Diseño de la DPD adaptable: flujo de trabajo de arriba abajo

Artículo cedido por Mathworks

www.mathworks.com

Autor: Kerry Schutz, MathWorks

La predistorsión digital (DPD) es una técnica de procesamiento de señales de banda base que corrige las deficiencias en amplificadores de potencia (PA) RF. Estas deficiencias provocan emisiones fuera de banda o recrecimiento espectral y distorsión en banda, lo que se correlaciona con una mayor tasa de errores de bit (BER).

Las señales de banda ancha con una alta relación de pico a promedio, como las existentes en transmisores LTE/4G, son especialmente susceptibles a estos efectos no deseados.

Una propuesta de modelado y simulación de arriba a abajo le permite identificar y corregir rápidamente estos problemas. En este artículo se presenta un flujo de trabajo de arriba a abajo para el modelado y la simulación de PA y DPD mediante MATLAB, Simulink, Signal Processing Toolbox, Control System Toolbox y DSP System Toolbox. Partiendo de mediciones de PA, derivamos un diseño estático de DPD basado en un polinomio de memoria, que corrige tanto las no linealidades como los efectos de memoria en los PA.

Construimos un modelo de nivel de sistema para evaluar la eficacia de la DPD. Dado que las características de los PA varían con el tiempo y las condiciones de funcionamiento, transformamos el diseño estático de la DPD en uno adaptable. Evaluamos dos diseños adaptables de DPD, uno basado en el algoritmo Least Mean Square (LMS, cuadrado de valor promedio mínimo) y un segundo basado en el algoritmo Recursive Predictor Error Method (RPEM, método recursivo de error de predicción).

Los modelos utilizados en este artículo están disponibles para su descarga.

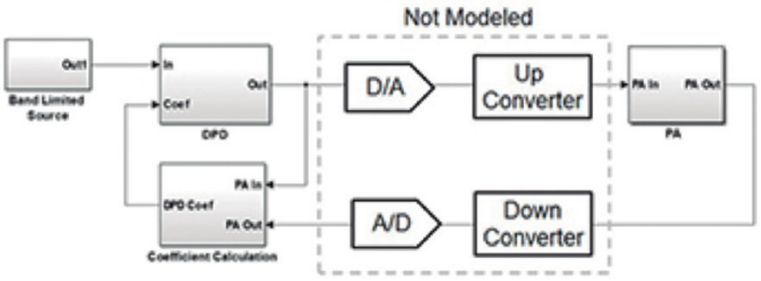


Figura 1. El modelo de nivel de sistema simplificado. La cuantización y los efectos de conversión arriba/abajo se sustituyen por transferencia.

Descripción del flujo de trabajo

Nuestro objetivo es desarrollar un modelo de simulación que represente con precisión las deficiencias de los PA, y un diseño adaptable de DPD que mitigue dichas deficiencias1. Dividimos el esfuerzo de modelado en cinco etapas:

1. Modelado y simulación del PA
2. Derivación de coeficientes de DPD
3. Evaluación del diseño estático de DPD
4. Conversión del diseño estático de DPD en uno adaptable
5. Evaluación de las variantes de LMS y RPEM

Para acelerar las simulaciones, realizaremos las siguientes simplificaciones (Figura 1):

- Modelado del PA como un sistema de tiempo discreto (en la práctica, el PA es un circuito analógico)
- Modelado de la señal de PA como un complejo de banda base (en la práctica, la señal de PA es una señal de banda de paso real)
- Utilización de tipos de datos de doble precisión y matemáticas (en la práctica, se utilizarían tipos de datos enteros y matemáticas)
- Omisión de los efectos de cuantización provocados por el ADC y DAC

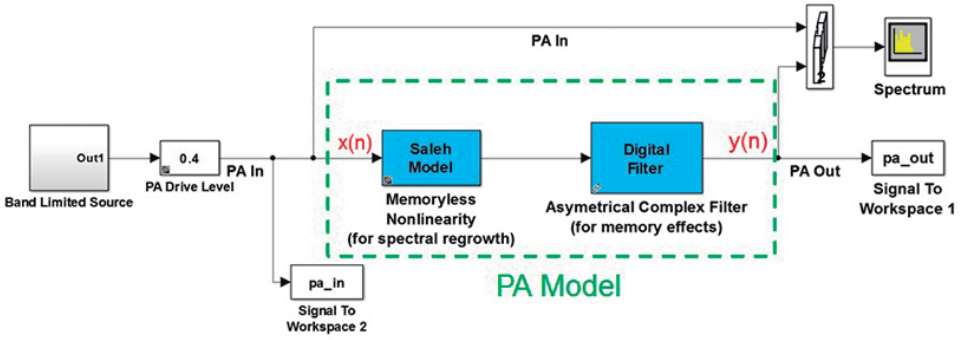


Figura 2. El modelo de PA.

Modelado y simulación del PA

El modelo de PA consta de un amplificador Saleh [3] en serie con un filtro complejo asimétrico (Figura 2). La excitación es una señal de ruido blanco aditivo gaussiano (AWGN) en la que se han utilizado filtros elípticos de paso bajo.

Ejecutamos el modelo y registramos las señales de entrada y salida, x e y, respectivamente, mientras supervisamos sus espectros (Figura 3).

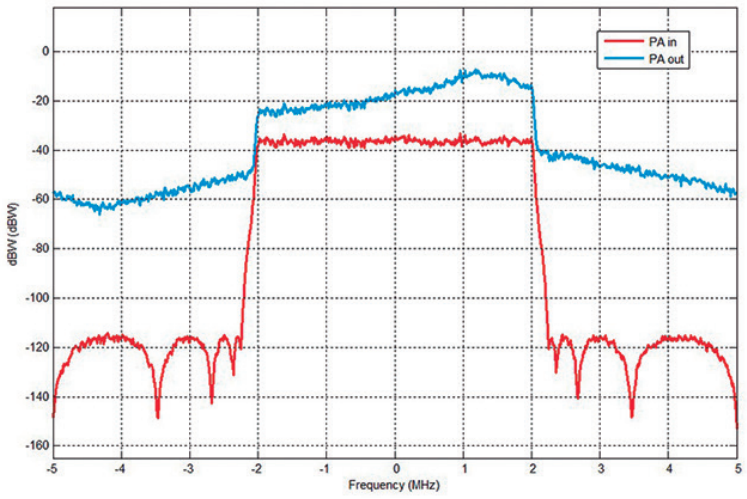


Figura 3. La salida del PA mostrando 20 dB de ganancia a costa del significativo recrecimiento espectral y de la distorsión en banda.

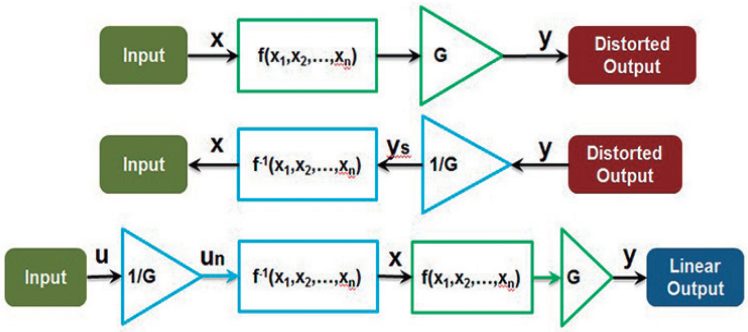


Figura 4. Diagramas de bloque que ilustran la derivación de la DPD y el proceso de implementación.

Derivación de los coeficientes de DPD

El diseño estático de DPD se deriva de las mediciones de PA (Figura 4). La ruta superior en la figura 4 representa el modelo de PA.

El PA se divide en una función no lineal seguida de una ganancia G lineal. La ruta intermedia muestra el PA funcionando en reversa. Esta ruta representa la DPD. No podemos ejecutar un PA en reversa físicamente, pero sí matemáticamente, y esta es la clave para la derivación de la DPD. En reversa, aplicamos la operación no lineal inversa, f-1(x1,x2,...xn). La ruta inferior en la figura 4 es la cascada de las dos rutas superiores, esto es, la de la DPD y el PA. La derivación de la DPD puede ahora continuar del siguiente modo:

1. Supongamos un formato de polinomio de memoria [1] para el operador no lineal del PA, f(x1,x2,...xn).
   donde:
   - x es la entrada del PA
   - y es la salida del PA
   - akm son los coeficientes polinómicos del PA
   - M, profundidad de memoria del PA
   - K, grado de no linealidad del PA
   - n, índice de tiempo

   El valor x de entrada, el valor y de salida, y los coeficientes akm son complejos.
2. Invirtiendo las funciones de x e y en 1 para modelar la función no lineal inversa de la DPD, f-1(x1,x2,...xn).

   La ganancia G lineal ha normalizado y(n), y se puede haber desplazado en el tiempo (opcionalmente).

   ys(n) = y(n)/G

   yss(n) = ys(n+desplazamiento)

   donde desplazamiento es un entero positivo fijo.

   Alinear la sincronización es vital. Si el PA tiene unos requisitos de memoria significativos, podríamos necesitar desplazar y en el tiempo antes de derivar los coeficientes. La DPD debe representar correctamente la demora positiva del PA. Ninguna estructura viable puede tener una demora negativa, pero usted puede derivar los coeficientes basándose en un desplazamiento temporal en la salida de PA.

   Y, recuerde, la salida de PA es una entrada en la derivación de coeficiente de la DPD. Esto significa que puede hacer que el valor x de salida responda antes a las muestras y «desplazamiento» de entrada basando la derivación de coeficientes en una nueva secuencia, yss(n) = ys(n+desplazamiento).

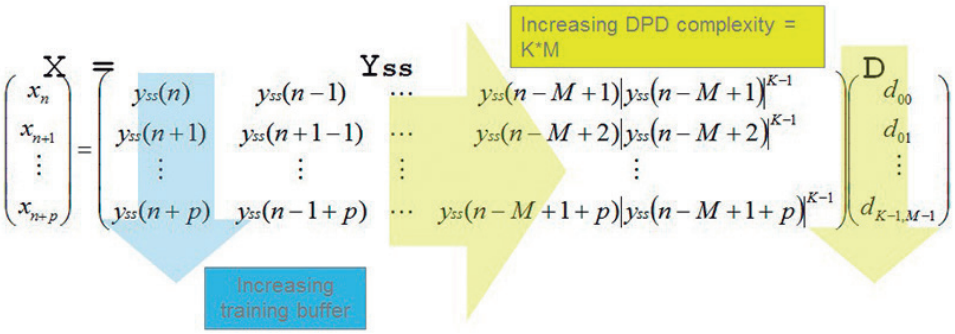


Figura 5. Ecuación 2 reorganizada como un sistema de ecuaciones lineales en formato matricial. La variable p denota el número de muestras de medición. El valor de p suele ser con mucha frecuencia mucho mayor que el producto K*M. Así estamos ante un sistema sobredeterminado.

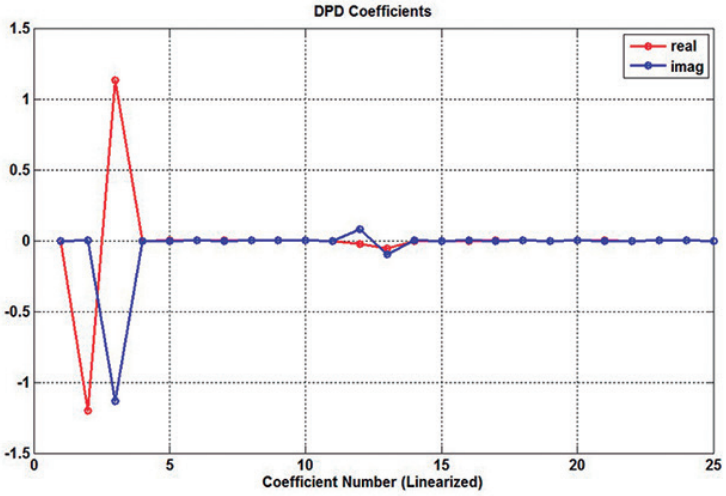


Figura 6. Los componentes reales e imaginarios de los coeficientes complejos de la DPD derivada.

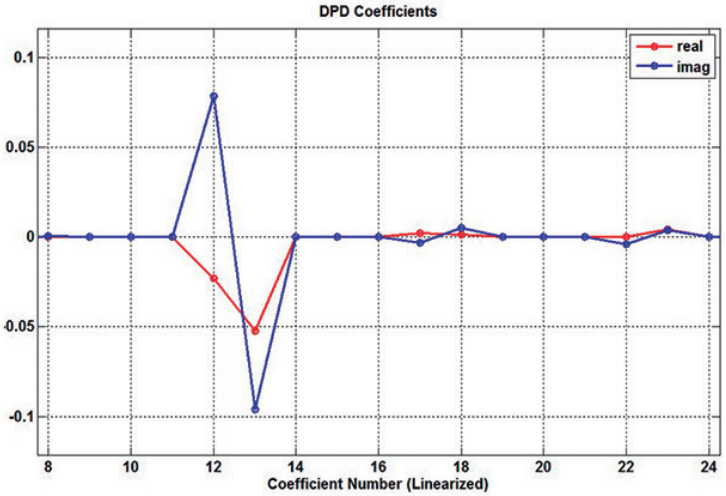


Figura 7. Aquí ampliamos los coeficientes 8 a 24. Estos coeficientes, aunque relativamente pequeños, resultan vitales para reducir el recrecimiento espectral.

Equation 2 in symbolic form	$x_{MP}(n) = \sum_{k=0}^{K-1}\sum_{m=0}^{M-1} d_{km}u(n-m)\lvert u(n-m)\rvert^k$

Equation 2 in MATLAB code	`function x=DPD(u,coef,K,M)`
	`persistent pipe`
	`if isempty(pipe)`
	`  pipe=complex(zeros(M,1));`
	`end`
	`pipe(1:end-1)=pipe(2:end);`
	`pipe(end) = u;`
	`y = complex(0,0);`
	`for k=1:K`
	`  for m=1:M`
	`  y=y+coef((k-1)*M+m)*pipe(m)*abs(pipe(m))^(k-1);`
	`  end`
	`end`

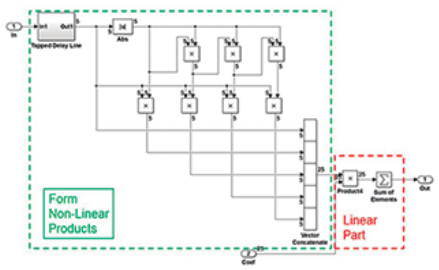
Equation 2 in Simulink block diagram form	

Tabla 1. Implementaciones de MATLAB y Simulink de la ecuación 2.

En nuestro ejemplo, utilizamos un desplazamiento de tres muestras.

3. Resolución de los coeficientes de la DPD, dkm. Reescribimos la ecuación 2 como un conjunto de sistemas de ecuaciones lineales (Figura 5). Al resolver dkm, se puede resolver un sistema sobredeterminado de ecuaciones lineales, ecuación 4.

Las mediciones de x e y son valores conocidos. Se elige cierto valor para K y M según la complejidad del PA. Elegimos M = K = 5 y así resolvemos los coeficientes complejos K*M = 25.

Utilizamos el operador de barra inversa de MATLAB para resolver este sistema sobredeterminado de ecuaciones para los coeficientes de DPD dkm y los resultados se muestran en las figuras 6 y 7.

Evaluación del diseño de DPD de coeficiente fijo

Para evaluar el diseño, hemos creado un modelo de sistema de la DPD y del PA (Figura 8).

La primera tarea es la implementación de la ecuación 2. Representar estas ecuaciones en MATLAB es algo directo, tal y como se muestra en la tabla 1.

Los resultados de simular el modelo de verificación para K = M = 5 se muestran en la figura 9.

Conversión en adaptable

Aunque nuestro diseño de DPD promete, no se presta del todo bien a una implementación adaptable. Además, el proceso matemático de inversión de matriz y los grandes bufferes necesarios para resolver un sistema sobredeterminado de ecuaciones no son viables para la implementación de hardware.

Utilizamos una arquitectura de aprendizaje indirecto [2] para implementar una DPD adaptable (Figura 10). El diseño consta de un subsistema de cálculo de coeficientes y de un subsistema de DPD. Se trata de una implementación en streaming sin inversión de matriz.

El subsistema de la DPD es idéntico en forma al mostrado en la tabla 1. Aquí nos centramos en el

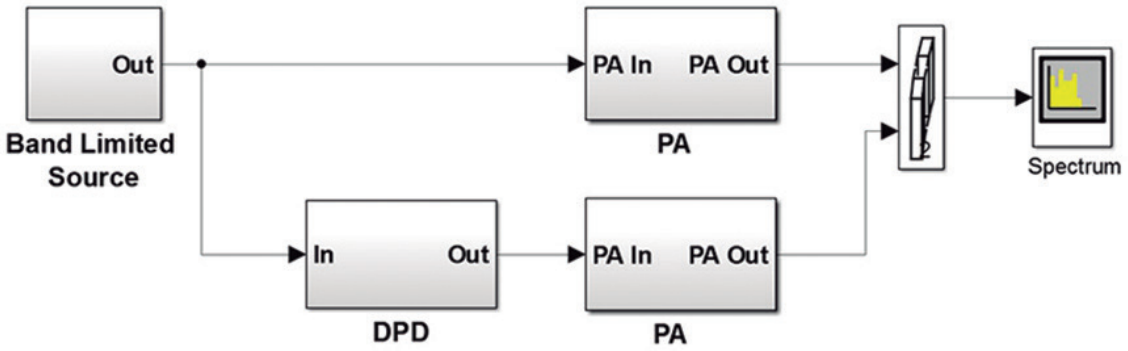


Figura 8. Modelo de verificación de DPD.

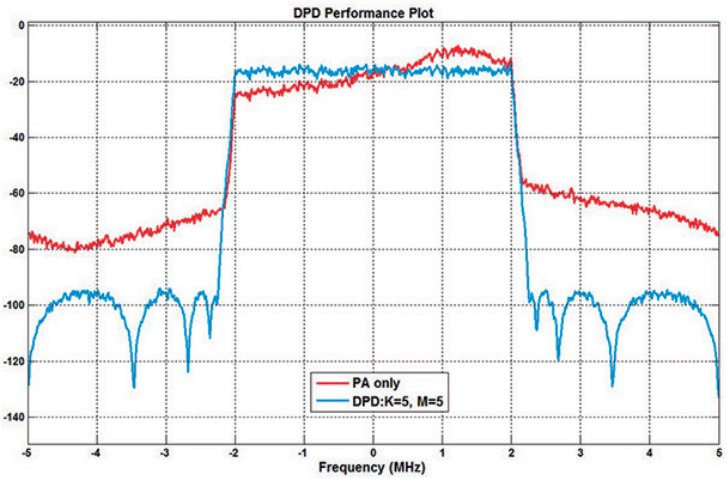


Figura 9. Gráfico que compara el rendimiento de la DPD para K = M = 5 contra el PA por sí solo.

subsistema de cálculo de coeficientes (Figura 11).

La arquitectura adaptable de la DPD tiene dos copias del algoritmo de DPD, una para aprender los coeficientes y otra para implementarlos. La combinación de los subsistemas NonLinear_Prod (Figura 12) y Coef_Compute implementa la copia de aprendizaje del subsistema de la DPD.

Los coeficientes de la DPD, dkm, se calculan mediante el algoritmo LMS (Figura 13) o mediante el algoritmo RPEM (Figura 14). El algoritmo LMS es relativamente simple en términos de recursos necesarios. El cálculo de coeficientes basado en RPEM es mucho más complejo que el LMS2. Tanto el algoritmo

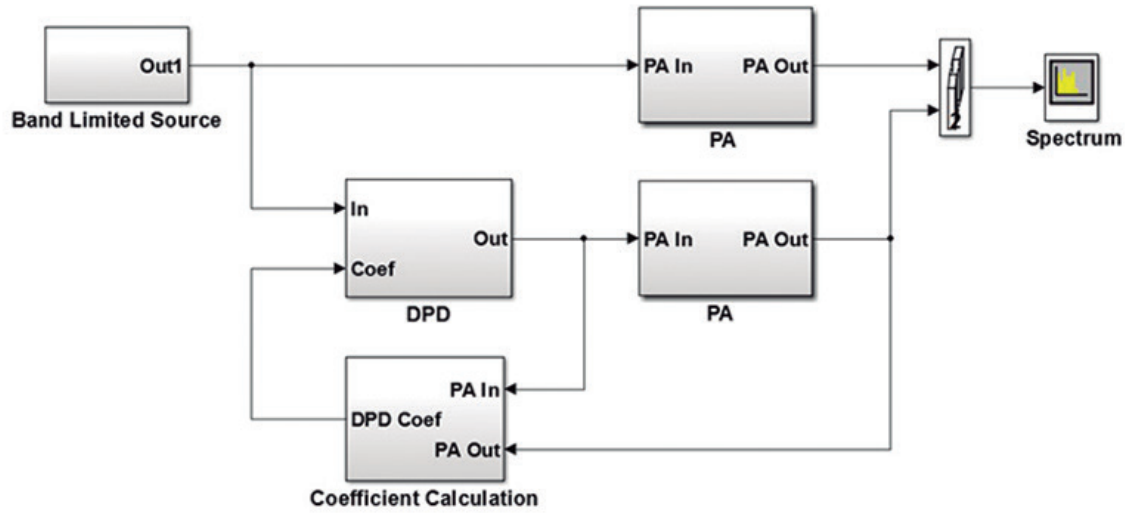


Figura 10. El banco de pruebas de la DPD adaptable. El subsistema de cálculo de coeficientes toma muestras de la entrada y la salida del PA y realiza cálculos matriciales para derivar un conjunto de coeficientes de la DPD. El subsistema de la DPD aplica estos coeficientes a un polinomio de memoria y genera una forma de onda predistorsionada.

Figura 11. Visualización de alto nivel del subsistema de cálculo de coeficientes.

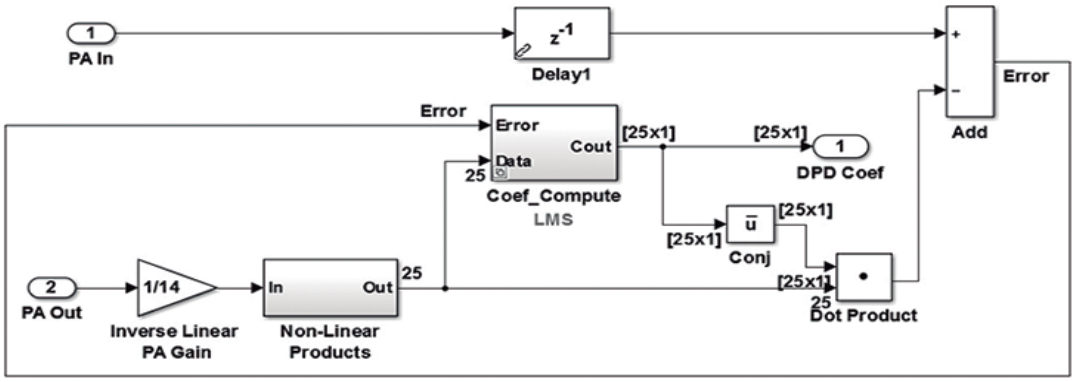
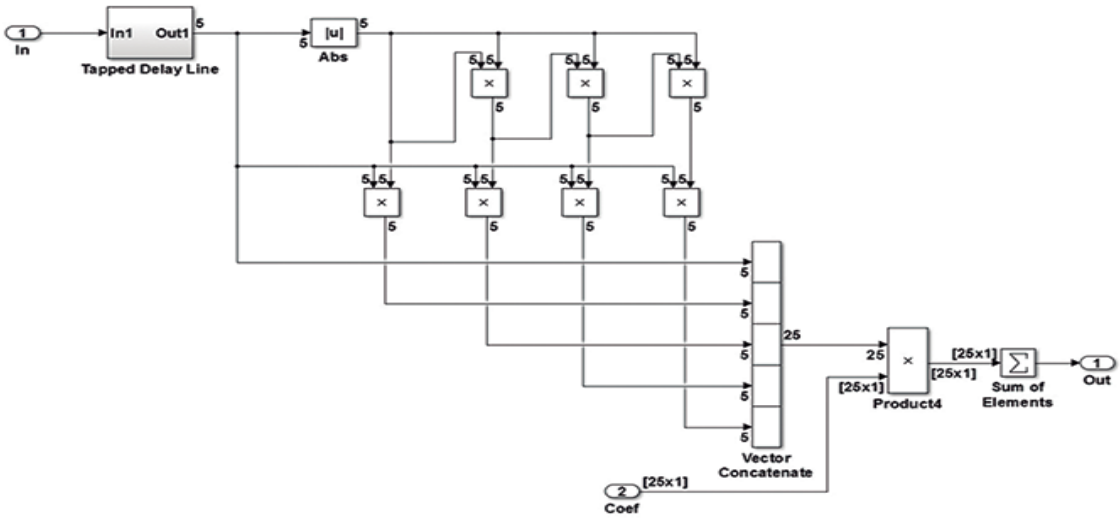


Figura 12. El subsistema NonLinear_Prod, que implementa los múltiplos no lineales en la ecuación de la DPD, ecuación 2.



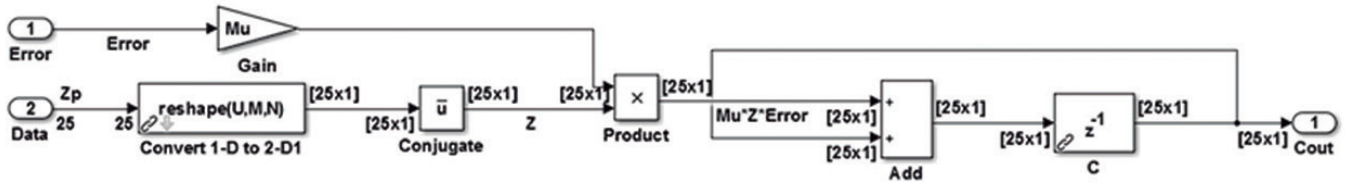


Figura 13. El cálculo de coeficientes basado en LMS.

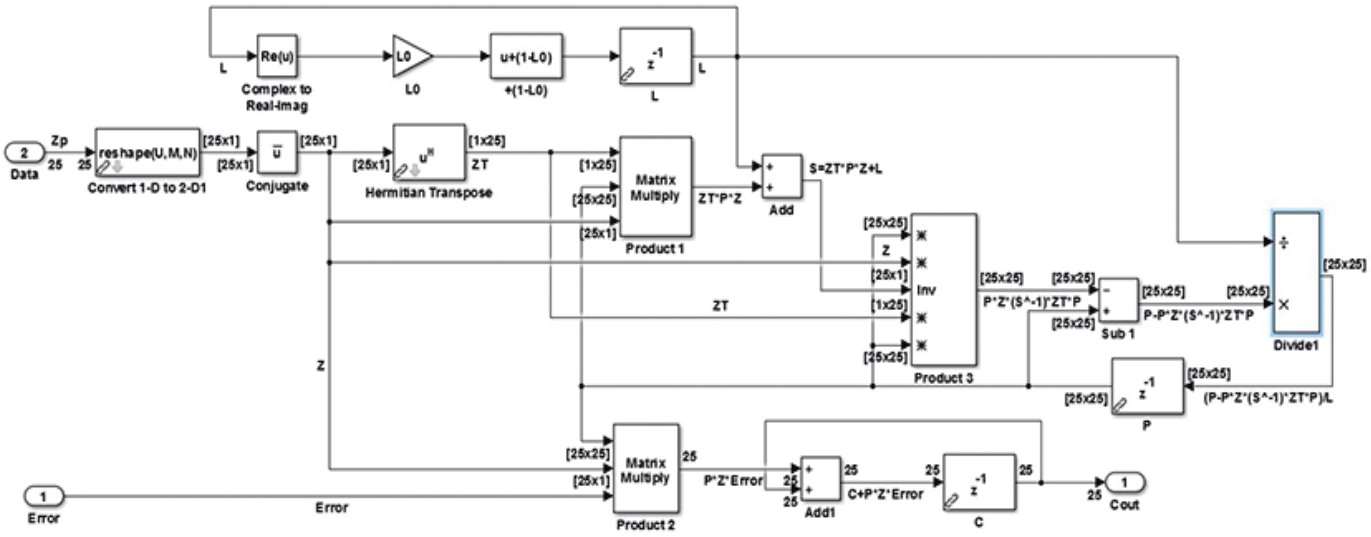


Figura 14. El cálculo de coeficientes basado en RPEM.

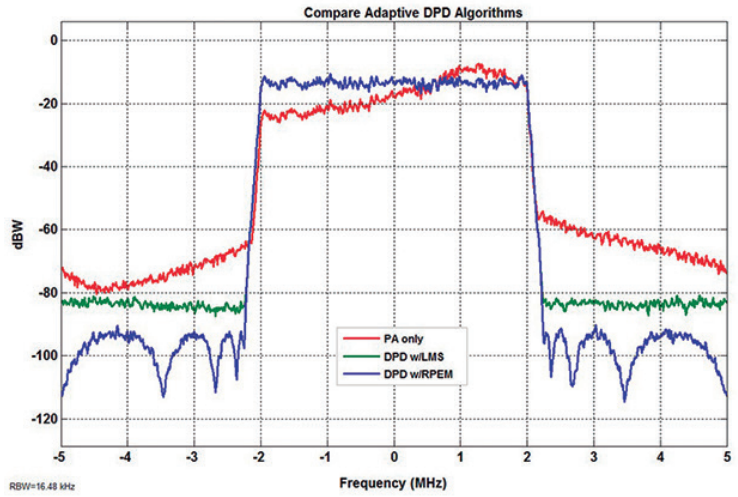


Figura 15. Gráfico que compara la eficacia de los algoritmos LMS y RPEM en estado estable en términos de reducción de recrecimiento espectral.

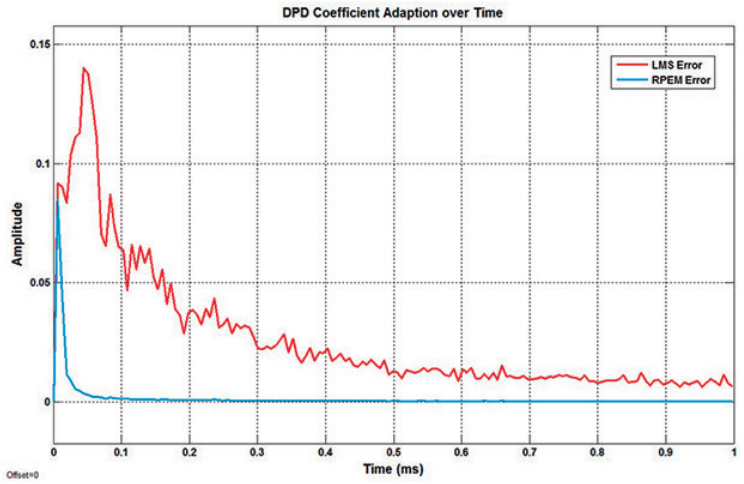


Figura 16. Gráfico que compara la tasa de convergencia de coeficientes para los algoritmos LMS y RPEM.

LMS como el RPEM calculan una señal de error en un bucle de realimentación. El error es la diferencia entre la entrada de PA medida y la entrada de PA estimada. El algoritmo trata de llevar este error a cero y al hacerlo converge en una mejor estimación de los coeficientes de la DPD. Comparamos el rendimiento de los dos métodos (figuras 15 y 16).

El método RPEM proporciona resultados significativamente mejores en términos tanto de reducción de crecimiento espectral, como de tasa de convergencia. El algoritmo RPEM converge en el mismo conjunto de coeficientes que los calculados a partir de mediciones de PA sin conexión, mientras que el algoritmo LMS nunca llega a converger en esta solución ideal. El algoritmo RPEM, sin embargo, presenta menos desventajas. El algoritmo RPEM que utiliza M = K = 5 requiere aproximadamente 75.300 múltiplos por actualización. El algoritmo LMS para los mismos M y K requiere aproximadamente 100 múltiplos por actualización. El algoritmo RPEM es así 753 veces más costoso en términos de cálculo. En su forma actual, el algoritmo RPEM no se ajusta bien a la implementación de hardware. Por otro lado, la propuesta basada en LMS se ajusta mucho mejor a la implementación de hardware, pero carece relativamente de reducción de recrecimiento espectral.

Resumen

En este artículo se demuestra un flujo de trabajo para el modelado y la simulación de PA y DPD. Mostramos el progreso del proceso de diseño de la DPD a partir de una derivación de procesamiento por lotes sin conexión, que implica cálculos de inversión de matriz, hasta una implementación adaptable totalmente en streaming, que no implica ninguna inversión de matriz. Utilizamos tres factores para evaluar la eficacia del diseño de la DPD adaptable: reducción del recrecimiento espectral, tasa de convergencia y complejidad de cálculo.

Comparamos dos variantes de la arquitectura adaptable de aprendizaje indirecto, una basada en el algoritmo LMS y la segunda basada en el algoritmo RPEM. El algoritmo RPEM se muestra superior en términos de reducción de recrecimiento espectral y tasa de convergencia, pero prohibitivo en términos de coste de cálculo.

Utilizamos una combinación de MATLAB y Simulink para este proyecto. MATLAB se utiliza para definir parámetros de sistema, probar algoritmos individuales y realizar cálculos sin conexión. Simulink se utiliza para integrar los componentes de PA y DPD individuales en un marco de pruebas donde los bucles de realimentación, los tamaños de los datos y las divisiones de diseño son fácilmente discernibles. Ω

REFERENCIAS

- *Morgan, Ma, Kim, Zierdt y Pastalan. "A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers." IEEE Trans. on Signal Processing, Vol. 54, No. 10, octubre 2006.*
- *Li Gan y Emad Abd-Elrady. "Digital Predistortion of Memory Polynomial Systems Using Direct and Indirect Learning Architectures." Institute of Signal Processing and Speech Communication, Universidad de Tecnología de Graz.*
- *Saleh, A.A.M., "Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers." IEEE Trans. Communications, vol. COM-29, pp.1715-1720, noviembre de 1981.*