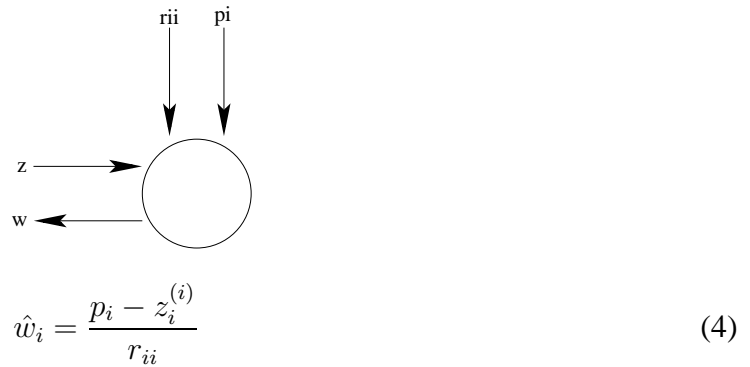


Figura 5: Funcionamiento de la celda de frontera la zona lineal

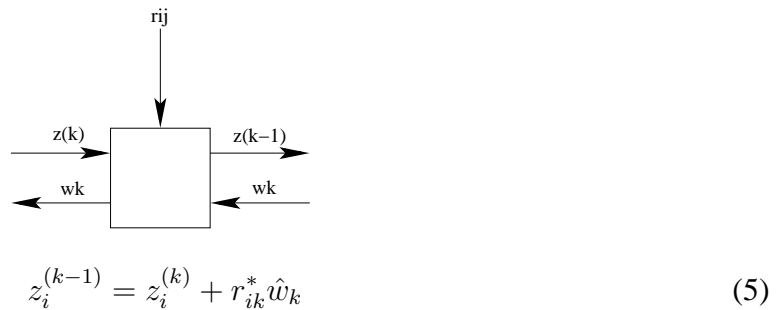


Figura 6: Funcionamiento de las celdas interiores de la zona lineal

4. Implementación

La implementación de las dos partes de los array sistólicos resulta trivial, el problema aparece con la necesidad de hacer llegar los datos desde una parte a la otra. Para ello podemos pensar en una serie de diferentes posibilidades:

- En primer lugar podemos almacenar todos los valores que van generando las celdas de la parte triangular. Con estos y haciendo que el reloj en la sección lineal funcione a la velocidad adecuada pueden conseguirse obtener el vector de pesos actualizado constantemente. Los resultados serán similares a los obtenidos con el RLS en su formulación

n	Frontera		Interior	
0				
1				
2		r_{22} y p_2		
3	\hat{w}_2		r_{12}	
4		r_{11} y p_1		r_{02}
5	\hat{w}_1		r_{01}	
6		r_{00} y p_0		
8	\hat{w}_0			

Cuadro 2: Orden de entrada y de salida de los datos en las celdas de la sección lineal. Las columnas representan las diferentes celdas, desde la frontera (primera y segunda columnas, para entrada y salida respectivamente) a la última de las celdas interiores (última columna)

tradicional con la ventaja de que proporciona un mejor aprovechamiento de los recursos por su codificación paralela. El problema principal de esta aproximación es la importante cantidad de espacio de almacenamiento necesario para implementar este sistema así como la necesidad de llevar un sistema que lo controle, incrementado a su vez la dificultad global del array e introduciendo un controlador centralizado. La memoria además deberá almacenar el contenido de R y de p para tanto tiempo como tarde el array en calcular todos sus valores. Por lo tanto, dado que son necesarios $2N - 1$ ciclos de reloj para obtener hasta la última componente del vector p en un instante, habrá que mantener una memoria de $\frac{(2N-1)}{2}(N^2 + N)$ posiciones.

- Otra posible opción es no calcular el vector de pesos todos los instantes sino realizar un muestro. De esta forma solo será necesario almacenar los valores de la matriz R y el vector p en un instante de tiempo, con la consiguiente reducción de coste que esto implica, y la posibilidad de que la sección lineal podrá trabajar a la misma velocidad que la del resto del array. Para optimizar se pueden colocar los valores de una pila en la que se pueden introducir de manera inversa a la que se van creando, según el cuadro 1. La extracción de los datos se realizará desde la parte de arriba de la pila para poder colocarlos en las celdas de la parte lineal en el orden que se muestra en el cuadro 2.

5. Resultados

Una vez implementado el array sistólico con el QRD-RLS los resultados obtenidos deberán ser similares a los que se consiguen con el resto de las implementaciones del RLS. En las figuras 7 y 8 pueden verse la evolución de los pesos reales de un sistema, como el array sistólico los sigue y que error comete, respectivamente. En este caso se ha empleado una $\lambda = 0,94$ y se actualiza el vector de pesos en cada instante de tiempo.

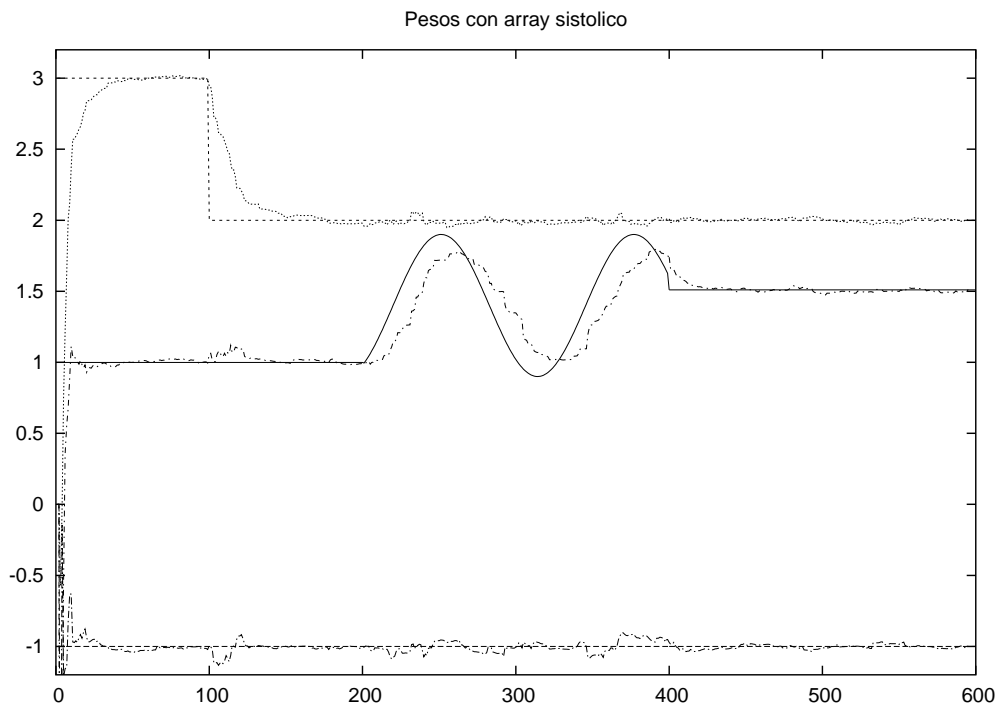


Figura 7: Pesos reales y pesos calculados con el array sistólico con $\lambda = 0,94$

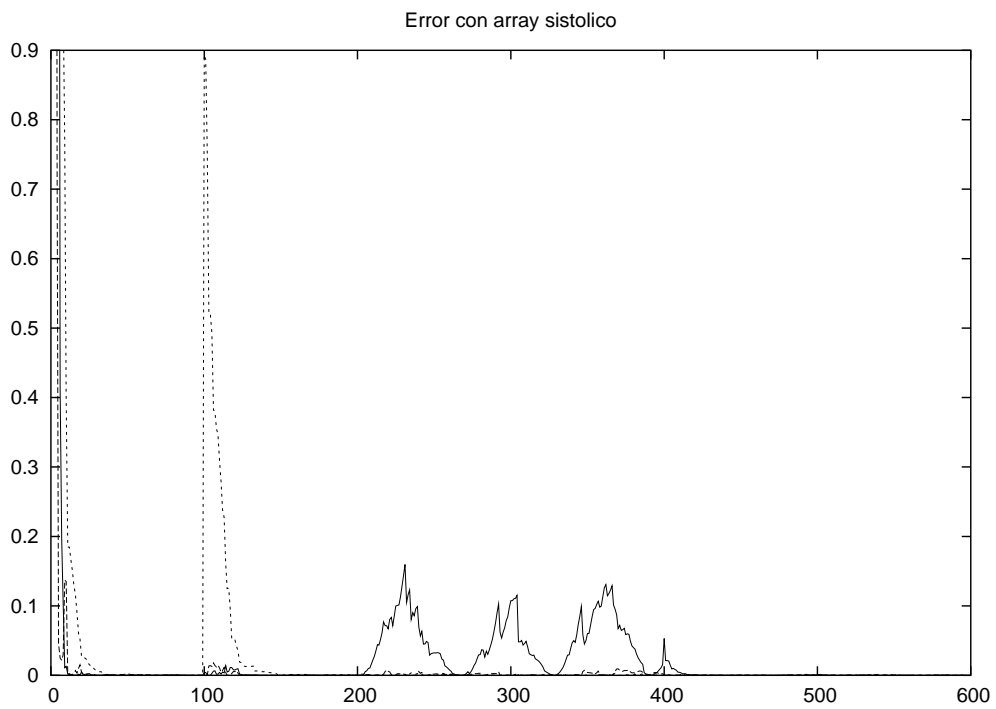


Figura 8: Error en los pesos cometido por el array sistólico

6. Conclusiones

En el presente trabajo se ha estudiado la correcta implementación de un RLS en un array sistólico, encontrando particularidades tales como el funcionamiento separado de las dos partes, incluso con un reloj diferente, y la necesidad de encontrar una forma correcta de almacenar y gestionar el contenido de las celdas. A su vez se ha conseguido realizar una implementación software del array y se ha comprobado su correcto funcionamiento.

Un dato de interés es la posibilidad de muestrear la salida del array sistólico, simplificando así los cálculos necesarios y la complejidad de la implementación. Con una estructura de tipo pila implementada para almacenar los contenidos de las celdas en determinados instantes se puede conseguir una implementación del RLS en array sistólico compacta y eficiente.