DISEÑO DE CIRCUITOS DE ALTA FRECUENCIA USANDO MAPEO ESPACIAL NEURAL CON NO-LINEALIDAD REGULADA

Vladimir Gutiérrez-Ayala y José Ernesto Rayas-Sánchez

Grupo de Investigación en Ingeniería Asistida por Computadora de Circuitos y Sistemas
Departamento de Electrónica, Sistemas e Informática
Instituto Tecnológico y de Estudios Superiores de Occidente (ITESO)
Tlaquepaque, Jalisco, México, 45090

E-mail: vladimirga@iteso.mx y erayas@iteso.mx

ABSTRACT

En este trabajo se realizan mejoras sustanciales al algoritmo de diseño de circuitos electrónicos basado en mapeo espacial neural. Dichas mejoras incluyen la regulación de la cantidad de no-linealidad utilizada por la red neuronal durante el entrenamiento del neuromodelo mapeado, así como una simplificación del proceso de obtención del siguiente punto predicho por el algoritmo, y la determinación automática de criterios de finalización del mismo. Con las mejoras implementadas se obtiene un algoritmo de diseño más eficiente y robusto. Para ilustrar el desempeño de este nuevo algoritmo se diseñan dos filtros en tecnología microcinta: un filtro rechaza-banda con "stubs" abiertos resonantes de un cuarto de longitud de onda, y un filtro notch de alta selectividad. Para ambos circuitos se utilizan simuladores electromagnéticos de onda completa.

1. INTRODUCCIÓN

El modelado y diseño de circuitos electrónicos mediante redes neuronales artificiales (RNA) es una área que ha experimentado un gran crecimiento en los últimos años [1]-[3]. El método convencional para diseñar circuitos mediante RNAs básicamente consiste de dos pasos: a) primero se desarrolla un neuromodelo del circuito, es decir, se entrena una RNA para que sus respuestas aproximen a las del circuito en una cierta región de interés; b) una vez desarrollado el neuromodelo, éste se utiliza para hacer diseño por métodos clásicos de optimización.

Este trabajo fue financiado en parte por CONACYT (Consejo Nacional de Ciencia y Tecnología, Gobierno Mexicano) bajo los proyectos I39341A y C02-42930A-1. Vladimir Gutiérrez-Ayala fue apoyado mediante beca CONACYT 182533.

Una desventaja del método anterior es que generalmente se requieren muchos datos de entrenamiento para generar un neuromodelo que cubra con suficiente precisión la región de interés. Generar los datos de entrenamiento puede resultar costoso cuando provienen de mediciones de laboratorio, o bien cuando son producidos por algún simulador que consuma muchos recursos computacionales (especialmente tiempo de simulación), como sucede con los simuladores electromagnéticos de onda completa. A este tipo de modelos, de alta precisión pero alto costo computacional, se les denomina "modelos finos".

El neuromodelado basado en mapeo espacial [4] es una técnica que reduce considerablemente la cantidad requerida de datos de aprendizaje provenientes del modelo fino, en comparación con el neuromodelado convencional descrito anteriormente. Además, mejora notablemente la habilidad de generalización del neuromodelo debido a que se hace uso de un modelo burdo o empírico, el cual es una aproximación del circuito a modelar. Los modelos burdos son generalmente modelos de circuitos equivalentes, los cuales son computacionalmente muy eficientes pero tienen un rango de validez limitado para sus parámetros. De esta manera, el modelo burdo es usado como fuente de conocimiento previo que reduce la cantidad de puntos de entrenamiento.

El algoritmo de diseño de circuitos electrónicos usando mapeo espacial neural [5] explota en cada iteración la técnica de neuromodelado basado en mapeo espacial [4]. Teniendo el neuromodelo mapeado con un error de entrenamiento suficientemente pequeño, se optimizan sus parámetros de entrada en cada iteración, para así obtener el siguiente punto en el algoritmo.

En el presente trabajo se realizan mejoras al algoritmo de diseño de circuitos electrónicos usando mapeo espacial neural [5], el cual es ahora implementado en MatLab^{MR} [6], [7]. Las mejoras consisten en tres cambios principales:

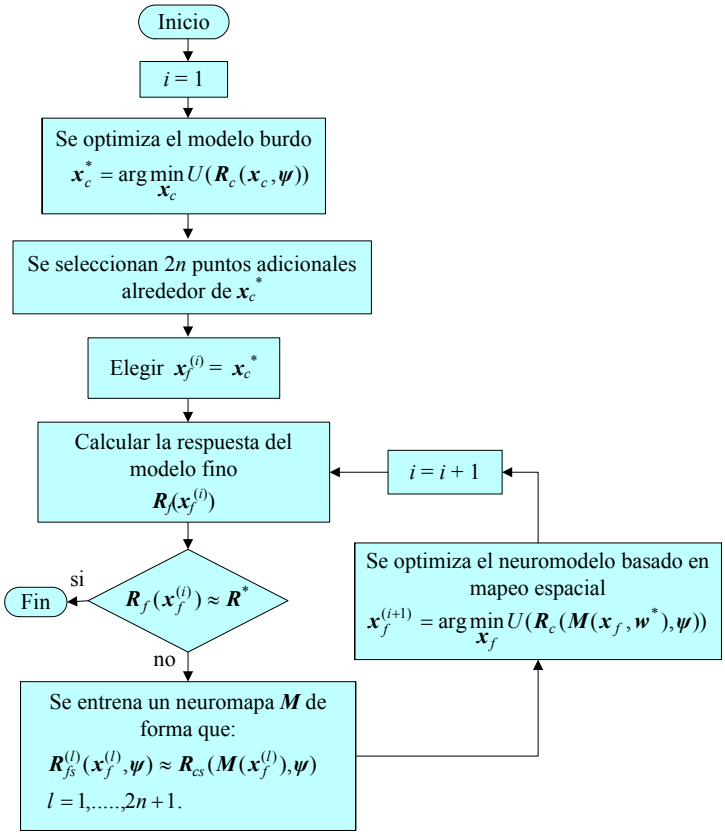


Fig. 1 Diagrama de flujo simplificado del algoritmo de diseño usando mapeo especial neural.

a) regulando la cantidad de no-linealidad del perceptrón de tres capas durante todo el entrenamiento, mejorando así la habilidad de generalización de la RNA; b) modificando la manera mediante la cual se predice el siguiente punto en el algoritmo, lo cual lo hace ahora de manera más eficiente; y (c) haciendo que el perceptrón de dos capas prediga el valor de error máximo permitido para la finalización del entrenamiento del neuromodelo mapeado, y el máximo error relativo permitido entre la respuesta deseada y la respuesta del modelo fino evaluado en el punto predicho por el algoritmo. Estos tres cambios han dado lugar a un algoritmo más robusto.

Para mostrar el desempeño del nuevo algoritmo se diseñan dos filtros de tipo microcinta: un filtro rechaza banda con "stubs" abiertos resonantes de un cuarto de longitud de onda, y un filtro notch de alta selectividad de rechazo. Para ambos circuitos se utilizan simuladores electromagnéticos de onda completa.

2. MAPEO ESPACIAL

El mapeo espacial [8] es una técnica que combina la eficiencia computacional de un modelo burdo, con la exactitud de un modelo fino. El método de mapeo espacial establece una liga matemática entre ambos modelos, dejando el trabajo intensivo al modelo burdo, mientras se mantiene la exactitud y precisión que caracteriza al modelo fino.

Los parámetros de diseño del modelo burdo y modelo fino están en $\boldsymbol{x}_c$ y $\boldsymbol{x}_f \in \Re^n$, respectivamente, mientras que $\boldsymbol{R}_c(\boldsymbol{x}_c)$ y $\boldsymbol{R}_f(\boldsymbol{x}_f)$ son sus respectivas respuestas (c por la inicial de burdo en inglés "*coarse*"). La estrategia del mapeo espacial consiste en encontrar un mapa apropiado $\boldsymbol{M}$ del espacio de entrada del modelo fino hacia el espacio de entrada del modelo burdo

$$\boldsymbol{x}_c = \boldsymbol{M}(\boldsymbol{x}_f) \tag{1}$$

de manera que,

$$\boldsymbol{R}_c(\boldsymbol{M}(\boldsymbol{x}_f)) \approx \boldsymbol{R}_f(\boldsymbol{x}_f) \tag{2}$$

Una vez que el mapa $\boldsymbol{M}$ es encontrado en la región de interés, el modelo burdo mapeado, $\boldsymbol{R}_c(\boldsymbol{M}(\boldsymbol{x}_f))$, puede ser usado para simulaciones eficientes y de alta precisión en esa región.

3. DISEÑO BASADO EN MAPEO ESPACIAL NEURAL

3.1. Descripción del Algoritmo

En la Fig. 1 se muestra el diagrama de flujo simplificado del algoritmo de diseño usando mapeo espacial neural [5]. Se inicia optimizando el modelo burdo para encontrar los parámetros óptimos de diseño $\boldsymbol{x}_c^*$ que hacen que su

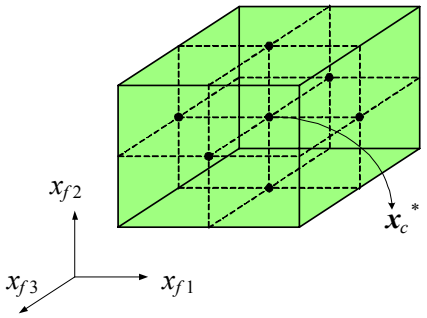


Fig. 2 Distribución de puntos en forma de estrella para un problema con tres parámetros de diseño.

respuesta cumpla con las especificaciones de diseño. Como se muestra en la Fig. 2, se seleccionan $2n$ puntos adicionales siguiendo una distribución de estrella, como en [4] y [9], centrados en los parámetros óptimos del modelo burdo $\boldsymbol{x}_c^*$, donde n es el número de parámetros de diseño. El porcentaje de desviación desde $\boldsymbol{x}_c^*$ para cada uno de los parámetros de diseño se puede determinar de acuerdo a las sensibilidades del modelo burdo, asumiendo que éstas son similares a las del modelo fino. En este trabajo, éste porcentaje de desviación se determina arbitrariamente.

En seguida se evalúa el modelo fino en la solución óptima del modelo burdo $\boldsymbol{x}_c^*$, y se obtiene su respuesta $\boldsymbol{R}_f$. Si $\boldsymbol{R}_f$ es aproximadamente igual a la respuesta deseada $\boldsymbol{R}^* = \boldsymbol{R}_c(\boldsymbol{x}_c^*, \boldsymbol{\psi})$, el algoritmo termina, de no ser así, se desarrolla un neuromodelo basado en mapeo espacial (ver Fig. 3), sobre los $2n+1$ puntos iniciales del modelo fino.

Una vez que se obtiene el neuromodelo mapeado con un error de entrenamiento lo suficientemente pequeño, se utiliza a éste como un modelo burdo mejorado optimizando sus parámetros de diseño $\boldsymbol{x}_f$, para obtener la respuesta deseada. La solución a este problema de optimización se convierte en el siguiente punto del algoritmo, el cual se incluye en el conjunto de puntos de entrenamiento del modelo fino para la siguiente iteración.

Con el punto predicho, se evalúa nuevamente el modelo fino $\boldsymbol{R}_f$ y se compara su respuesta con la respuesta deseada $\boldsymbol{R}^*$. Si siguen siendo aún muy diferentes, se vuelve a entrenar el neuromodelo basado en mapeo espacial sobre el conjunto extendido de puntos de aprendizaje, y se continúa con el algoritmo. Si las respuestas son lo suficientemente parecidas, el algoritmo termina.

3.2. Optimización del Modelo Burdo

El primer paso del algoritmo es encontrar los parámetros óptimos de diseño $\boldsymbol{x}_c^*$, los cuales hacen que el modelo burdo genere la respuesta o respuestas deseadas $\boldsymbol{R}^*$ cumpliendo con las especificaciones de diseño, sobre un rango de interés para las variables independientes contenidas en el vector $\boldsymbol{\psi}$.

Para problemas en el dominio de la frecuencia, el vector $\boldsymbol{\psi}$ contiene todos los puntos en frecuencia, P_F, sobre los

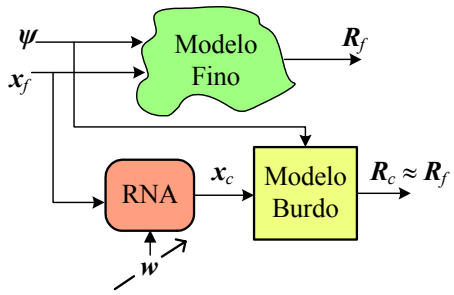


Fig. 3 Entrenamiento del neuromodelo basado en mapeo espacial.

cuales están muestreadas las respuestas del circuito. El vector de respuestas optimizables del modelo burdo, se puede denotar de la siguiente manera,

$$\boldsymbol{R}_c(\boldsymbol{x}_c) = \left[\boldsymbol{R}_c^1(\boldsymbol{x}_c)^T \ldots \boldsymbol{R}_c^m(\boldsymbol{x}_c)^T\right]^T \quad (3)$$

donde,

$$\boldsymbol{R}_c^r(\boldsymbol{x}_c) = \left[R_c^r(\boldsymbol{x}_c, \psi_1) \ldots R_c^r(\boldsymbol{x}_c, \psi_{P_F})\right]^T \quad r = 1,\ldots m \quad (4)$$

y r, es el número de respuestas de interés.

La formulación del problema de optimización del modelo burdo se puede plantear de la siguiente manera, como en [10],

$$\boldsymbol{x}_c^* = \arg\min_{\boldsymbol{x}_c} U(\boldsymbol{R}_c(\boldsymbol{x}_c, \boldsymbol{\psi})) \quad (5)$$

donde U es la función objetivo, usualmente minimax, la cual está expresada en términos de las especificaciones de diseño para cada respuesta y rango de frecuencia.

3.3. Entrenamiento del Neuromodelo basado en Mapeo Espacial

Durante el entrenamiento del neuromodelo mapeado a cada iteración i del algoritmo, queremos encontrar el mapa $\boldsymbol{M}^{(i)}$ más simple posible, que haga que las respuestas del modelo burdo mapeado se aproximen a las respuestas del modelo fino en todos los puntos de entrenamiento (ver Fig. 3). Para poder llevar a cabo la aproximación del mapa $\boldsymbol{M}$ de una manera más confiable y eficiente, en vez de usar las respuestas optimizables de cada modelo $\boldsymbol{R}_c$ y $\boldsymbol{R}_f$, se toman en cuenta todo el conjunto de respuestas características disponibles en ambos modelos, $\boldsymbol{R}_{cs}$ y $\boldsymbol{R}_{fs}$, como en [11], y se resuelve el siguiente problema de optimización:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left\| \left[\ldots \boldsymbol{e}_l^T \ldots\right]^T \right\|_2 \quad (6)$$

$$\boldsymbol{e}_l = \boldsymbol{R}_{fs}(\boldsymbol{x}_f^{(l)}) - \boldsymbol{R}_{cs}(\boldsymbol{M}(\boldsymbol{x}_f^{(l)}, \boldsymbol{w})) \quad l = 1,\ldots 2n+i \quad (7)$$

donde l es el contador de puntos de entrenamiento, siendo $2n+1$ los puntos de entrenamiento iniciales.

Se puede notar que durante este proceso siempre se busca obtener un vector de parámetros de entrada del modelo burdo que hagan que sus respuestas estén lo más cercanas posible a las respuestas del modelo fino. El vector respecto del cual se va a llevar a cabo la optimización es $\boldsymbol{w}$, el cual contiene los parámetros internos

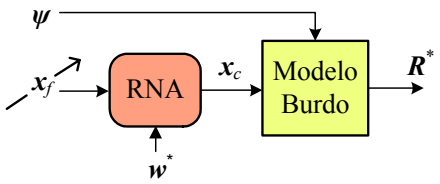


Fig. 4 Optimización del neuromodelo mapeado.

de la RNA (pesos, polarizaciones, etc.).

En cada iteración *i*, el entrenamiento se inicia con un mapa unitario, ya que se considera que el modelo burdo en realidad no es tan inexacto,

$$\boldsymbol{M}^{(0)}(\boldsymbol{x}_f^l, \boldsymbol{w}) = \boldsymbol{x}_f^l \qquad l = 2n + i \tag{8}$$

donde $\boldsymbol{w}$ contiene los parámetros que generan el mapa unitario.

Durante el entrenamiento, la complejidad de la RNA se incrementa gradualmente, controlando este proceso con el error final de entrenamiento ε_L, iniciando con un perceptrón de dos capas (mapa lineal), y de ahí se pasa a un perceptrón de tres capas. El error de entrenamiento se obtiene de la siguiente manera,

$$\varepsilon_L = \left\| \left[\ldots \boldsymbol{e}_l^T \ldots \right]^T \right\|_2 \tag{9}$$

donde $\boldsymbol{e}_l$ se obtiene de (7), usando los actuales valores óptimos de los parámetros internos de la RNA, $\boldsymbol{w}^*$.

El método de optimización que se utiliza para obtener (6) emplea una estrategia de región confiable basada en el método de Newton reflectivo (se utiliza la función "lsqnonlin", *nonlinear least-squares*, que se encuentra en el Toolbox de optimización de Matlab).

3.4. Optimización del Neuromodelo Mapeado

Una vez que se tiene el neuromodelo mapeado, entrenado con un error de aprendizaje lo suficientemente pequeño, se toma a éste como un modelo burdo mejorado, para después optimizar sus parámetros de entrada (ver Fig. 4), y lograr generar la respuesta deseada $\boldsymbol{R}^*$, y de ahí obtener el siguiente punto en el algoritmo. La respuesta de nuestro neuromodelo mapeado $\boldsymbol{R}_{NM}$ se define de la siguiente manera,

$$\boldsymbol{R}_{NM}(\boldsymbol{x}_f) = \boldsymbol{R}_c(\boldsymbol{M}(\boldsymbol{x}_f, \boldsymbol{w}^*), \boldsymbol{\psi}) \tag{10}$$

manejando *m* diferentes respuestas del circuito, la respuesta sería,

$$\boldsymbol{R}_{NM}(\boldsymbol{x}_f) = \left[\boldsymbol{R}_{NM}^1(\boldsymbol{x}_f)^T \ldots \boldsymbol{R}_{NM}^m(\boldsymbol{x}_f)^T \right]^T \tag{11}$$

$$\boldsymbol{R}_{NM}^r(\boldsymbol{x}_f) = \left[R_f^r(\boldsymbol{x}_f, \psi_1) \ldots R_f^r(\boldsymbol{x}_f, \psi_{P_F}) \right]^T \tag{12}$$

donde $r = 1,\ldots,m$ y cada respuesta está muestreada en frecuencia P_F veces.

La solución del siguiente problema de optimización nos entrega el punto predicho por el algoritmo,

$$\boldsymbol{x}_f^{(i+1)} = \arg\min_{\boldsymbol{x}_f} U(\boldsymbol{R}_c(\boldsymbol{M}(\boldsymbol{x}_f, \boldsymbol{w}^*), \boldsymbol{\psi})) \tag{13}$$

donde la función objetivo *U* se define igual que en (5). A la solución entregada por el algoritmo de diseño basado en mapeo espacial neural se le define como $\boldsymbol{x}_f^{NM}$.

3.5. Criterios de Finalización del Algoritmo

En éste algoritmo existen cinco criterios de terminación: el primero es exclusivamente para detener el entrenamiento del neuromodelo mapeado en la *i*-ésima iteración, lo cual se hace cuando el máximo error relativo entre las respuestas características de ambos modelos sea lo suficientemente pequeño,

$$\frac{\left\| \boldsymbol{R}_{fs}\left(\boldsymbol{x}_f^{(k)}\right) - \boldsymbol{R}_{cs}\left(\boldsymbol{x}_c^{(k)}\right) \right\|_\infty}{\left\| \boldsymbol{R}_{fs}\left(\boldsymbol{x}_f^{(k)}\right) \right\|_\infty} \le \varepsilon_1 \qquad k = 1,..,\, 2n+i \tag{14}$$

Los otros cuatro criterios son para la terminación general del algoritmo: cuando el máximo error relativo entre la respuesta del modelo fino y la respuesta deseada sea lo suficientemente pequeño (15a); cuando el cambio relativo en los parámetros de diseño del modelo fino sea lo suficientemente pequeño (15b); cuando un número máximo de iteraciones sea alcanzado (15c), o cuando se ha alcanzado un número máximo de neuronas en la capa oculta de la RNA (15d).

$$\frac{\left\| \boldsymbol{R}_f\left(\boldsymbol{x}_f^{(i)}\right) - \boldsymbol{R}_c\left(\boldsymbol{x}_c^*\right) \right\|_\infty}{\left\| \boldsymbol{R}_f\left(\boldsymbol{x}_f^{(i)}\right) \right\|_\infty} \le \varepsilon_2 \tag{15a}$$

$$\left\| \boldsymbol{x}_f{}^{(i+1)} - \boldsymbol{x}_f{}^{(i)} \right\|_2 \le \varepsilon_3 \left(\varepsilon_3 + \left\| \boldsymbol{x}_f{}^{(i)} \right\|_2 \right) \tag{15b}$$

$$i > 5n \tag{15c}$$

$$h > 3n \tag{15d}$$

4. MEJORAS IMPLEMENTADAS AL ALGORITMO

4.1. Control de la no Linealidad

Como se mencionó anteriormente, durante el entrenamiento del neuromodelo se requiere encontrar el mapa $\boldsymbol{M}^{(i)}$ más simple que haga que las respuestas del modelo burdo mapeado se aproximen lo más posible a las respuestas del modelo fino en todos los puntos de entrenamiento disponibles en la iteración *i* del algoritmo. Asumiendo que el modelo burdo no es tan inexacto, en cada iteración el entrenamiento se inicia con un mapa lineal, primero para el perceptrón de dos capas mediante

$$\boldsymbol{x}_c = \boldsymbol{W}^o \boldsymbol{x}_f + \boldsymbol{b}^o \tag{16}$$

donde $\boldsymbol{W}^o \in \Re^{n \times n}$ es la matriz de pesos de las neuronas de salida, $\boldsymbol{b}^o \in \Re^n$ es el vector de polarizaciones de las neuronas de salida. La inicialización se lleva a cabo haciendo $\boldsymbol{W}^o = \boldsymbol{I}$ y $\boldsymbol{b}^o = \boldsymbol{0}$ (mapa unitario). El perceptrón de tres capas se implementa usando

$$\boldsymbol{z} = \boldsymbol{W}^h \boldsymbol{x}_f + \boldsymbol{b}^h, \quad \boldsymbol{x}_c = \boldsymbol{W}^o \boldsymbol{\Phi}(\boldsymbol{z}) + \boldsymbol{b}^h \tag{17ab}$$

donde $\boldsymbol{W}^h \in \Re^{h \times n}$, $\boldsymbol{b}^h \in \Re^h$ y $\boldsymbol{z} \in \Re^h$ son la matriz de pesos,

vector de polarizaciones y el vector de respuestas de las neuronas de la capa oculta, respectivamente, y h es el número de neuronas en la capa oculta y $\boldsymbol{\Phi} \in \Re^h$ es el vector de funciones de activación no lineal (en este caso se usan tangentes hiperbólicas). El mapa unitario inicial requerido se aproxima con $\boldsymbol{b}^o = \mathbf{0}$, $\boldsymbol{b}^h = \mathbf{0}$, $\boldsymbol{W}^h = 0.1[\boldsymbol{I}\ \mathbf{0}]^T$ y $\boldsymbol{W}^o = 10[\boldsymbol{I}\ \mathbf{0}]$.

Para poder regular la cantidad de no-linealidad del mapa entre el modelo burdo y el modelo fino durante todo el entrenamiento a cada iteración del algoritmo se utiliza

$$\boldsymbol{\Phi}(z) = \frac{1}{\beta}\begin{bmatrix} \varphi(\beta z_1) \\ \varphi(\beta z_2) \\ \vdots \\ \varphi(\beta z_h) \end{bmatrix} \tag{18}$$

donde $0.1 \le \beta \le 1$, siendo ésta la variable la que controla la no linealidad del mapa $\boldsymbol{M}$. Cuando β = 0.1 se esta implementando un mapa prácticamente lineal, y cuando β = 1 se esta usando la no linealidad plena de las funciones de activación.

4.2. Punto Predicho por el Algoritmo

En el algoritmo original, el punto predicho se obtiene optimizando el modelo burdo mapeado, como se muestra en la Fig. 4. Es decir, el siguiente punto en el algoritmo se predice resolviendo (13). Para la versión mejorada implementada en este trabajo (ver Fig. 5), el siguiente punto en el algoritmo se predice minimizando la diferencia entre la salida de la RNA y los parámetros óptimos de diseño del modelo burdo, $\boldsymbol{x}_c^*$, de la siguiente forma

$$\boldsymbol{x}_f^{(i+1)} = \arg\min_{\boldsymbol{x}_f} \left\| \left(\boldsymbol{M}\left(\boldsymbol{x}_f, \boldsymbol{w}^*\right) - \boldsymbol{x}_c^* \right) \right\|_2 \tag{20}$$

siendo $\boldsymbol{w}^*$ el vector que contiene los parámetros óptimos de la RNA en la i-ésima iteración.

4.3. Errores Máximos usados en los Criterios de Finalización

En el algoritmo original, los valores máximos permitidos para los criterios de finalización descritos anteriormente se especificaban de manera arbitraria.

En este algoritmo mejorado, el valor del máximo error relativo entre las respuesta características de ambos modelos, ε_1, y el valor del máximo error relativo entre la respuesta deseada y la respuesta del modelo fino evaluado en el punto predicho por el algoritmo, ε_2, los determina el perceptrón de dos capas con el máximo error obtenido por éste al terminar su entrenamiento, el cual se realiza en la primer iteración del algoritmo, únicamente. Este cambio resulta adecuado debido a que el error en el perceptrón de dos capas es capaz de darnos una idea de que tan no lineal es el mapa entre los dos modelos, y por lo tanto es un buen criterio para decidir cuando terminar los posteriores entrenamientos del neuromodelo mapeado y por lo tanto cuando finalizar el algoritmo.

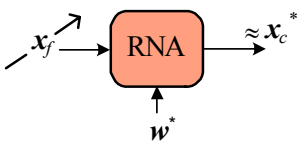


Fig. 5 Se encuentra $\boldsymbol{x}_f$ que hacen que la salida de la RNA sea aproximadamente igual a $\boldsymbol{x}_c^*$.

5. FILTRO RECHAZA BANDA TIPO MICROCINTA

Aplicamos el nuevo algoritmo a un filtro rechaza banda tipo microcinta con "stubs" abiertos resonantes de un cuarto de longitud de onda, mostrado en la Fig. 6. El modelo burdo esta implementado en Aplac^{MR} [12] utilizando líneas de transmisión ideales sin pérdidas para modelar cada sección de línea microcinta (ver Fig. 7). Para modelar cada segmento de línea microcinta se utilizaron las formulas de [13] para obtener la capacitancia y la inductancia por unidad de longitud, además de la constante dieléctrica efectiva.

Como se puede apreciar en la Fig. 6, L_1 y L_2 son las longitudes de los "stubs" abiertos mientras que W_1 y W_2 son sus correspondientes anchos. Las variables de diseño del modelo burdo son $\boldsymbol{x}_c = [W_{c1}\ W_{c2}\ L_{c0}\ L_{c1}\ L_{c2}]^T$ (mil). El modelo fino considera la estructura real del filtro, el cual usa alumina como substrato la cual tiene una constante dieléctrica ε_r = 9.4, con una altura H = 25 mil. El ancho de la línea de alimentación es W_{50} = 25 mil para mantener una impedancia característica a la entrada de 50 Ω. Para simular el modelo fino se utilizó Sonnet em^{MR} [14], empleando una resolución de 0.5mil×0.5mil por celda, consumiendo aproximadamente 13 minutos por cada simulación usando una computadora con procesador a 2.4 GHz y 512 MB de memoria, mientras que la simulación del modelo burdo es prácticamente instantánea usando la misma computadora. Las variables de diseño del modelo fino son $\boldsymbol{x}_f = [W_{f1}\ W_{f2}\ L_{f0}\ L_{f1}\ L_{f2}]^T$ (mil). Se puede apreciar en las Fig. 6 y 7, que $L_{c2} = L_2 - W_{50}/2$, $L_{c1} = L_1 - W_{50}/2$, y

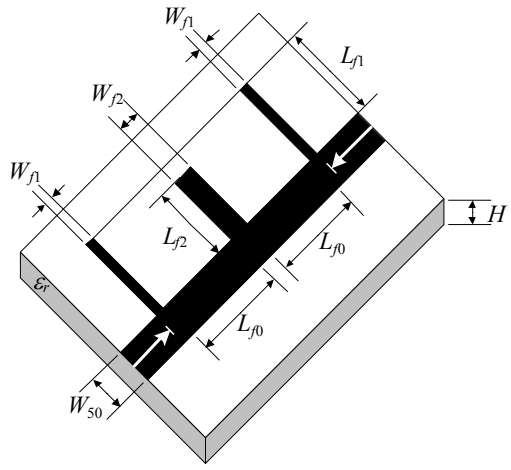


Fig. 6 Filtro rechaza banda con stubs resonantes abiertos.

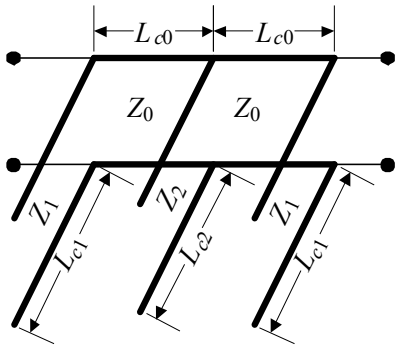


Fig. 7 Filtro rechaza banda tipo microcinta usando líneas de transmisión ideales.

$L_{c0} = L_0 - W_1/2 - W_2/2$.

Las especificaciones de diseño son: $|S_{21}| \leq 0.05$ de 9.3 GHz a 10.7 GHz, $|S_{21}| \geq 0.9$ de 5 GHz a 8 GHz y de 12GHz a 15 GHz. Los valores utilizados para los criterios de finalización son: $\varepsilon_1 = 0.61$, $\varepsilon_2 = 0.61$ y $\varepsilon_3 = 0.01$.

La solución óptima del modelo burdo es $\boldsymbol{x}_c^* = [4.5 \;\; 9.5 \;\; 105 \;\; 109.5 \;\; 107.5]^T$ (mil). En este caso se usó un porcentaje de desviación de 50% para W_{f1}, W_{f2} y L_{f0},

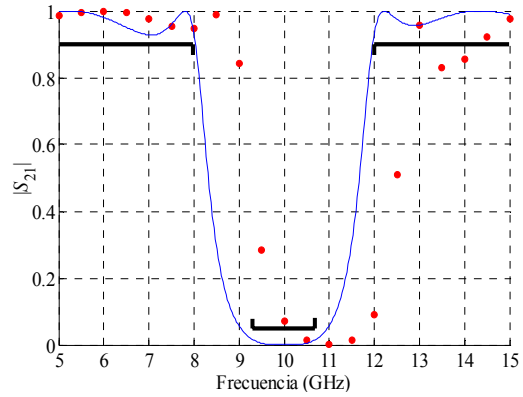


Fig. 8 Respuesta del modelo burdo (–) y fino (•) en la solución óptima del modelo burdo $\boldsymbol{x}_c^*$.

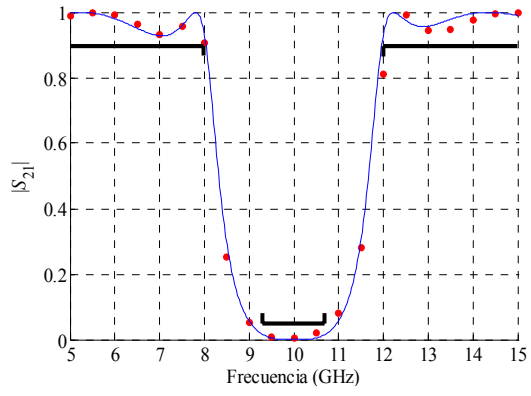


Fig. 9 Respuesta del modelo burdo (–) y fino (•) en la solución encontrada por el algoritmo $\boldsymbol{x}_f^{NM}$.

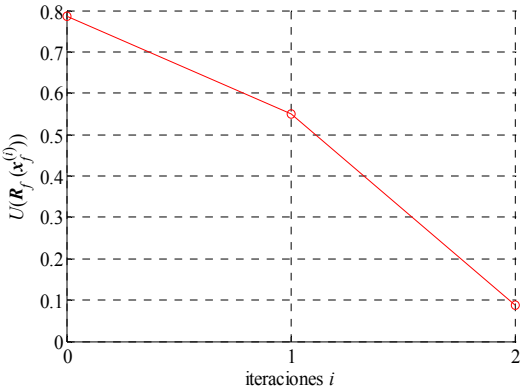


Fig. 10 Valor de la función objetivo del modelo fino a cada iteración del algoritmo (-o-).

mientras que para L_{f1} y L_{f2}, se usó 15% para generar los $2n$ puntos alrededor de $\boldsymbol{x}_c^*$.

Las respuestas del modelo burdo y fino evaluadas en la solución óptima $\boldsymbol{x}_c^*$ aparecen en la Fig. 8, donde se aprecia que la respuesta del modelo fino está violando dos de las especificaciones de diseño, obteniendo un máximo error relativo del 84% con respecto a la respuesta deseada.

La solución encontrada por el algoritmo es, $\boldsymbol{x}_f^{NM} = [6.5 \;\; 18 \;\; 101.5 \;\; 123.5 \;\; 115.5]^T$, cumpliendo el segundo de los criterios de finalización del algoritmo, ya que se obtiene un máximo error relativo del 12% entre la respuesta óptima del modelo burdo y la respuesta obtenida con el siguiente punto predicho por el algoritmo.

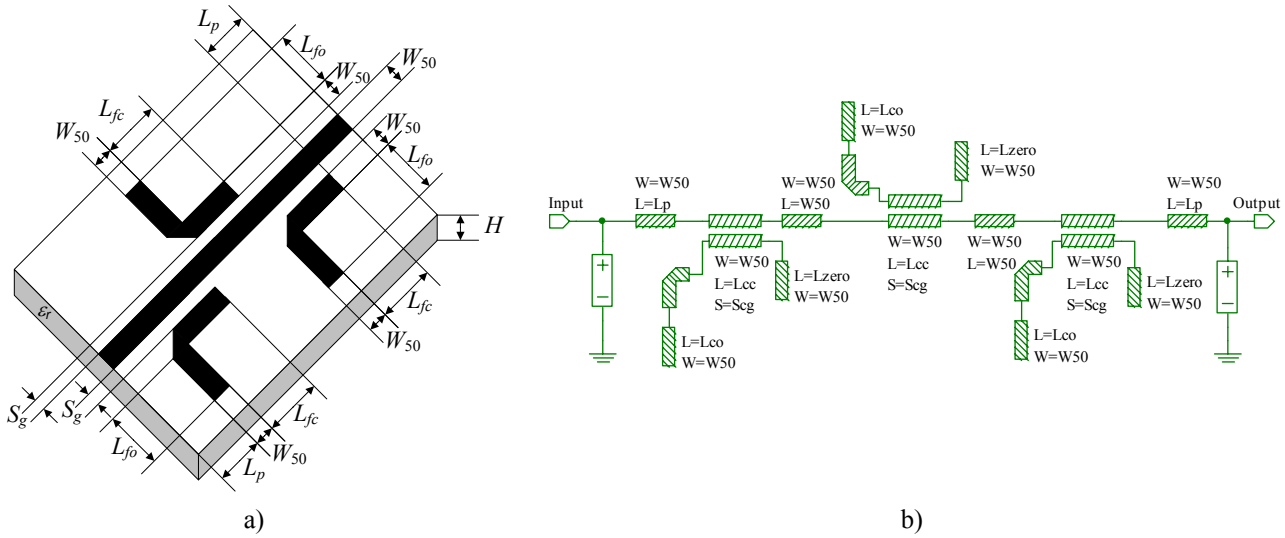
La respuesta del modelo fino evaluado en la solución del algoritmo, $\boldsymbol{x}_f^{NM}$, y la respuesta deseada, se muestran en la Fig. 9, mientras que los valores que tomó la función objetivo a cada iteración del algoritmo se muestran en la Fig. 10.

Como se puede apreciar, el resultado se obtuvo en dos iteraciones , es decir, 13 simulaciones del modelo fino (11 iniciales más dos).

6. FILTRO NOTCH TIPO MICROCINTA

En esta ocasión se diseña un filtro notch tipo microcinta de alta selectividad de rechazo, el cual se muestra en la Fig. 11a. En este caso, el modelo burdo (Fig. 11b) esta implementado con líneas de transmisión microcinta, y es simulado circuitalmente con Aplac. Las variables de diseño del modelo burdo son $\boldsymbol{x}_c = [L_{cc} \;\; L_{c0} \;\; S_{cg}]^T$ (mil).

El modelo fino es la estructura real del filtro, Fig. 11a, cuyo substrato es RT-Duroid-5880 con una constante dieléctrica relativa de $\varepsilon_r = 2.2$ y una perdida tangencial del dieléctrico de 9×10^{-4}. El substrato tiene una altura $H = 10$ mil, el ancho de la línea de alimentación es $W_{50} = 10$ mil para mantener una impedancia característica de entrada de 50 Ω.



a) b)

Fig. 11 Fitro notch tipo microcinta: a) modelo fino, b) modelo burdo.

Para simular el modelo fino se utilizó Sonnet *em*$^{\text{MR}}$, empleando una resolución de 1mil×1mil por celda, con la cual una computadora con procesador a 2.4 GHz y 512 MB de memoria consume 65 minutos por cada simulación, mientras que la simulación del modelo burdo es casi instantánea usando la misma computadora. Las variables de diseño del modelo fino son $\boldsymbol{x}_f = [L_{fc}\ \ L_{f0}\ \ S_{fg}]^T$ (mil).

Las especificaciones de diseño son: $|S_{21}| \leq 0.025$ de 13.19 GHz a 13.21 GHz, $|S_{21}| \geq 0.95$ de 12.8 GHz a 13.07 GHz y de 13.33 GHz a 13.8 GHz. Como se puede apreciar en las especificaciones, el filtro a diseñar es bastante selectivo en su banda de rechazo. Los valores utilizados para los criterios de finalización son: $\varepsilon_1 = 0.65$, $\varepsilon_2 = 0.65$ y $\varepsilon_3 = 0.01$.

La solución óptima del modelo burdo encontrada es $\boldsymbol{x}_c^* = [130\ \ 170\ \ 9]^T$ (mil). En este caso se usó un porcentaje de desviación del 2% para L_{fc} y L_{f0}, y 30% para S_{fg}, para generar los $2n$ puntos alrededor de $\boldsymbol{x}_c^*$.

La respuestas del modelo burdo y del modelo fino evaluadas en la solución óptima $\boldsymbol{x}_c^*$ aparecen en la Fig. 12, donde se aprecia que la respuesta del modelo fino está violando las especificaciones de diseño, obteniendo un máximo error relativo del 94% con respecto a la respuesta deseada.

Después de aplicar el nuevo algoritmo a este problema, se encontró la siguiente solución, $\boldsymbol{x}_f^{NM} = [127\ \ 175\ \ 10]^T$, cumpliendo con el segundo de los criterios de finalización del algoritmo, ya que se obtiene un máximo error relativo del 19% entre la respuesta óptima del modelo burdo y la respuesta obtenida con el siguiente punto predicho por el algoritmo.

La respuesta del modelo fino evaluado en la solución del algoritmo, $\boldsymbol{x}_f^{NM}$ y la respuesta deseada, se muestran en la Fig. 13, mientras que los valores que tomó la función objetivo a cada iteración del algoritmo se muestran en la Fig.14. Cabe señalar que en [15] se llevó a cabo el diseño y construcción de este filtro notch, tanto las simulaciones electromagnéticas como las mediciones del circuito muestran que su respuesta no puede baja a menos de −28 dB, debido a esto, es muy probable que no sea posible

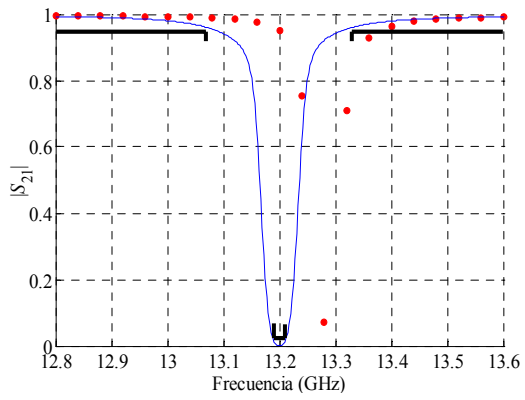


Fig. 12 Respuesta del modelo burdo (–) y fino (•) en la solución óptima del modelo burdo $\boldsymbol{x}_c^*$.

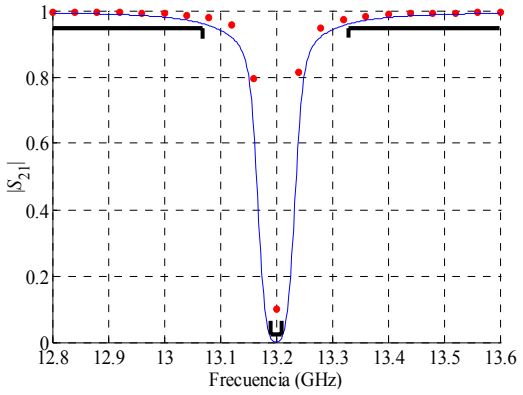


Fig. 13 Respuesta del modelo burdo (–) y fino (•) en la solución predicha por el algoritmo $\boldsymbol{x}_f^{NM}$.

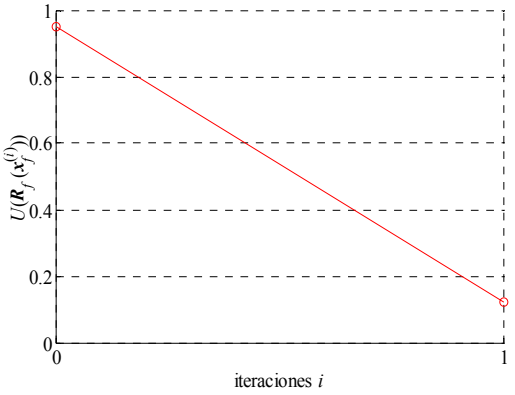


Fig. 14 Valor de la función objetivo del modelo fino a cada iteración del algoritmo (-o-)

cumplir con las especificaciones de diseño para el modelo fino establecidas al inicio del problema.

Para obtener el resultado el algoritmo realizó una sola iteración, es decir, 8 simulaciones del modelo fino (7 iniciales más una).

7. CONCLUSIONES

En este trabajo se realizan mejoras al algoritmo de diseño de circuitos electrónicos basado en mapeo espacial neural. La modificación de más importancia es la que se lleva a cabo durante el entrenamiento del neuromodelo mapeado, controlando la no-linealidad del mapa entre el modelo fino y el modelo burdo durante todo el entrenamiento mediante un factor β. También la nueva forma como se predice el siguiente punto en el algoritmo hace a éste más eficiente, y el dejar que el perceptrón de dos capas establezca los valores de error para la terminación del entrenamiento del neuromodelo mapeado y para terminar el algoritmo, vuelve al algoritmo más robusto y menos dependiente de la intervención del usuario.

Se comprobó el desempeño del algoritmo diseñando dos circuitos microcinta: un filtro rechaza banda con "stubs" abiertos resonantes de un cuarto de longitud de onda, y un filtro notch, y para ambos circuitos se utilizan simuladores electromagnéticos, lo cual hace los diseños más realistas, y por lo tanto se prueba el objetivo principal del algoritmo, que es el de diseñar circuitos con el menor número de simulaciones intensivas. Los resultados obtenidos verifican que el algoritmo utiliza pocas evaluaciones del modelo fino para poder encontrar una solución satisfactoria al problema.

8. REFERENCIAS

[1] P. Burrascano and M. Mongiardo, "A review of artificial neural networks applications in microwave CAD," Int. *J. RF and Microwave CAE*, vol. 9, pp. 158-174, May 1999.

[2] V.K. Devabhaktuni, M.C.E. Yagoub, Y. Fang, J.J. Xu, Q.J. Zhang, "Neural networks for microwave modeling: model development issues and nonlinear modeling techniques," *Int. J. RF and Microwave CAE*, vol. 11, pp. 4-21, Jan. 2001.

[3] J. E. Rayas-Sánchez, "EM-based optimization of microwave circuits using artificial neural networks: the state of the art," *IEEE Trans. Microwave Theory Tech*., vol. 52, pp. 420-435, Jan. 2004.

[4] J. W. Bandler, M. A. Ismail, J. E. Rayas-Sánchez, and Q. J. Zhang, "Neuromodeling of microwave circuits exploiting space mapping technology," *IEEE Trans. Microwave Theory Tech*., vol. 47, pp. 2417–2427, Dec. 1999.

[5] M.H. Bakr, J.W. Bandler, M.A. Ismail, J.E. Rayas-Sánchez and Q.J. Zhang, "Neural space mapping optimization for EM-based design," *IEEE Trans. Microwave Theory Tech*., vol. 48, pp. 2307-2315, Dec. 2000.

[6] V. Gutiérrez-Ayala and J. E. Rayas-Sánchez, "Implementación en MATLAB del algoritmo de diseño de circuitos electrónicos basado en mapeo espacial neural," Internal Report *CAECAS-04-01-R*, Nov. 2004.

[7] Matlab^{MR} Optimization Toolbox, Version 2.1 (R12), The MathWorks, Inc., 3 Apple Hill Drive, Natick MA 01760-2098, 2000.

[8] J. W. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Trans. Microwave Theory Tech*., vol. 42, pp. 2536–2544, Dec. 1994.

[9] R. M. Biernacki, J. W. Bandler, J. Song, and Q. J. Zhang, "Efficient quadratic approximation for statistical design," *IEEE Trans. Circuit Syst*., vol. 36, pp. 1449–1454, Nov. 1989.

[10] J. W. Bandler and S. H. Chen, "Circuit optimization: The state of the art," *IEEE Trans. Microwave Theory Tech*., vol. 36, pp. 424–443, Feb. 1988.

[11] J. W. Bandler, M. A. Ismail, J. E. Rayas-Sánchez, and Q. J. Zhang, "Neural inverse space mapping EM-optimization," *IEEE MTT-S Int. Microwave Symp. Dig*., Phoenix, AZ, July 2001, pp. 1007–1010.

[12] APLAC^{MR} Version 7.70b, APLAC Solutions Corporation, Lars Sonckin kari 16, Helsinki, Finland, FIN-02600 Espo, 2002.

[13] M. Pozar, *Microwave Engineering*. Amherst, MA: Wiley, 1998, p. 162.

[14] *em*^{MR} Suite Version 8.52, Sonnet Software, Inc., 1020 Seventh North Street, Suite 210, Liverpool, NY 13088, 2002.

[15] M. S. Narayana, N. Gogia, "Accurate design of a notch filter using electromagnetic simulators," *Applied Microwave & Wireless*, vol. 12, pp. 44-48, Nov. 2000.