Universitat
Autònoma
de Barcelona

CALIBRACIÓN TLR Y LRM EN MATLAB PARA MEDIDAS CON GUÍAS DE ONDA

Memòria del Treball Final de Carrera d'Enginyeria Tècnica de Telecomunicació, especialitat Sistemes Electrònics realitzat per

.....

i dirigit per

.....

Bellaterra,.....de.....…....de 200...

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, ...

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: .....

Bellaterra, ...de.....de 200.....

ÍNDICE

1 INTRODUCCIÓN

1.1 Guías de onda y líneas de transmisión

Un concepto que se debe introducir en el proyecto para su buena comprensión es el de guías de onda o líneas de transmisión, que se pueden definir como aquellas estructuras destinadas a la propagación orientada y confinada de la radiación electromagnética que se pueden construir con materiales conductores o dieléctricos en función de la frecuencia. Las guías de onda de un solo conductor tienen una frecuencia de corte por debajo de la cual la transmisión es imposible.

La principal diferencia que se establece entre las líneas de transmisión o guías de onda y los circuitos clásicos es el tamaño eléctrico. En las líneas de transmisión, la longitud de onda de la señal transmitida es muy inferior al tamaño físico de la red mientras que para los circuitos clásicos, el tamaño físico sea inferior a la longitud de onda.

Las consecuencias de esta diferencia entre las líneas de transmisión y los circuitos clásicos, es que las primeras no pueden analizarse de la misma manera, pues la intensidad y el voltaje variaran a lo largo de la línea. Además al aumentar la frecuencia de funcionamiento, los elementos pasivos tales como condensadores y bobinas usados en circuitos clásicos, presentan alteraciones. Eso no significa que no podamos introducir elementos que se comporten como condensadores o bobinas.

Otro concepto a tener en cuenta que también dificulta el análisis es el que se produce cuando se genera una onda estacionaria. Las líneas de transmisión tienen una impedancia característica; si la línea termina en una impedancia igual a la característica diremos que está acoplada y no se generará ninguna onda estacionaria. Si por el contrario la impedancia del final de la línea es distinta a la característica se genera una onda estacionaria con lo que, para establecer la onda que viaja a través de la línea (que será la suma de la onda incidente más la reflejada) se tendrá que añadir el concepto de coeficiente de reflexión, necesario para definir la onda que rebota por causa del desacoplamiento de las impedancias

Por todo esto, el análisis mediante voltajes e intensidades, aunque exista, resulta más complicado por lo que para definirlas también se usan los llamados parámetros S o parámetros de *scattering* que introducimos a continuación y que simplifican su análisis.

1.2 Parámetros de Scattering

A altas frecuencias una red eléctrica podría ser caracterizada por una matriz de impedancias o admitancias que relacionase las tensiones y corrientes en cada uno de sus puertos pero esto supondría tener que contar con dos grandes inconvenientes: primero el hecho que la medida de la impedancia es muy sensible a desplazamientos del plano de referencia y segundo que las condiciones de contorno que se deben aplicar para obtener la matriz de impedancias o admitancias son muy difíciles de obtener en alta frecuencia. Por dichos motivos para definir una red eléctrica a alta frecuencia se suele usar una variante de dicha matriz conocida como matriz de parámetros de scattering o de dispersión (parámetros S).

Por tanto podemos definir los parámetros de dispersión como parámetros que se utilizan para describir el comportamiento de redes eléctricas usados especialmente en redes de radiofrecuencias o microondas por los motivos que acabamos de explicar, a pesar de que podrían aplicarse a cualquier frecuencia. Los parámetros S vienen dados en función de la frecuencia y están

definidos por matrices en los que cada valor se expresa como un número complejo ya sea mediante valores de la parte real y la parte imaginaria o mediante los valores de su módulo y su fase. Gracias a dichos parámetros se pueden expresar numerosas propiedades útiles de las redes tales como la ganancia, la pérdida por retorno, la relación de onda estacionaria o el coeficiente de reflexión.

A continuación presentamos la relación que mantienen los parámetros de scattering con las ondas entrantes y salientes en una red multi-puerto genérica. Para ello se asume que todos los puertos excepto el que se encuentra bajo consideración están bajo la condición de adaptación de impedancia y, por tanto, sólo existe excitación en el puerto que estamos mirando. Así pues la relación queda así [5]:

$$\begin{pmatrix} b_1 \\ . \\ . \\ . \\ b_n \end{pmatrix} = \begin{pmatrix} s_{11} & s_{12} & . & s_{1n} \\ s_{21} & . & . & s_{2n} \\ . & . & . & . \\ . & . & . & . \\ s_{n1} & . & . & s_{nn} \end{pmatrix} \begin{pmatrix} a_1 \\ . \\ . \\ . \\ a_n \end{pmatrix}$$

Donde [b] i [a] , suponiendo que todos los puertos tengan una misma impedancia característica, son:

$b_n = \frac{V_n^-}{\sqrt{Z_o}}$ correspondiente a la onda saliente del circuito

$a_n = \frac{V_n^+}{\sqrt{Z_o}}$ correspondiente a la onda entrante del circuito

Y los parámetros de dispersión se obtienen:

$$s_{ij} = \left. \frac{b_i}{a_j} \right|_{a_k = 0 \, \forall \, k \neq i}$$

Cabe resaltar que al tomar los elementos de la diagonal obtendremos el coeficiente de reflexión pues estaremos dividiendo la onda reflectada por la onda incidente en el mismo puerto.

1.3 Calibración de medidas obtenidas con el analizador vectorial de redes

Hasta ahora hemos visto qué son los parámetros S y cuáles son sus utilidades, a partir de ahora veremos como se obtienen de forma no manual, es decir, de manera automática. Para medir los parámetros S de una red se utiliza un instrumento llamado analizador vectorial de redes. El problema que surge al usar dicho aparato es que los resultados obtenidos no corresponden a los resultados reales debido a una serie de no idealidades que introducen cierto error en la medida.

Por este motivo, a lo largo del tiempo, se han ido desarrollando distintos métodos matemáticos de calibración para corregir dichos errores tales como SOLT (*Short-Open-Load-Thru*) o los que veremos más a fondo y sobre los cuales está basado este proyecto: TRL (*Thru-Reflect-Line*) y LRM (*Line-Reflect-Match*).

Dichos métodos matemáticos están basados en la resolución de una serie de ecuaciones, obtenidas a partir de la medida de ciertas redes concretas tales como circuitos abiertos,

cortocircuitos, circuitos con carga adaptada...La resolución de los mismos no es un proceso sencillo ni rápido motivo por el cual es conveniente tener un programa que nos los resuelva con sólo introducirle los datos necesarios. Para ello, en este proyecto se implementarán en *matlab* los algoritmos correspondientes tanto del método TRL como del método LRM con el fin de automatizar la resolución de los mismos de manera que cualquier usuario pueda hacerlo aún sin tener conocimiento de como funcionan. Una vez hecho esto se realizará una comparación entre los dos métodos de calibración a través de los resultados obtenidos tanto para uno como para el otro en las medidas de los parámetros s de distintos circuitos.

El motivo por el cual no hemos implementado el método de calibración SOLT es debido a que nos es muy difícil definir bien el *open* en una guía de onda. También hay que resaltar que, a parte de los tres métodos presentados, existe algún otro como el TSD (*Thru-Short-Delay*) aunque no entraremos en ello debido a la poca repercusión que tienen.

En el laboratorio contamos con dos modelos de analizador vectorial. El primero es el *Agilent 8714ES* que opera entre las frecuencias de 300 KHz – 3 GHz. Este analizador cuenta con dos *kits* de calibración, el 85033E/3.5mm y el 85032B/Tipo N que nos permiten tomar las medidas necesarias que luego necesitaremos para realizar la calibración. En el caso de este analizador, si queremos hacer una calibración TRL o LRM, hay que hacerlas aparte con un programa como el que hemos desarrollado en este proyecto, pues sólo lleva incorporado el programa de calibración SOLT.

El segundo analizador del que disponemos es el *Agilent N5230A* (apéndice figura 8.2)cuyo rango de operación es de 10Mhz – 20 Ghz. Tiene el *kit* de calibración 85052D/3.5mm y, a diferencia del otro, incorpora un programa que hace automáticamente la calibración. Este nos servirá para comprobar que el algoritmo desarrollado realmente funciona para casos reales.

1.4 Programas usados en el proyecto

En este apartado haremos una breve introducción de los dos principales programas que usaremos para desarrollar este proyecto: *matlab*, potente herramienta matemática especializada en operaciones con matrices y *puff*, programa usado para la simulación de líneas de transmisión.

1.4.1 Matlab

El nombre de *matlab* proviene de *"matrix laboratory"* y, como ya hemos avanzado, es un programa orientado para llevar a cabo elevados cálculos matemáticos y la visualización gráfica de los mismos y está especialmente desarrollado para el cálculo con matrices. Esta propiedad junto con el hecho de que en *matlab* se puedan crear programas que realicen una cierta función a partir de unas entradas hacen que sea especialmente útil para nuestro propósito.

Este programa se creó al surgir la necesidad de programas de cálculo más potentes y su creador fue Cleve Moler, un matemático y programador especializado en análisis numéricos. Los inicios de este programa se encuentran en la creación de las librerías *linpack* y *eispack* por parte de dicho matemático usadas en el lenguaje de programación *fortran* para cálculos numéricos. Posteriormente, Cleve Moler, desarrollaría *matlab* con la finalidad de que los estudiantes universitarios no tuvieran que usar el ya nombrado lenguaje de programación para usar las librerías *linpack* y *eispack*. Finalmente en 1984, junto con Jack Little, fundaron *The Math Works* para comercializar *matlab*. A lo largo de los años se le han ido añadiendo funciones y mejoras hasta convertirse en la herramienta informática de cálculo que hoy en día conocemos.

En el proyecto usaremos este programa para desarrollar funciones implementadas mediante operaciones matemáticas y lenguaje de programación que realicen todos los pasos necesarios para corregir los errores de medición aplicando los sistemas de corrección TRL o LRM. Dichas funciones se podrán ejecutar desde *matlab* y al hacerlo tendremos que introducir los datos necesarios para su correcto funcionamento.

1.4.2 Puff

El programa que presentamos a continuación sirve para calcular los parámetros S de un determinado circuito que puede estar formado de elementos pasivos (tales como resistencias, condensadores, bobinas...), lineas de transmisión u otros elementos. *Puff* nos da la opción de crear el circuito del cual extraeremos los parámetros S y, una vez hecho, escribimos los parámetros de simulación deseados (como puede ser el rango de frecuencia en el que deseamos simular) para, acto seguido, iniciar la simulación mediante la cual el programa nos muestra una gráfica en la carta de *smith* correspondiente a los parámetros S buscados y otra en la que nos salen los módulos de dichos parámetros (dB) en función de la frecuencia (GHz) . También nos guarda los resultados numéricos en ficheros para después poder operar con ellos.

Este programa está dividido en cuatro partes. La primera de ellas llamada *layout* y tal como su nombre indica es la que se utiliza para dibujar el esquemático del circuito a simular. La segunda llamada *plot* nos permite seleccionar aquellos parámetros S queremos que ponga tanto en la gráfica de los módulos como en la carta de *smith*. Después encontramos otra ventana en la que definimos las características de cada uno de los elementos que hemos incluido en nuestro esquemático, como podría ser por ejemplo, la impedancia y la longitud si habláramos de una línea de transmisión o la capacidad si habláramos de un condensador. Finalmente hay otra ventana llamada *board* en la que podemos definir parámetros más generales (hacen referencia a todo el circuito y no a elementos particulares) tales como la impedancia de normalización (por defecto 50 ohmios), la frecuencia de diseño o la constante dieléctrica del sustrato, para nombrar los más importantes.

2 CALIBRACIÓN TRL

2.1 Introducción

Al hablar de este método de calibración tenemos que hablar de Glenn F. Engen, persona que propuso una solución matemática para el mismo. Glenn F. Engen nació en Battle Creek el 26 de abril de 1925. Recibió el título de matemático y físico en 1947 en la universidad Andrews y, posteriormente, en 1965, el título de ingeniero electrónico en la universidad de Colorado. Se especializó en el campo de las microondas, tema sobre el cual escribió numerosos artículos de entre los cuales destacaremos, por ser el que en este proyecto tratamos, el que proponía una solución matemática para el método de corrección de errores Thru-Reflect-Line, escrito en 1979.

Cabe destacar que este método no era del todo correcto pues asumía que la línea de transmisión (*line*) era no reflectora. Posteriormente y, a partir de la solución propuesta por Engen, se ideó otra solución sin despreciar ningún parámetro en la línea de transmisión.

2.2 Método de calibración TRL (Thru-Reflect-Line)

Como ya comentamos en la introducción, el analizador vectorial, introduce ciertos parámetros de error, que a la hora de realizar esta calibración los definiremos como se muestra en la figura 2.1. Es importante fijarse en la notación de esta figura respecto a los nombres que damos a los parámetros S correspondientes a las caja de error A y B, pues dicha notación será usada en las ecuaciones que nos explican este método de calibración.

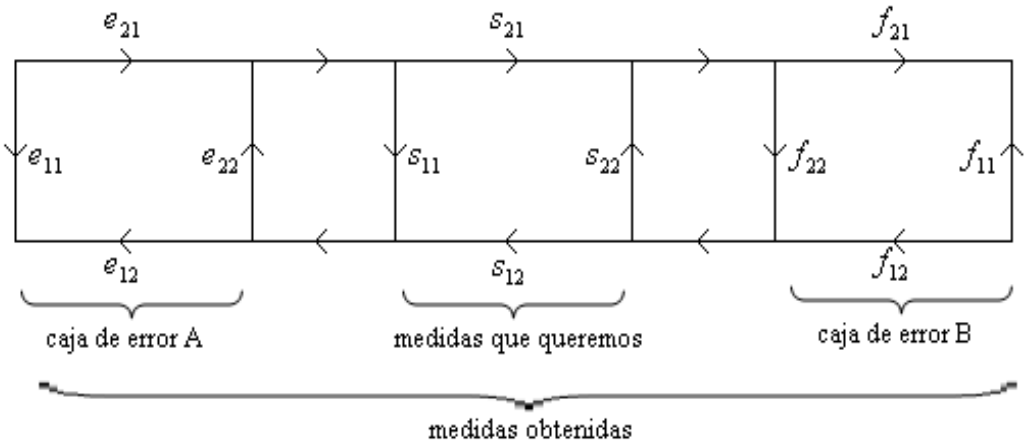


Figura 2.1 – Representación de los errores introducidos por el analizador

Así pues, para empezar la calibración mediante este método precisamos de tres líneas de transmisión que son, tal y como su nombre indica, una línea de transmisión muy corta o nula (*thru*), otra acabada en circuito abierto o cerrado (*reflect*) y finalmente una linea de transmisión de longitud indefinida (*line*). En el caso del *line,* hablamos de longitud indefinida pues la única condición que tiene que cumplir es que sea bastante más larga que la linea de transmisión usada para el *thru.* A continuación se muestra un diagrama donde se pueden observar las tres conexiones que debemos realizar para llevar a cabo este método de calibración. Tenemos que resaltar que no hemos incluido, por ser obvio, la medida de la muestra que viene representada en la figura 2.1 que acabamos de mostrar.

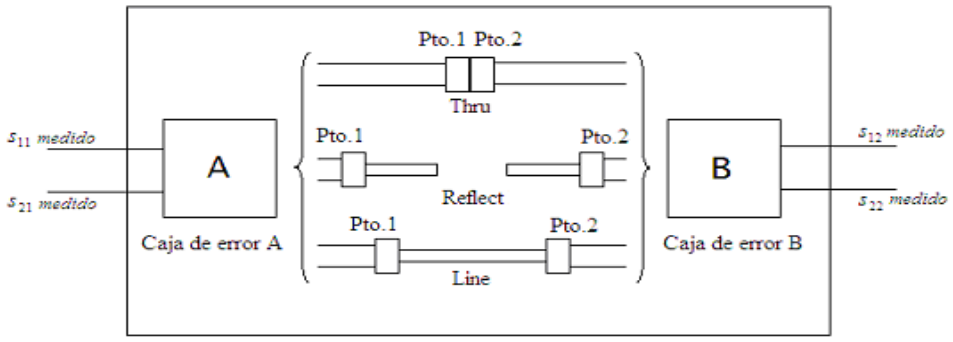


Figura 2.2 – Medidas a tomar para la calibración TRL

Una vez medidos los parámetros S en los tres casos se procederá a transformarlos en los llamados parámetros de transferencia o parámetros R, puesto que estos nos permiten analizar mejor varias redes de transmisión conectadas en cascada como es el caso. La relación entre ambos tipos de parámetros viene dada por:

$$R=\frac{1}{s_{21}}\begin{pmatrix}-\Delta & s_{11}\\ -s_{22} & 1\end{pmatrix} \quad donde \quad \Delta=s_{11}s_{22}-s_{12}s_{21} \tag{1}$$

Para empezar la calibración, siguiendo la metodología del capítulo 7 de [3], primero conectamos el *trhu* y mediante los parámetros S medidos obtenemos sus parámetros R. Definimos la matriz obtenida como:

$$R_{MT}=R_X R_{TH} R_Y \tag{2}$$

Donde R_X y R_Y representan las matrices de los parámetros R correspondientes a las cajas de error A y B respectivamente. Por otra parte, R_{TH} representa la matriz de parámetros R correspondiente al *thru* y tiene los siguientes valores:

$$R_{TH}=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \tag{3}$$

de lo que se obtiene:

$$R_{MT}=R_X R_Y \tag{4}$$

Aislando obtenemos:

$$R_Y=R_X^{-1} R_{MT} \tag{5}$$

A continuación obtenemos la matriz de parámetros R de la conexión del *line* que definimos como:

$$R_{ML}=R_X R_L R_Y \tag{6}$$

Si substituimos R_Y de (6) por (5) y operamos obtenemos como resultado:

$$MR_X=R_X R_L \tag{7}$$

donde

$$M=R_{ML}R_{MT}^{-1}=\begin{pmatrix}m_{11} & m_{12}\\ m_{21} & m_{22}\end{pmatrix} \tag{8}$$

por tanto

$$\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}=\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}\begin{pmatrix} e^{-\gamma L} & 0 \\ 0 & e^{+\gamma L} \end{pmatrix} \tag{9}$$

que representa el siguiente sistema de ecuaciones:

$$m_{11}x_{11}+m_{12}x_{21}=x_{11}e^{-\gamma L} \tag{10}$$

$$m_{21}x_{11}+m_{22}x_{21}=x_{21}e^{-\gamma L} \tag{11}$$

$$m_{11}x_{12}+m_{12}x_{22}=x_{12}e^{+\gamma L} \tag{12}$$

$$m_{21}x_{12}+m_{22}x_{22}=x_{22}e^{+\gamma L} \tag{13}$$

Resolvemos dicho sistema de ecuaciones dividiendo la primera y la tercera ecuación por la segunda y la cuarta respectivamente; el resultado son dos ecuaciones de segundo grado con los mismos términos constantes:

$$m_{21}\left(\frac{x_{11}}{x_{21}}\right)^2+(m_{22}-m_{11})\left(\frac{x_{11}}{x_{21}}\right)-m_{12}=0 \tag{14}$$

$$m_{21}\left(\frac{x_{12}}{x_{22}}\right)^2+(m_{22}-m_{11})\left(\frac{x_{12}}{x_{22}}\right)-m_{12}=0 \tag{15}$$

donde definiremos

$$a=\frac{x_{11}}{x_{21}}=e_{11}-\frac{(e_{21}e_{12})}{e_{22}} \quad y \quad b=\frac{x_{12}}{x_{22}}=e_{11} \tag{16}$$

Sabiendo que el módulo de *a* tiene que ser mayor que el módulo de *b*, podemos resolver las anteriores ecuaciones de segundo grado y asignarle el valor correcto a cada una de las dos incógnitas.

El siguiente paso a realizar es volver a la ecuación de salida del *trhu*, (4), pero esta vez aislamos la matriz de error X obteniendo:

$$R_X=R_{MT}R_Y^{-1} \tag{17}$$

substituyendo en la ecuación del *line*, (6)

$$R_Y N=R_L R_Y \quad donde \quad N=R_{MT}^{-1}R_{ML}=\begin{pmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{pmatrix} \tag{18}$$

Operamos con el sistema de ecuaciones formado por estas matrices de manera análoga al caso anterior y obtenemos dos ecuaciones de segundo grado:

$$n_{12}\left(\frac{y_{11}}{y_{12}}\right)^2+(n_{22}-n_{11})\left(\frac{y_{11}}{y_{12}}\right)-n_{21}=0 \tag{19}$$

$$n_{12}\left(\frac{y_{21}}{y_{22}}\right)^2+(n_{22}-n_{11})\left(\frac{y_{21}}{y_{22}}\right)-n_{21}=0 \tag{20}$$

donde definiremos

$$c=\frac{y_{11}}{y_{12}}=-f_{11}+(f_{12}\frac{f_{21}}{f_{22}}) \quad y \quad d=\frac{y_{21}}{y_{22}}=-f_{11} \tag{21}$$

En este caso el módulo de *c* deberá ser superior al módulo de *d* sabiendo así que solución escoger en cada caso de las dos que las ecuaciones de segundo grado nos dan.

Una vez obtenidos estos valores, buscamos la ecuación del coeficiente de reflexión medido en el puerto uno cuando tenemos conectado el *reflect* que será:

$$T_{MRX}=e_{11}+\frac{(e_{21}e_{12}T_R)}{(1-e_{22}T_R)} \tag{22}$$

donde T_R es el coeficiente de reflexión real del *reflect*. Operando con la ecuación obtenemos:

$$T_R=\frac{1}{e_{22}}\frac{(b-T_{MRX})}{(a-T_{MRX})} \tag{23}$$

De la misma manera buscamos y operamos la ecuación del coeficiente de reflexión en el puerto dos cuando el *reflect* está conectado:

$$T_{MRY}=f_{11}+\frac{(f_{21}f_{12}T_R)}{(1-f_{22}T_R)} \tag{24}$$

$$T_R=\frac{1}{f_{22}}\frac{(d+T_{MRY})}{(c+T_{MRY})} \tag{25}$$

Igualamos y operamos las dos ecuaciones obtenidas del coeficiente de reflexión real del *reflect*, (23) y (25), quedándonos:

$$\frac{1}{f_{22}}=\frac{1}{e_{22}}\frac{(b-T_{MRX})}{(a-T_{MRX})}\frac{(c+T_{MRY})}{(d+T_{MRY})} \tag{26}$$

Esta ecuación contiene dos incógnitas con lo que necesitamos otra ecuación para poder obtener los valores de e_{22} y f_{22}. Para ello buscaremos la ecuación del coeficiente de reflexión medido en el puerto uno cuando tenemos conectado el *thru*:

$$T_{MT}=e_{11}+\frac{(e_{21}e_{12}f_{22})}{(1-e_{22}f_{22})} \tag{27}$$

$$e_{22}=\frac{1}{f_{22}}\frac{(b-T_{MT})}{(a-T_{MT})} \tag{28}$$

Substituimos $\frac{1}{f_{22}}$ de (28) por (26) y aislamos e_{22}

$$e_{22}^2=\frac{(b-T_{MRX})}{(a-T_{MRX})}\frac{(c+T_{MRY})}{(d+T_{MRY})}\frac{(b-T_{MT})}{(a-T_{MT})} \tag{29}$$

Esta ecuación nos da dos posibles valores de e_{22} . Para elegir el valor correcto tendremos en cuenta que T_R tiene que ser 1 en caso de que el *reflect* sea un circuito abierto o -1 en el caso que sea cortocircuito. Por lo tanto substituiremos los dos valores de e_{22} obtenidos en la ecuación de T_R y escogeremos el valor para el cual el resultado sea lo más cercano a 1 o -1 según si hemos usado un circuito abierto o un cortocircuito como *reflect*. Una vez determinado el valor de e_{22} pasamos a buscar el valor de f_{22}

$$f_{22}=\frac{1}{e_{22}}\frac{(b-T_{MT})}{(a-T_{MT})} \tag{30}$$

A continuación buscamos la ecuación que define los parámetros R medidos de la línea de transmisión de la que queremos obtener los parámetros S:

$$R_M=\frac{1}{(e_{21}f_{12})}\begin{pmatrix}-\Delta_x & e_{11}\\ -e_{22} & 1\end{pmatrix}R_A\begin{pmatrix}-\Delta_y & f_{22}\\ -f_{11} & 1\end{pmatrix} \tag{31}$$

donde

$$\Delta_x=e_{11}e_{22}-e_{12}e_{21}=be_{22}-(b-a)e_{22}=ae_{22} \tag{32}$$

$$\Delta_y=f_{11}f_{22}-f_{12}f_{21}=-df_{22}-(c-d)f_{22}=-cf_{22} \tag{33}$$

Ahora sólo nos queda encontrar el valor de $e_{21}f_{12}$ para poder buscar los valores de la matriz R_A y lo hacemos mediante la ecuación del parámetro s_{21} medido del *thru*

$$s_{21\mathrm{MT}}=\frac{(e_{21}f_{12})}{(1-e_{22}f_{22})} \tag{34}$$

$$e_{21}f_{12}=s_{21\mathrm{MT}}(1-e_{22}f_{22}) \tag{35}$$

Una vez obtenido el valor del producto aislamos la matriz R_A para obtener los parámetros R de la línea

$$R_A=(e_{21}f_{12})\begin{pmatrix}-ae_{22} & b\\ -e_{22} & 1\end{pmatrix}^{-1}R_M\begin{pmatrix}cf_{22} & f_{22}\\ d & 1\end{pmatrix}^{-1} \tag{36}$$

2.3 Algoritmo de calibración TRL

A continuación presentaremos el algoritmo (mostrado en el apéndice) usado para implementar el método de calibración TRL en *matlab* para un matriz de parámetros S en función de la frecuencia.

El primer paso para ello será crear una función para comprobar que se hayan introducido en el *matlab* matrices con los parámetros S correspondientes al *thru,* al *reflect*, al *line* y al *sample* pues para la ejecución del programa necesitaremos de ellas. Estas matrices deberán introducirse bajo sus nombres correspondientes para el correcto funcionamiento del programa. Si no se introducen bajo

su nombre correcto o si simplemente se nos olvida de introducir alguna el programa enviará un mensaje de error especificando cual o cuales faltan.

Seguidamente el programa irá cogiendo los elementos de dichas matrices uno por uno y calculando los valores correctos de la medida. Para ello haremos un bucle de manera que el programa se ejecute *n* veces donde *n* es el número de frecuencias distintas que evaluaremos. Para cada ejecución tomará los valores de parámetros S correspondientes a la frecuencia en la que estemos.

Las operaciones que se ejecutarán dentro de este bucle serán los que indica el método de calibración descrito con anterioridad. De esta parte resaltaremos dos cosas: primero el valor que asignaremos a *a* y *b* así como a *c* y *d*, y segundo como elegiremos la raíz correcta de la ecuación de segundo grado que nos da e_{22} .

Para lo primero, si observamos el apartado anterior, vemos que para obtener las parejas de valores *a* y *b* así como *c* y *d* tenemos una misma ecuación de segundo grado para cada pareja. Para asignar los valores correctos a cada variable de las parejas debemos hacer una pequeña función que evalúe el valor absoluto de *a* y *b* asignando a *a* el valor cuyo módulo sea mayor y a *b* el valor cuyo módulo sea menor. Para la otra pareja de variables el procedimiento será el mismo.

Para obtener el valor correcto de e_{22} lo que haremos es lo siguiente. Primero guardaremos los dos posibles valores en variables intermedias. A continuación haremos una función que evalúe el valor que tomaría T_R en cada uno de los dos casos y escogeremos el e_{22} que nos de un valor más cercano a -1 que es el valor que teóricamente debería tener T_R . No hay ninguna función que compruebe directamente cual de los dos posibles de T_R es más próximo a -1 con lo que para comprobarlo lo que haremos es sumar uno a los dos posibles valores de T_R y quedarnos con el que tenga el valor absoluto menor mediante una comparación de ambos.

También resaltaremos que este algoritmo permite un punto de comprobación para que, en caso que no de los resultados esperados, tener una idea de donde se ha producido el error y no tener que revisar el programa entero. Esta comprobación la podemos hacer una vez obtenidos los valores *b* y *d* que son e_{11} y $-f_{11}$ respectivamente, es decir, son los valores correspondientes a los parámetros s_{11} (o $-s_{11}$) de las cajas de error A y B que podemos obtener mediante una simulación con el *puff* para poder comprobar si los pasos seguidos hasta ese momento en el programa de calibración son correctos.

Finalmente, una vez obtenida la matriz de los cuatro parámetros S correspondiente a la medida correcta, es decir, a la medida una vez calibrada, incluiremos en el programa una pequeña sección que nos grafique el módulo y el ángulo de s_{11} , s_{12} , s_{21} y s_{22} en función de la frecuencia. Como es posible que las gráficas presentadas no sean suficiente para el usuario o éste desee hacer otras complementarias, los valores correspondientes a la matriz de los parámetros S una vez ya corregido el error,se guardarán en una variable a la que el usuario podrá acceder para hacer las gráfica que desee o para hacer cálculos.

A continuación, en la figura 2.3, se muestra el diagrama de flujo correspondiente al algoritmo con el que ejecutaremos el método de calibración *thru-reflect-line*. De dicho diagrama de flujo es preciso comentar que las instrucciones s_x => r_x implican la transformación de parámetros S en parámetros R.

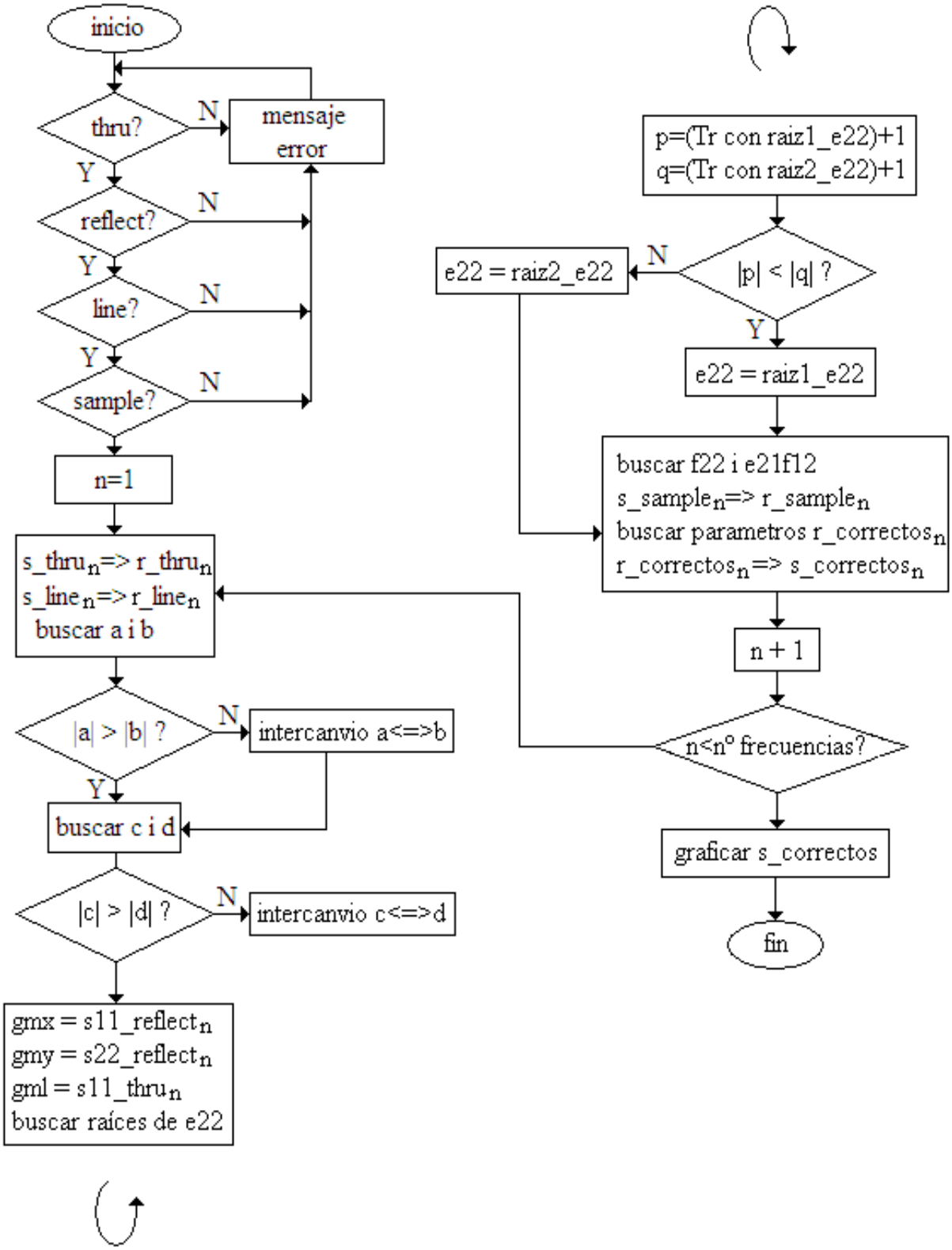


Figura 2.3 - Diagrama de bloques del programa TRL

2.4 Comprobación del algoritmo TRL mediante simulación

Para acabar este capítulo debemos comprobar el correcto funcionamiento del algoritmo al introducir los datos. Para ello tenemos que utilizar el programa *puff* , ya introducido previamente, que nos sirve por una banda, para simular una muestra de la que obtener los parámetros S, y por otra, para simular los errores introducidos por el analizador vectorial.

Así pues, las cajas de error A y B ilustradas en la figura 2.1, las simulamos como dos líneas de transmisión que añadimos a banda y banda de la muestra a medir. La muestra será también una línea de transmisión de distinta longitud e impedancia característica. A partir de aquí los pasos que seguiremos serán:

- Obtendremos los resultados mediante una simulación de los parámetros S correspondientes a la muestra más las dos cajas de error.
- Haremos otra simulación que nos de los parámetros S de las dos cajas de error conectadas directamente, con lo que obtendremos los datos del *thru*.
- Conseguiremos los datos del *reflect* haciendo una simulación en la que conectaremos las cajas de error A i B directamente a tierra.
- Buscaremos los parámetros S del *line* incluyendo en la simulación una línea de transmisión entre las cajas de error A i B.
- Finalmente haremos una simulación de la muestra sola para obtener los parámetros S correctos de dicha muestra.

Una vez realizados todos estos pasos con el *puff* y guardados los datos obtenidos ya podremos comprobar el correcto funcionamiento del programa creado. Para ello introduciremos los resultados obtenidos en los cuatro primeros apartados de los citados anteriormente (nos los pide el programa al ejecutarlo si no los tiene) y ejecutaremos el programa, el cual corregirá los errores mediante el método de calibración TRL.

La simulación que haremos para comprobar que el algoritmo funcione será del circuito que presentamos a continuación, en la figura 2.4, donde las cajas de error A y B están representadas por las dos líneas de transmisión de nombre *b* cuya impedancia característica es 40Ω y su longitud 100º. La muestra la representamos como dos líneas de transmisión distintas conectadas entre sí, una llamada *c* de impedancia 30Ω y longitud 70º, y la otra llamada *a* de impedancia 70Ω y longitud 40º. La frecuencia de diseño, es 2.8 Ghz.

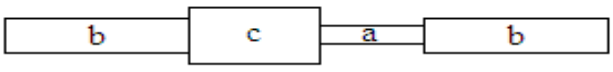


Figura 2.4 - Ejemplo usado para comprobar el funcionamiento del algoritmo

A continuación podemos ver los resultados obtenidos al aplicar todo lo explicado; en la figura 2.5 vemos los módulos superpuestos de s_{11} correspondientes a la simulación de la muestra sola, representado en color azul, y al resultado de ejecutar el programa sobre la muestra con las cajas de error representado en color rojo. La figura 2.6 corresponde a las fases superpuestas de lo mismo, siendo otra vez el azul para la simulación de la muestra sola y el rojo para los resultados de aplicar el algoritmo sobre los datos de la muestra con error.

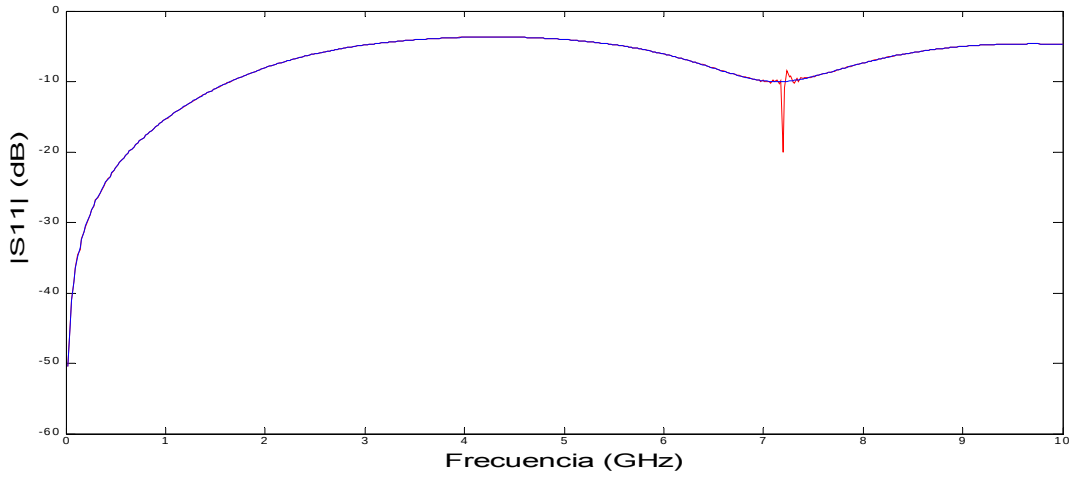


Figura 2.5 - Modulo de S11 para calibración TRL (rojo). En azul resultados de la muestra sola

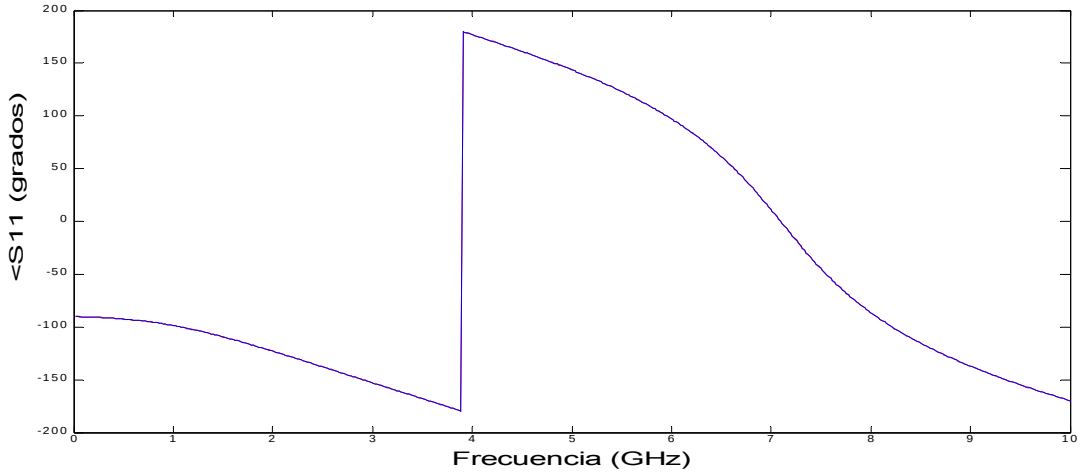


Figura 2.6 - Angulo de S11 para calibración TRL (rojo). En azul resultados para la muestra sola

Como podemos observar, en el caso del módulo, eligiendo las cajas de error que hemos elegido, el resultado de las simulación sin error y de la simulación con error que después corregimos con el método TRL, es prácticamente el mismo a excepción de ciertos pico entre los 7 y 8 GHz, lo que nos indica que el programa funciona correctamente aunque el método matemático de calibración TRL, tiene ciertas limitaciones. En el caso del ángulo, da exactamente igual motivo por el cual casi no se puede apreciar el rojo, salvo en algún puntito donde el azul no está completamente superpuesto.

Haremos una aclaración en este punto, y es que en este caso el error introducido es pequeño pues solo nos interesaba ver si el algoritmo realizaba la función para la que lo habíamos creado, pero tenemos que saber, que el analizador vectorial introduce más error y esto hace, que por las limitaciones propias del método usado, los resultados no sean tan satisfactorios, apareciendo más pico indeseados tanto en el módulo como en la fase como más adelante podemos comprobar.

3 CALIBRACION LRM

3.1 Introducción

El método de calibración LRM que usamos en nuestro proyecto fue desarrollado por Kimmo Silvonen, nacido en Koski, Finlandia, el 10 de octubre de 1957. Kimmo fue profesor de la Universidad de Tecnología Finlandia y en 1979 entró en el Laboratorio de Teoría de Circuitos de la misma universidad. Sus principales intereses fueron, a parte de la enseñanza, los procedimientos de calibración de analizadores vectoriales y la teoría de circuitos. Es gracias a esto y al conocimiento que adquirió sobre dichos temas que pudo desarrollar el método de calibación que explicaremos a continuación.

3.2 Método de calibración LRM (Line-Reflect-Match)

En este apartado explicaremos en que consiste el método de calibración LRM ideado por Kimmo Silvonen, siguiendo el método propuesto en [1]. Pero hay que decir que no entraremos en los procesos matemáticos que siguió para obtenerlo pues no es preciso para el desarrollo del proyecto.

Las medidas adicionales que tenemos que tomar para usar este método son: una línea de transmisión (*line*), otra acabada en cortocircuito o circuito abierto (*reflect*) y una impedancia adaptada (*match*). Seguidamente mostramos una imagen que representa lo que acabamos de explicar:

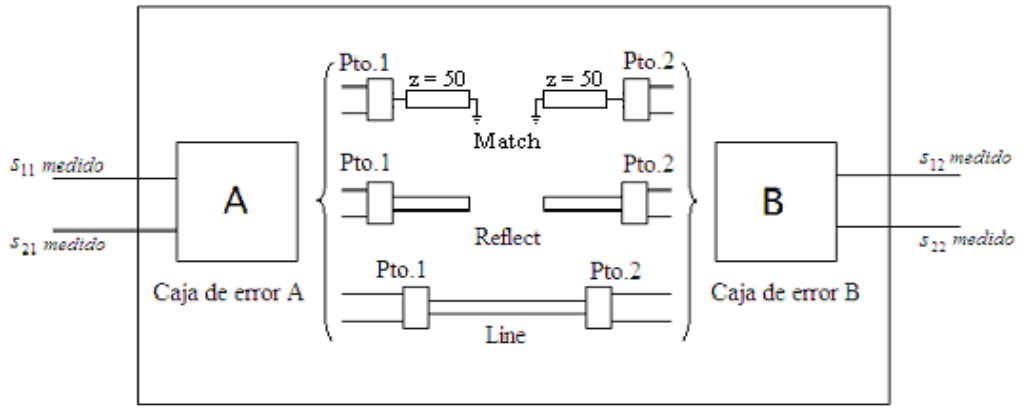


Figura 3.1 - Medidas a tomar para calibración LRM

Para empezar a explicar esta calibración partiremos de la figura que mostramos a continuación cuya notación es imprescindible para entenderlo.

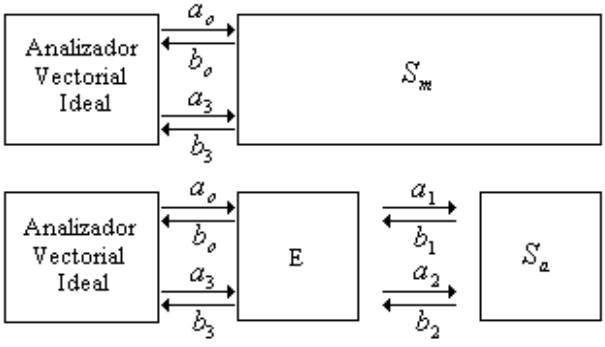


Figura 3.2 - Representación de la medición

Donde S_m representa los párametros S que se obtienen al medir la muestra (incluyendo el error), S_a los parámetros S reales de la muestra (sin error) y E la matriz de parámetros S equivalente a los errores que se producen en la medición. Finalmente, a_i y b_i son las ondas incidentes, reflectadas o transmitidas en las entradas y salidas de los terminales. Así podemos establecer para la matriz S_a :

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = S_a \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} s_{a11} & s_{a12} \\ s_{a21} & s_{a22} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{37}$$

Del mismo modo, para la matriz de los parámetros S con error, medidos por el analizador, tenemos:

$$\begin{bmatrix} b_0 \\ b_3 \end{bmatrix} = S_m \begin{bmatrix} a_0 \\ a_3 \end{bmatrix} = \begin{bmatrix} s_{m11} & s_{m12} \\ s_{m21} & s_{m22} \end{bmatrix} \begin{bmatrix} a_0 \\ a_3 \end{bmatrix} \tag{38}$$

Finalmente, para la matriz que corresponde a los parámetros R (ya vimos en el capítulo anterior que eran) de los errores introducidos en la medida:

$$\begin{bmatrix} b_0 \\ b_3 \\ \\ a_0 \\ a_3 \end{bmatrix} = \begin{bmatrix} T_1 & T_2 \\ T_3 & T_4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} t_0 & t_1 & t_4 & t_5 \\ t_2 & t_3 & t_6 & t_7 \\ & & & \\ t_8 & t_9 & t_{12} & t_{13} \\ t_{10} & t_{11} & t_{14} & t_{15} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \\ b_1 \\ b_2 \end{bmatrix} \tag{39}$$

Observamos que la matriz de errores en este caso la representamos de manera distinta a como lo hicimos para la calibración TRL (recordemos que para ello usamos dos matrices cuadradas dos por dos), esto se debe a que, para esta calibración, se parte de un modelo de error más complejo de 16 términos mientras que para la otra usábamos un modelo de error de 8 términos.

A partir de (39) obtenemos las siguientes ecuaciones:

$$\begin{bmatrix} b_0 \\ b_3 \end{bmatrix} = T_1 \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + T_2 \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{40}$$

$$\begin{bmatrix} a_0 \\ a_3 \end{bmatrix} = T_3 \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + T_4 \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{41}$$

Substituimos (37) en (40), operamos e igualamos a (38) y nos queda:

$$\begin{bmatrix} b_0 \\ b_3 \end{bmatrix} = (T_1 S_a + T_2) \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = S_m \begin{bmatrix} a_0 \\ a_3 \end{bmatrix} \tag{42}$$

Substituimos (37) en (41) y operamos:

$$\begin{bmatrix} a_0 \\ a_3 \end{bmatrix} = (T_3 S_a + T_4) \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{43}$$

Ahora solo nos queda substituir (43) en (42) y aislar S_a :

$$T_1 S_a + T_2 = S_m (T_3 S_a + T_4) \tag{44}$$

$$S_a = (T_1 - S_m T_3)^{-1} (S_m T_4 - T_2) \tag{45}$$

Una vez tenemos está ecuación nos queda encontrar los valores de las cuatro matrices dos por dos que conforman la matriz T del modelo de error de 16 términos. Para ello definiremos las matrices A, B y C como las matrices de los cuatro parámetros S obtenidos del *thru*, del *match* y del *reflect* respectivamente. A continuación tendremos que definir dos nuevas matrices que serán:

$$D = \begin{pmatrix} s_{11}\, reflect & 0 \\ 0 & s_{22}\, match \end{pmatrix} \tag{46}$$

$$E = \begin{pmatrix} s_{11}\, match & 0 \\ 0 & s_{22}\, reflect \end{pmatrix} \tag{47}$$

A partir de estas matrices obtenemos estas otras necesarias para la calibración:

$$M = (A - C) N \quad donde \quad N = (E - A)^{-1} (B - E) \tag{48}$$

$$P = (A - C) R \quad donde \quad R = (D - A)^{-1} (B - D) \tag{49}$$

$$O = B - C \tag{50}$$

Ahora, calculamos los coeficientes *m*, *n*, *o*, y *p* que son funciones de los elementos de las matrices que acabamos de definir.

$$m = (P_{21} + O_{21}) M_{22} - (P_{22} + O_{22}) M_{21} \tag{51}$$

$$n = O_{21} P_{12} - O_{22} P_{11} \tag{52}$$

$$o = (M_{12} + O_{12}) P_{11} - (M_{11} + O_{11}) P_{12} \tag{53}$$

$$p = O_{12} M_{21} - O_{11} M_{22} \tag{54}$$

A partir de aquí buscamos algunos de los parámetros de error que, a su vez, nos serán imprescindible para encontrar el resto. Para ello definimos el parámetro t_{12} con un valor igual a la unidad pues es un parámetro de escalado. A continuación definimos los otros tres parámetros pertenecientes a la matriz T_4 :

$$t_{15} = \sqrt{\frac{mo}{np}} \frac{P_{11}}{M_{22}} t_{12} \tag{55}$$

$$t_{14} = -\frac{M_{21}}{M_{22}} t_{12} \tag{56}$$

$$t_{13} = -\left(\frac{P_{12}}{P_{11}}\right) t_{15} \tag{57}$$

Pasamos ahora a buscar los parámetros correspondientes a la matriz T_3 , necesarios para la obtención de las matrices T_1 y T_2

$$t_8=(R_{11}t_{12}+R_{12}t_{14})\frac{1}{\Gamma}-t_{13}\frac{1}{T} \tag{58}$$

$$t_9=(N_{11}t_{13}+N_{12}t_{15})\frac{1}{\Gamma}-t_{12}\frac{1}{T} \tag{59}$$

$$t_{10}=(R_{21}t_{12}+R_{22}t_{14})\frac{1}{\Gamma}-t_{15}\frac{1}{T} \tag{60}$$

$$t_{11}=(N_{21}t_{13}+N_{22}t_{15})\frac{1}{\Gamma}-t_{14}\frac{1}{T} \tag{61}$$

Donde

$$T=-\Gamma\frac{p}{o}\frac{P_{11}t_{12}}{M_{22}t_{15}} \quad y \quad \Gamma=-1 \tag{62}$$

Finalmente encontramos el resto de valores que corresponden a los parámetros de error que nos faltan. Primero buscamos los de la matriz T_1 :

$$t_0=C_{11}t_8+C_{12}t_{10}-\frac{1}{\Gamma}(O_{11}t_{12}+O_{12}t_{14}) \tag{63}$$

$$t_1=C_{11}t_9+C_{12}t_{11}-\frac{1}{\Gamma}(O_{11}t_{13}+O_{12}t_{15}) \tag{64}$$

$$t_2=C_{21}t_8+C_{22}t_{10}-\frac{1}{\Gamma}(O_{21}t_{12}+O_{22}t_{14}) \tag{65}$$

$$t_3=C_{21}t_9+C_{22}t_{11}-\frac{1}{\Gamma}(O_{21}t_{13}+O_{22}t_{15}) \tag{66}$$

Ahora buscamos los de la matriz T_2 :

$$t_4=B_{11}t_{12}+B_{12}t_{14} \tag{67}$$

$$t_5=B_{11}t_{13}+B_{12}t_{15} \tag{68}$$

$$t_6=B_{21}t_{12}+B_{22}t_{14} \tag{69}$$

$$t_7=B_{21}t_{13}+B_{22}t_{15} \tag{70}$$

Una vez tenemos las cuatro matrices que forman la matriz de error T sólo nos queda substituir en (45) para, con los datos erróneos de la muestra obtenidos, encontrar los parámetros S correctos de dicha muestra.

Como ya advertimos aquí solo hemos explicado los pasos que se deben seguir para realizar la calibración LRM, pero no el proceso matemático por el cual fueron obtenidos pues dichos procesos son bastante complejos y no entran en el propósito del proyecto.

3.3 Algoritmo de calibración LRM

A continuación nos proponemos explicar como hemos desarrollado el algoritmo en *matlab* (apéndice 8.1.3) que permita implementar el método de calibración LRM guardando el resultado en una matriz que contenga los parámetros S de la muestra correcto en función de la frecuencia.

Al igual que para la calibración TRL, lo primero que incluimos en el algoritmo es una función que compruebe si hemos introducido los datos necesarios bajo su nombre correspondiente, pues sin ello no se puede seguir ejecutando el programa. Así que en el caso de que falte el *line*, el *reflect* o el *match* no podremos seguir avanzando y el programa nos dará un mensaje de error mostrando que es lo que falta.

A continuación creamos un bucle como el usado en el apartado 2.3 para que el programa empiece a buscar los parámetros S correspondientes a la primera frecuencia y, una vez encontrados, realice lo mismo con los datos de la segunda frecuencia y así, iterativamente, hasta llegar a la última.

Así pues dentro de este bucle tenemos que ejecutar todos los pasos mostrados en el apartado anterior. En este caso, resulta cualitativamente más sencillo que para el método TRL, pues no tenemos que crear funciones que decidan elegir un resultado u otro, como nos pasaba al buscar los valores *a* y *b*, *c* y *d*, y e_{22} . Lo que tenemos que hacer es coger los datos introducidos por el usuario y guardarlos bajo los nombre de A, B, C, D y E según corresponda. Una vez hecho esto escribimos todas las ecuaciones a partir de la (48) hasta la (70), eso sí, tenemos que vigilar como trata *matlab* las matrices para que el programa funcione correctamente.

Una vez hecho esto guardamos todos los parámetros de error ($t_{0,}t_1 ... t_{15}$) en sus correspondientes matrices T_1, T_2, T_3 y T_4 , e introducimos la ecuación (45) de manera que, con los datos erróneos de la muestra correctamente guardados, se encargará de corregirlos.

Finalmente, antes de saltar al inicio del bucle (o salir de él en el caso que ya haya calculado los datos para todas las frecuencias), solo tenemos que guardar los resultados obtenidos en la fila correspondiente de la matriz usada con este fin. Para ello introducimos una variable, iniciada en uno, que nos indicará en que fila tenemos que guardarlos y que se irá incrementado en uno cada vez que recorramos el bucle.

Para acabar el programa, añadimos ciertos elementos que permitan tener una noción rápida al usuario respecto a que éste se ha ejecutado correctamente. Para ello introducimos una función que avise cuando el programa haya terminado de ejecutarse y también incluimos una sección al final del mismo para que nos muestre gráficamente el resultado tanto de la fase y el módulo de los cuatro parámetros S en función de la frecuencia. Añadir, que si el usuario quiere hacer otros gráficos o cálculos, tiene todos los datos numéricos almacenados en una matriz.

De este algoritmo queremos destacar dos cosas. Primero que la implementación del mismo, como ya hemos dicho, resulta más sencilla que para el caso TRL. En segundo lugar, recalcar que no tiene ningún punto intermedio en el que podamos comprobar si vamos por el camino adecuado, pero esto tampoco supone un gran contratiempo por la primera de las cosas que hemos destacado.

A continuación presentamos un diagrama de flujo correspondiente a los pasos que hemos seguido para la implementación de este algoritmo ya explicado en este apartado. El código del programa, tal como lo hemos escrito en *matlab*, lo incluimos en el anexo.

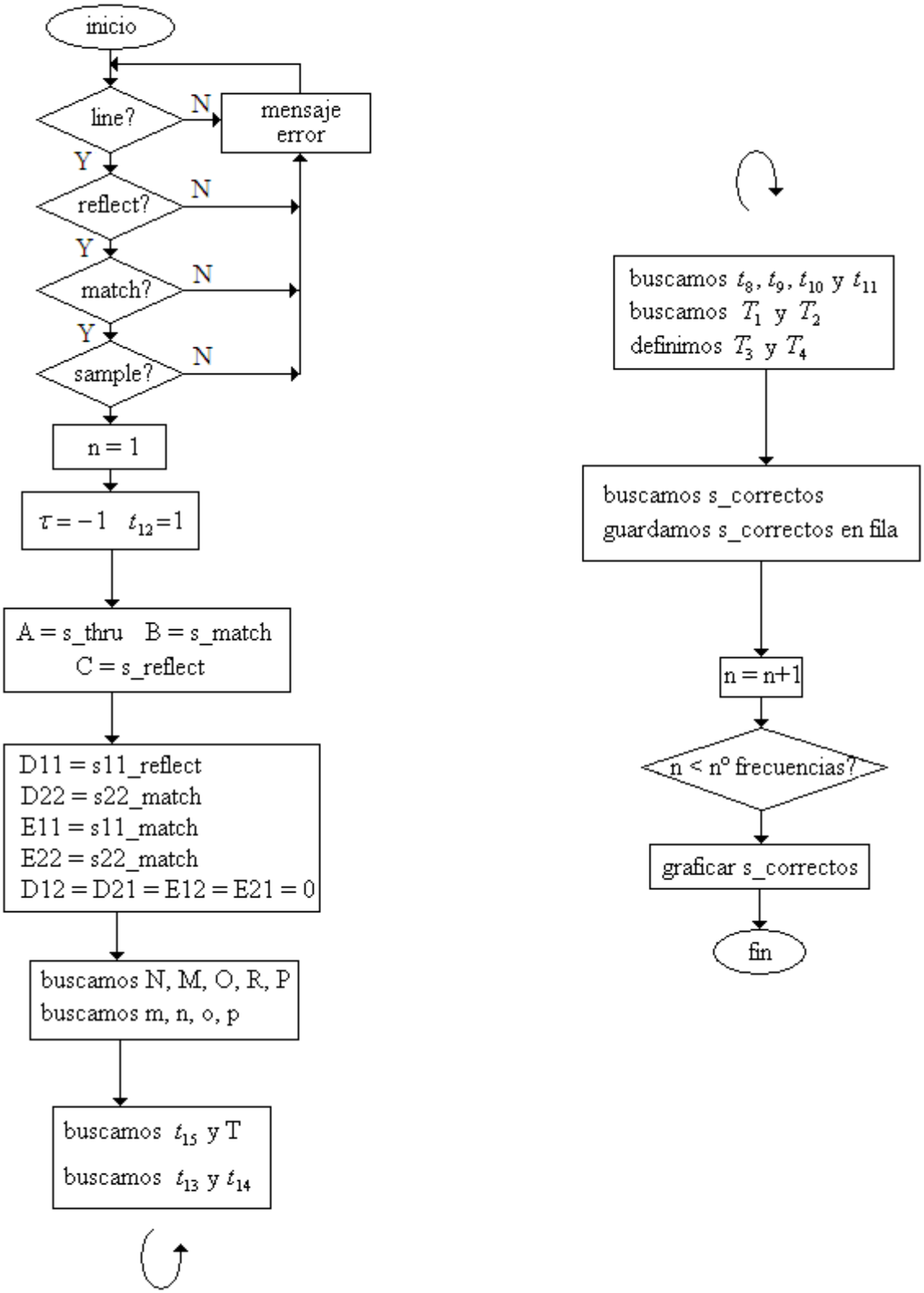


Figura 3.3 - Diagrama de bloques del programa LRM

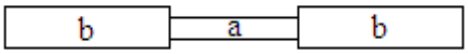
3.4 Comprobación del algoritmo LRM mediante simulación

Una vez tenemos implementado el programa para la calibración LRM, nos queda comprobar su funcionamiento mediante la simulación. Para esto, como ya hicimos en el segundo apartado, usaremos el programa *puff* con el que obtendremos los parámetros S correctos de la muestra escogida y, posteriormente, con el mismo programa, le añadiremos líneas de transmisión a la muestra a modo de simulación de los errores que se producen en la medición real, y obtendremos los parámetros S de lo que sería la medición de la muestra con error.

Así pues, los pasos a seguir, de manera análoga a como ya hicimos, serán los que mostramos a continuación:

- Obtenemos los parámetros S correctos de la muestra.
- Añadimos el error a la muestra y guardamos los resultados obtenidos.
- Conseguimos los datos de *line* substituyendo al modelo de la muestra con error, la muestra por una línea de transmisión.
- Los parámetros del *reflect* los obtenemos conectando a masa las líneas de transmisión usadas para añadir error y midiendo sus parámetros S.
- Finalmente, los datos del *match* los medimos conectando una impedancia adaptada entre las masa y las líneas de transmisión del paso anterior.

Una vez hecho esto ya podemos comprobar el correcto funcionamiento del algoritmos y para ello usamos el circuito de la figura 3.4 donde *a* es la muestra a medir y es una línea de transmisión de 50Ω y 70º, y las líneas de transmisión *b* que hay a banda y banda de la línea d transmisión *a,* representan los errores del sistema y son de 40Ω y 100º. Todo para una frecuencia de diseño de 2.8 GHz.



3.4 Ejemplo para la comprobación del algoritmo

A continuación mostramos los resultados obtenidos tanto para el módulo como para la fase en las figuras siguientes donde usamos el azul para los resultados de la muestra sola y el rojo para los resultados de aplicar el algoritmo de calibración sobre la muestra con error. Comentar que los resultados mostrados son sólo los correspondientes a s_{11} pues con ello ya es suficiente para nuestro propósito.

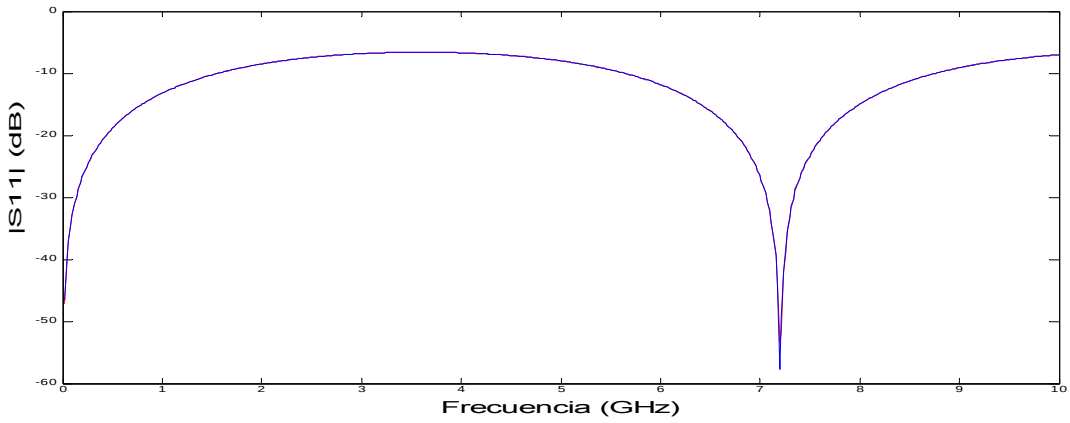


Figura 3.5 - Módulo de S11 para calibración LRM (rojo). En azul resultados para la muestra sola

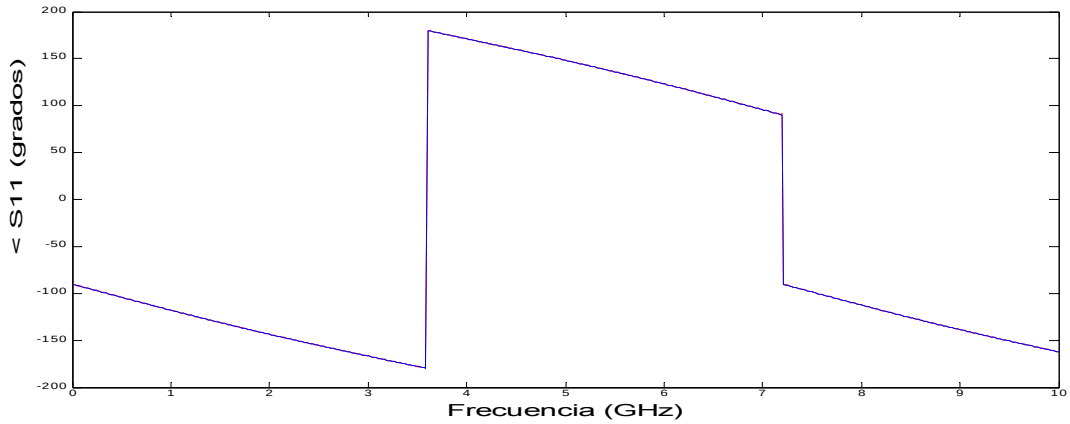


Figura 3.6 – Fase de S11 para calibración LRM (rojo). En azul resultados para la muestra sola

Como vemos, al tratarse de un ejemplo sencillo, donde el error introducido no es muy grande, el método se ajusta a la perfección haciendo que el color azul se superponga al rojo y es por esto que no lo podemos ver. Esto nos supone un buen síntoma, pues implica que hemos desarrollado bien el algoritmo en *matlab.* Como veremos más adelante, para ejemplos más complejos, los resultados no son tan coincidentes pues aunque el método es bueno, tiene sus limitaciones.

4 COMPARACIÓN ENTRE LA CALIBRACIÓN TRL Y LA LRM

Una vez implementados los algoritmos en *matlab* correspondientes a los métodos de calibración TRL y LRM nos proponemos ha hacer un apartado en el que, mediante diversas medidas, podremos ver cual es la respuesta de ambos con el fin de poder establecer una comparación que nos lleve a determinar cual es el método mejor según nuestras necesidades. Para ello haremos diversas simulaciones con el programa *puff* en el que introduciremos distintas líneas de transmisión a medir y en el que simularemos los errores introducidos por el analizador de espectros mediante líneas de transmisión incorporadas a ambos bandos de la muestra a medir.

4.1 Picos de error

Empezaremos con un circuito como el presentado en la figura 4.1, en el que podemos ver todas las medidas que tomaremos para hacer las dos calibraciones y poderlas comparar. También se incluye la imagen de la muestra sola pues la usaremos para comparar las dos calibraciones, a parte de entre ellas mismas, con el resultado que deberíamos obtener.

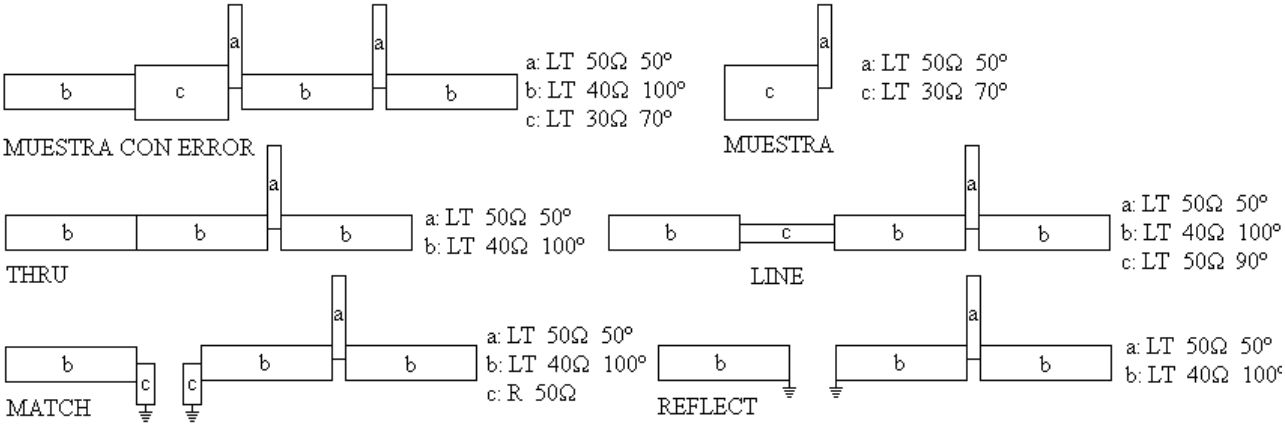


Figura 4.1 - Medidas a tomar para las dos calibraciones a una frecuencia de diseño de 2.8 GHz

A continuación representaremos gráficamente los resultados del módulo y la fase de s_{11} tanto para la simulación de la muestra sin las cajas de error (negro) como de los resultados que hemos obtenido cogiendo los datos de la muestra con errores y corrigiéndolos usando las calibraciones TRL (rojo) y LRM (azul).

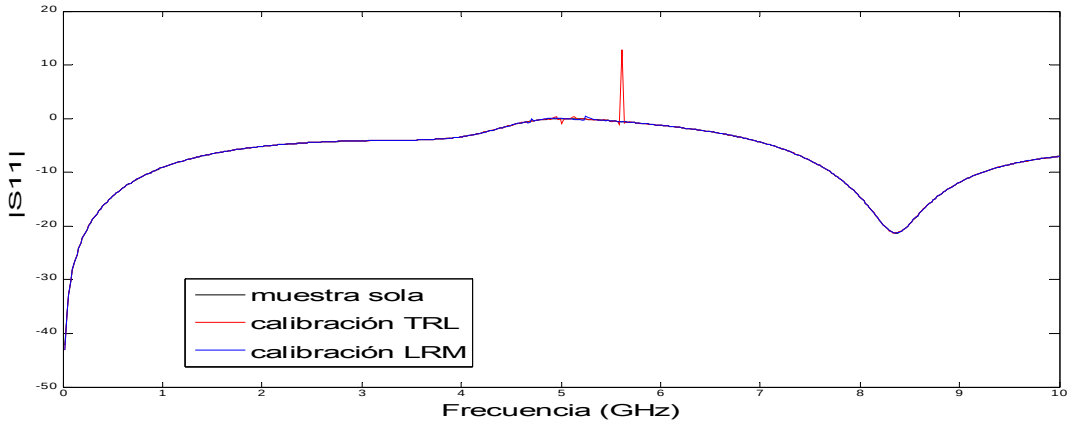


Figura 4.2 - Módulo de S11, primer caso, correspondiente al circuito de la figura 4.1

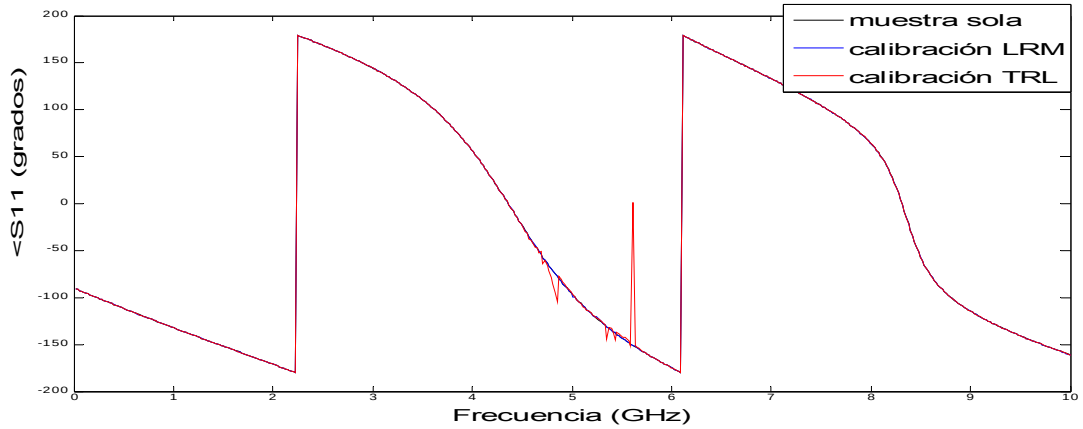


Figura 4.3 - Fase S11, primer caso, correspondiente al circuito de la figura 4.1

Como podemos ver, tanto para la fase como para el módulo, los dos métodos de calibración nos dan unos resultados muy precisos, pero en el caso de la calibración TRL, existe la aparición de unos picos indeseados, más notables en la fase que en el módulo. Así pues, aunque los dos métodos de calibración son buenos, podemos concluir que el método LRM es mejor.

4.2 Error introducido por los dos sistemas de calibración

En este apartado veremos el error que introduce cada uno de los dos sistemas de calibración. Para ello mostraremos las gráficas de la resta entre los parámetros de la muestra y la calibración TRL, y entre los parámetros de la muestra y la calibración LRM. Usaremos nuevamente el circuito de la figura 4.1. A continuación vemos las gráficas resultantes de aplicar lo que acabamos de decir tanto para el módulo como para la fase.

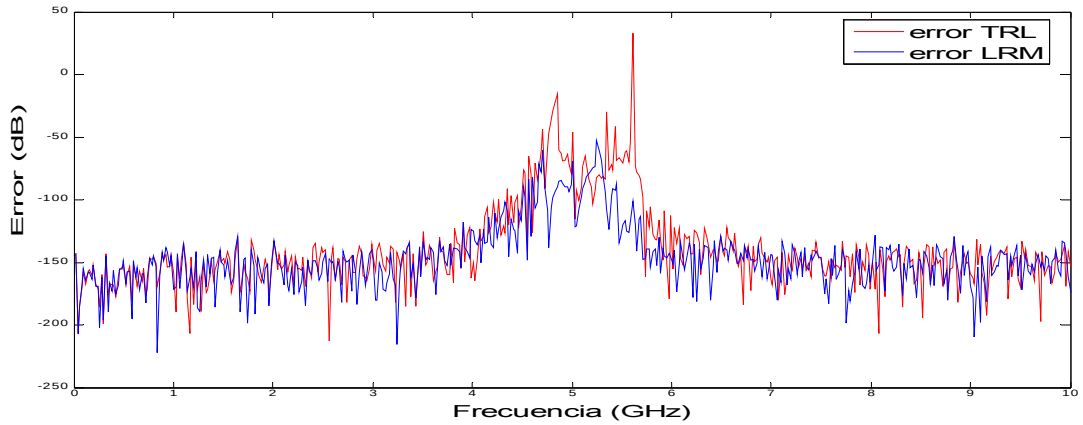


Figura 4.4 - Valor absoluto en dB de la resta entre la muestra y la calibración TRL, y la muestra y la calibración LRM para el circuito de la figura 4.1

Como podemos observar, el error introducido por las calibraciones es un error muy pequeño motivo por el cual en la gráfica 4.2 los resultados obtenidos se superponían al de la gráfica de los parámetros de la muestra sin error. Hay una excepción que es entre los 4 y 6 GHz, donde el error aumenta, sobretodo en el caso de la calibración TRL, motivo por el cual podemos ver esos picos indeseados en las figura 4.2.

5 COMPROBACIÓN DEL ALGORITMO TRL CON MEDIDAS REALES

Una vez comprobado el correcto funcionamiento de los algoritmos de calibración mediante simulaciones, nos proponemos a examinar el comportamiento del algoritmo TRL en una aplicación real (ver programa en apéndice 8.1.1)

Como ya comentamos en la introducción, el modelo del analizador vectorial del que disponemos actualmente, permite distintos modos de calibración a diferencia del modelo más antiguo que no nos daba tal facilidad. Gracias a ello podremos calibrar, mediante el mismo analizador vectorial, la muestra que estamos midiendo para obtener los resultados correctos y así poder compararlos con los valores que hemos obtenido con nuestra calibración, hecha a través de *matlab*. Para ello realizamos los siguientes pasos:

- Primero cogemos una muestra de PTFE (apéndice figura 8.1), más conocido como teflón, y, utilizando el analizador vectorial, obtenemos los parámetros S. Estos parámetros contendrán errores que nuestro programa debe corregir.
- A continuación, seleccionamos la opción de calibrar el analizador mediante el método TRL. Siguiendo las instrucciones que el mismo analizador vectorial nos indica y mediante el *kit* de calibración, tomamos las medidas que corresponden a los parámetros S del *thru*, del *reflect* y del *line*.
- Seguidamente, tomamos medidas de la muestra una vez calibrado el analizador vectorial para que nos de los resultados correctos.
- Para acabar introducimos los valores de los parámetros S correspondientes a la muestra sin los errores corregidos junto con los valores del *thru*, del *reflect* y del *line*. Una vez que los hemos introducido ejecutamos el programa. Se tiene que resaltar que el formato con el con son guardados los datos desde el analizador (.s2p) da problemas al abrirlo con *matlab* pues no reconoce los números como tal. Así pues, lo que hacemos es abrirlos desde *excel* y los guardamos como hoja de cálculos de *excel*. Para importar los datos guardados de esta manera *matlab* no presenta ningún problema, solo tenemos que tener en cuenta que las primeras filas no contienen datos numéricos, pero el programa que hemos hecho ya tiene en cuenta eso.

En la figura 5.1 que adjuntamos a continuación, podemos ver la gráfica de los valores de la muestra junto con el error introducido por el analizador vectorial tanto del módulo como del ángulo de s_{11} , expresados en dB y grados respectivamente. Adjuntamos esta imagen para que se pueda apreciar cuan grande es el error que nos introduce el analizador vectorial.

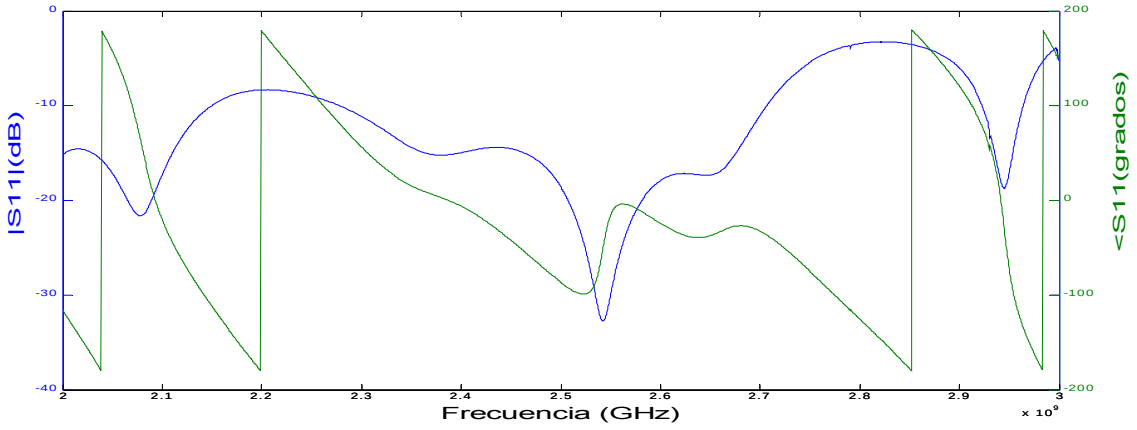


Figura 5.1 – Módulo y fase de S11 sin calibrar

A continuación, en las figuras 5.2 y 5.3, podemos observar los valores superpuestos del módulo y la fase de s_{11} después de la calibración con el programa hecho en *matlab* (azul), con el programa que lleva incorporado el analizador de espectros (rojo) y los datos correctos (negro)

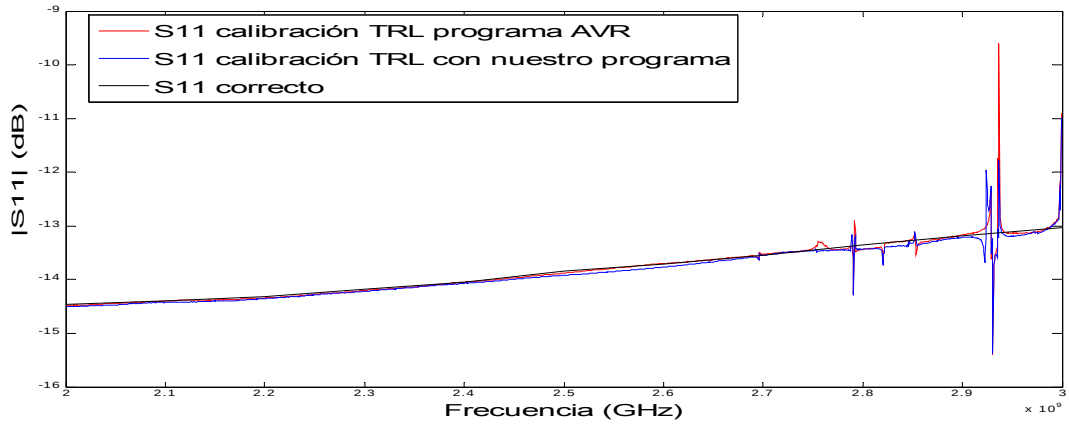


Figura 5.2 - Módulo de S11 de la muestra PTFE usada para comprobar el algoritmo

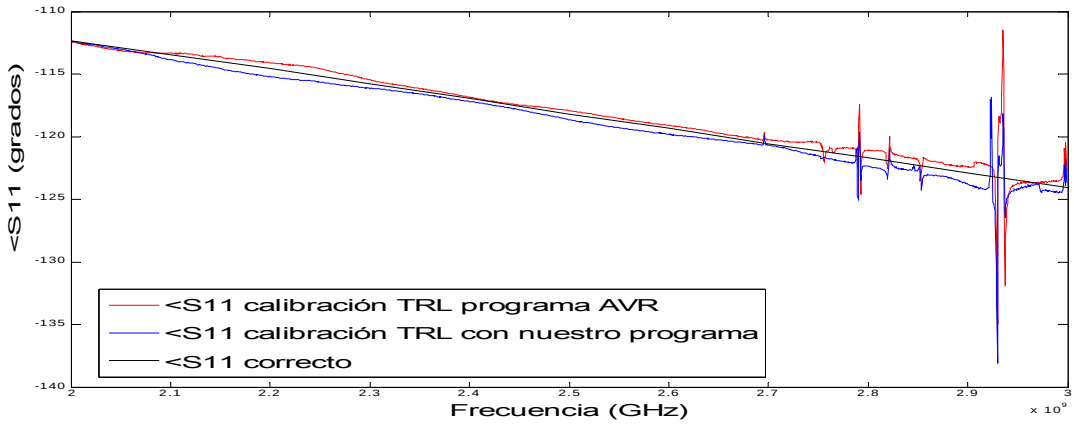


Figura 5.3 - Fase de S11 de la muestra PTFE usada para comprobar el algoritmo

A continuación repetimos el procedimiento pero esta vez para los valores correspondientes a s_{21} . El resultado se muestra en las figuras 5.4 y 5.5 donde el resultado de la calibración del analizador vectorial está en rojo y el resultado de la calibración de nuestro programa está azul.

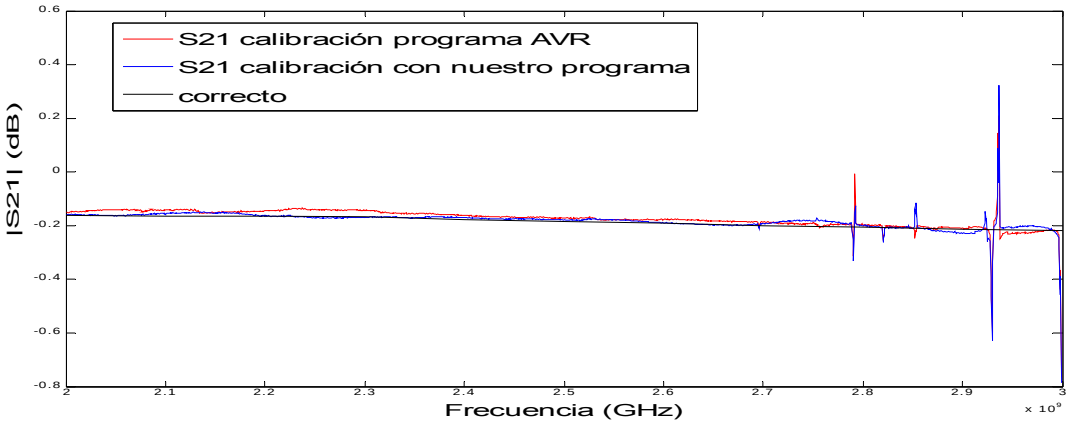


Figura 5.4 - Módulo S21 de la muestra PTFE usada para comprobar el algoritmo

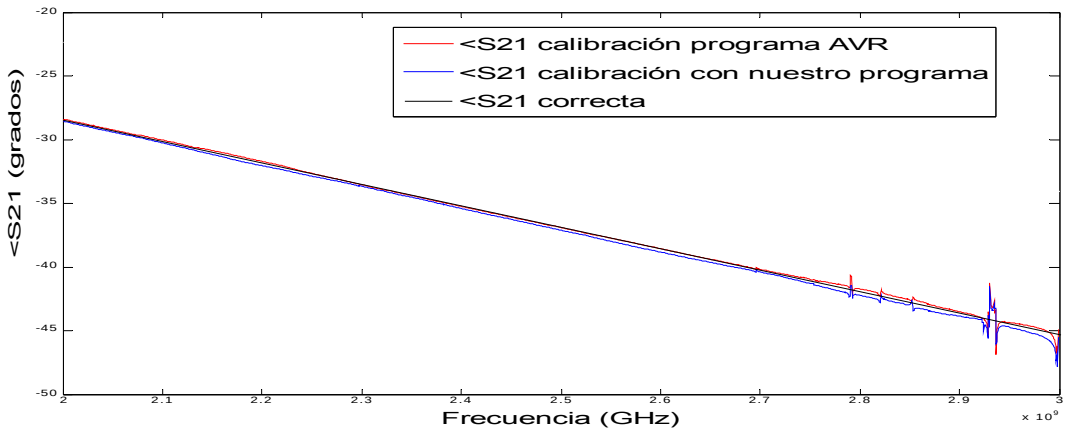


Figura 2.5 - Fase de S21 de la muestra PTFE usada para comprobar el algoritmo

Como podemos comprobar, los resultados obtenidos son satisfactorios, pues los parámetros S obtenidos mediante la calibración del analizador y los obtenidos mediante nuestro programa con las medidas de analizador son prácticamente iguales.

6 CONCLUSIONES

Para acabar este proyecto vamos a hablar de la experiencia del mismo así como de la experiencia que a mí me ha aportado y de las conclusiones a la que se ha llegado mediante su realización así como de las utilidades del mismo.

6.1 A nivel general

La primera conclusión que podemos obtener de este proyecto a nivel general, es que, una vez probados los algoritmos TRL y LRM a nivel de simulación para distintos casos, podemos decir que funcionan correctamente. De la misma manera, tras la realización del apartado 5, podemos decir que el programa del método TRL también funciona correctamente. Finalmente, por el hecho de que el programa para la calibración TRL, una vez implementado y comprobado a nivel de simulación, nos dio buenos resultados a nivel práctico haciendo leves modificaciones concernientes sólo a la manera en que el programa coge los datos, creemos que si hiciéramos las mismas modificaciones en el programa para la calibración LRM, éste funcionaría también.

Gracias al apartado 4, podemos extraer otra conclusión y es que, a nivel comparativo, el método de calibración LRM ofrece mejores resultados que la calibración TRL, pues aunque los dos dan buenos resultados, el método LRM se muestra más robusto ante la aparición de picos indeseados.

Respecto a los picos que aparecen al hacer la calibración TRL, hemos observado que aparecen cuando las medidas de todos los elementos para hacer la calibración, tienden a un módulo de 0dB. Pero parece que también hay otras causas que no hemos podido determinar para la aparición de dichos pico, porque, aunque siempre que todos los módulos de los parámetros usados para la calibración son cercanos a los 0dB aparecen picos en el resultado de la calibración, también aparecen picos en otros momentos en los que esto no se cumple. En la imagen siguiente mostramos gráficamente un ejemplo de lo que acabamos de explicar.

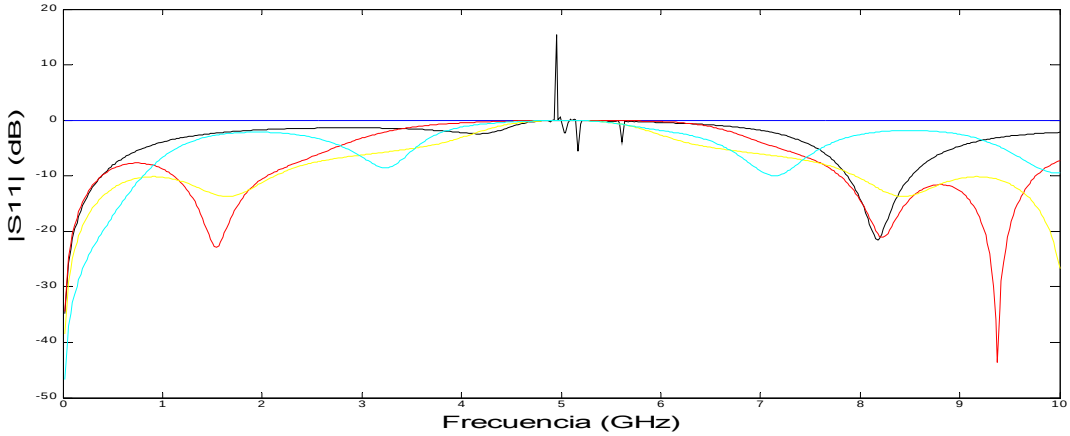


Figura 5.1 - Ejemplo de pico indeseado

En este ejemplo el negro pertenece a la muestra calibrada, el azul al *reflect*, el amarilloal *thru*, el cyan al *line* y el rojo a la muestra con error. Podemos ver que cuando los cuatro elementos de la calibración (*thru*, *reflect*, *line* y la muetra con error) cogen valores de aproximadamente 0 dB, surgen los picos indeseados.

6.2 A nivel personal

Este proyecto ha presentado ciertas dificultades a lo largo de su realización. La primera de ellas fue la de familiarizarme con los dos programas que debía usar. Es cierto que dichos programas no eran nuevos para mí pues a lo largo de la carrera los había utilizado pero no había profundizado tanto como lo que tuve que hacerlo para desarrollar el proyecto. Respecto al *puff* no tuve mucha dificultad pues, a parte de haberlo usado ya, es un programa más bien sencillo cuya principal dificultad consiste en conocer unos cuantos comandos que se pueden encontrar en el manual del programa. Con *matlab* la cosa cambió pues tiene muchos comandos, pero hay que decir que gracias al uso anterior, gracias al código de programación que usa (muy similar al lenguaje C) y gracias a un gran manual que encontré sólo fue cuestión de dedicarle tiempo.

Otra dificultad que entrañó el proyecto fue el entender en que consistían exactamente los métodos de calibración TRL y LRM, pues era algo básico para después poder implementar los algoritmos que los desarrollasen de forma automatizada. Una vez conseguido esto empecé a escribir el programa en *matlab* correspondiente al método de calibración TRL. Este método tenía la gran ventaja, como ya comentamos en el segundo apartado del proyecto, de tener de forma natural un punto de test, para comprobar si hasta ese momento el planteamiento había sido el adecuado. El desarrollo del algoritmo hasta ese punto fue algo lento y tedioso pues daba errores y costaba encontrarlos. Solucionados los problemas hasta allí, la cosa mejoró y pude realizar el resto del algoritmo de una manera más rápida pues a costa de errar una y otra vez había mejorado mis conocimientos de *matlab* y entendía mejor el método de calibración, dos cosas que eran las principales fuentes de error. Una vez hecho el programa entero añadí ciertas funciones que le diesen más profesionalidad como la comprobación previa a la ejecución del programa que nos indica si faltan datos por importar y las gráficas que nos muestra para que podamos ver simple vista si el programa nos da los resultados esperados. Completado el programa de calibración TRL para datos provenientes del *puff* solo hizo falta hacer pequeños retoques para realizar unas segunda versión para datos provenientes del analizador vectorial. La realización del algoritmo LRM fue mucho más rápida y satisfactoria a raíz de la experiencia adquirida.

Después de la realización del algoritmo TRL y antes de la realización del algoritmo LRM, pude comprobar el funcionamiento del primero con medidas reales. Para ello realizamos, junto con el tutor y un experto, medidas reales con el analizador vectorial, del cual pude ver su funcionamiento.

En cuanto la aportación a mi formación personal debida a la realización de este proyecto de final de carrera me gustaría valorar:

- El conocimiento adquirido de los métodos de calibración en general y, en particular y de una manera mucho más profunda el conocimiento, comprensión y asimilación de los métodos TRL y LRM.
- La profundización en programas ya usados como estudiante cuyo conocimiento era superficial o estaba relegada al olvido. De los dos programas usado quiero destacar el *matlab* pues me parece una herramienta cuyo conocimiento puede ser más útil al ser un programa de propósito más general y que puede ser usado en todas las vertientes de la carrera.
- La ampliación de lo aprendido hasta ahora en la carrera con respecto a las líneas de transmisión, parámetros S y el analizador vectorial de redes.

7 BIBLIOGRAFÍA

[1] Kimmo Silvonen. “LMR 16 – A Self-Calibration Procedure for a Leaky Network Analyzer”. IEEE Transactions on Microwave Theory and Techniques, Vol. 45, Nº 7, July 1997

[2] Susana Padilla Corral e Israel García Ruiz.”Método de Corrección de Errores del Analizador de Redes Empleando Como Referencia las Líneas Coaxiales Patrón”. Simposio de Metrología 2004

[3] M.L. Edwards. “MICROWAVE & RF CIRCUITS: Analysis, Design, Fabrication & Measurement”

[4] Glenn F. Engen and Cletus A. Hoer. “Thru-Reflect-Line: An Improved Technique for Calibrating the Dual Six-Port Automatic Network Analizer”. IEEE Transactions Microwave Theory and Techniques, Vol. MIT-27, Nº 12, December 1979

[5] Pedro de Paco. “Apuntes Radiación y Ondas Guiadas”. Departamento de Telecomunicaciones y de Inegniería de Sistemas

[6] Andrew Davidson, Eric Strid, Keith Jones. “Achieving Greater On-Wafe S-Parameters Accuracy with LRM Calibration Technique”.

[7] Javier García de Jalón, José Igancio Rodríguez, Jesús Vidal. “Aprenda Matlab 7.0 Como Si Estuviera en Primero”

8 APÈNDICE

8.1 Códigos de los programas

8.1.1 Programa del método TRL para el analizador vectorial de redes

```
%Miramos que se hayan introducido los datos correctamente
m=exist('THRU');
if m~=1
    disp('Error: no ha introducido los parametros s del THRU')
    return
end
m=exist('REFLECT');
if m~=1
    disp('Error: no ha introducido los parametros s del REFLECT')
    return
end
m=exist('LINE');
if m~=1
    disp('Error: no ha introducido los parametros s del LINE')
    return
end
m=exist('SAMPLE');
if m~=1
    disp('Error: no ha introducido los parametros s de la linea medida x')
    return
end
%iniciamos el bucle que ira calculando los parámetros S para cada
%frecuencia
comptador=5;
n=1;
while n<1600
%parametros s del thru
th11=THRU(comptador,2)+THRU(comptador,3)*i;
th22=THRU(comptador,8)+THRU(comptador,9)*i;
th12=THRU(comptador,6)+THRU(comptador,7)*i;
th21=THRU(comptador,4)+THRU(comptador,5)*i;
%definimos la matriz TH correspondiente a los param. R del thru
At=th11*th22-th21*th12;
TH=(1/th21)*[-At th11;-th22 1];
%parametros s del line
d11=LINE(comptador,2)+LINE(comptador,3)*i;
d12=LINE(comptador,6)+LINE(comptador,7)*i;
d21=LINE(comptador,4)+LINE(comptador,5)*i;
d22=LINE(comptador,8)+LINE(comptador,9)*i;
%definimos la matriz D correspondiente a los param. R del line
Ad=d11*d22-d21*d12;
D=(1/d21)*[-Ad d11;-d22 1];
%definimos la matriz T=D*inv(TH)
T=D*inv(TH);
%escribimos ec. de segundo grado y buscamos a=x11/x22 i b=x12/x22=e11
pol1=[T(2,1) (T(2,2)-T(1,1)) (-T(1,2) )];
B=roots(pol1);
a=B(1);
b=B(2);
%miramos si el modulo de a es mayor que el de b, si lo es OK sino
%intercambiamos valores
if abs(a)< abs(b)
    p=b;
    b=a;
```

```
    a=p;
end
%Definimos la matriz N=inv(TH)*D
N=inv(TH)*D;
%buscamos c=y11/y12 i d=y21/y22=-f11 a partir de N
pol2=[N(1,2) (N(2,2)-N(1,1)) (-N(2,1) )];
C=roots(pol2);
c=C(1);
d=C(2);
%miramos si el modulo de c es mayor que el de d, si lo es OK sino
%intercambiamos valores
if abs(c)< abs(d)
    q=d;
    d=c;
    c=q;
end
%definimos coeficientes de reflexion a la entrada de los dos reflect
gmx=REFLECT(comptador,2)+REFLECT(comptador,3)*i;
gmy=REFLECT(comptador,8)+REFLECT(comptador,9)*i;
%definimos coeficiente de reflexion del thru
gml=th11;
%buscamos los dos posibles valores de e22
e22a= sqrt(((b-gmx)/(a-gmx))*((c+gmy)/(d+gmy))*((b-gml)/(a-gml)));
e22b=-sqrt(((b-gmx)/(a-gmx))*((c+gmy)/(d+gmy))*((b-gml)/(a-gml)));
%miramos cual de los dos es correcto
gra=(1/e22a)*((b-gmx)/(a-gmx))+1;
grb=(1/e22b)*((b-gmx)/(a-gmx))+1;
if abs(gra)< abs(grb)
    e22=e22a;
else
    e22=e22b;
end
%encontramos los valores de f22, e21*e12, f21*f12, e21*f12 i f21*e12
f22=(1/e22)*((b-gml)/(a-gml));
e21e12=(b-a)*e22;
f21f12=(c-d)*f22;
e21f12=th21*(1-(e22*f22));
f21e12=th12*(1-(e22*f22));
%definimos parametros s medidos de la muestra
rm11=SAMPLE(comptador,2)+SAMPLE(comptador,3)*i;
rm12=SAMPLE(comptador,6)+SAMPLE(comptador,7)*i;
rm21=SAMPLE(comptador,4)+SAMPLE(comptador,5)*i;
rm22=SAMPLE(comptador,8)+SAMPLE(comptador,9)*i;
%definimos la matriz de parametros R de la muestra
Arm=rm11*rm22-rm21*rm12;
RM=(1/rm21)*[-Arm rm11;-rm22 1];
%obtenemos L y M
L=[(-a*e22) b;-e22 1];
M=[(c*f22) f22;d 1];
%gracias a L y M obtenemos la matriz de parametros R correctos de muestra
RA=e21f12*inv(L)*RM/M;
%a partir de los parametros R buscamos los parametros S correctos
s21=1/RA(2,2);
s11=RA(1,2)/RA(2,2);
s22=-RA(2,1)/RA(2,2);
s12=(RA(1,1)*s21+s11*s22)/s21;
%guardamos en f el valor de la frecuencia a la que estamos trabajando
f=SAMPLE(comptador,1);
%guardamos en la linea correspondiente de la matriz los datos correctos y
%la frecuencia a la que pertenecen
S(n,1)=f;
```

```
S(n,2)=s11;
S(n,3)=s12;
S(n,4)=s21;
S(n,5)=s22;
n=n+1;
comptador=comptador+1;
end
%mensajes que advierten que ha acabado el programa
disp('El programa ha finalizado correctamente')
disp('Los parametros s obtenidos se encuentran guardados en la matriz S')
%hacemos unas graficas a modo de muestra
subplot(2,2,1)
H=plotyy(S(:,1),20*log(abs(S(:,2))),S(:,1),angle(S(:,2))*180/pi)
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S11|(dB)')
set(get(H(2),'Ylabel'),'String','<S11(grados)')
subplot(2,2,2)
H=plotyy(S(:,1),20*log(abs(S(:,3))),S(:,1),angle(S(:,3))*180/pi)
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S12|(dB)')
set(get(H(2),'Ylabel'),'String','<S12(grados)')
subplot(2,2,3)
H=plotyy(S(:,1),20*log(abs(S(:,4))),S(:,1),angle(S(:,4))*180/pi)
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S21|(dB)')
set(get(H(2),'Ylabel'),'String','<S21(grados)')
subplot(2,2,4)
H=plotyy(S(:,1),20*log(abs(S(:,5))),S(:,1),angle(S(:,5))*180/pi)
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S11|(dB)')
set(get(H(2),'Ylabel'),'String','<S11(grados)')
```

8.1.2 Programa del métdodo TRL para puff

```
%Miramos que se hayan introducido los datos correctamente
m=exist('THRU');
if m~=1
    disp('Error: no ha introducido los parametros s del THRU')
    return
end
m=exist('REFLECT');
if m~=1
    disp('Error: no ha introducido los parametros s del REFLECT')
    return
end
m=exist('LINE');
if m~=1
    disp('Error: no ha introducido los parametros s del LINE')
    return
end
m=exist('SAMPLE');
if m~=1
    disp('Error: no ha introducido los parametros s de la linea medida x')
    return
end
comptador=2;
n=1;
while n<500
%parametros s del thru
th11=THRU(comptador,2)*cos(THRU(comptador,3)*pi/180)+THRU(comptador,2)*sin(THRU(
comptador,3)*pi/180)*i;
```

```
th22=THRU(comptador,8)*cos(THRU(comptador,9)*pi/180)+THRU(comptador,8)*sin(THRU(
comptador,9)*pi/180)*i;
th12=THRU(comptador,4)*cos(THRU(comptador,5)*pi/180)+THRU(comptador,4)*sin(THRU(
comptador,5)*pi/180)*i;
th21=THRU(comptador,6)*cos(THRU(comptador,7)*pi/180)+THRU(comptador,6)*sin(THRU(
comptador,7)*pi/180)*i;
%definimos la matriz TH correspondiente a los param. R del thru
At=th11*th22-th21*th12;
TH=(1/th21)*[-At th11;-th22 1];
%parametros s del line
d11=LINE(comptador,2)*cos(LINE(comptador,3)*pi/180)+LINE(comptador,2)*sin(LINE(c
omptador,3)*pi/180)*i;
d12=LINE(comptador,4)*cos(LINE(comptador,5)*pi/180)+LINE(comptador,4)*sin(LINE(c
omptador,5)*pi/180)*i;
d21=LINE(comptador,6)*cos(LINE(comptador,7)*pi/180)+LINE(comptador,6)*sin(LINE(c
omptador,7)*pi/180)*i;
d22=LINE(comptador,8)*cos(LINE(comptador,9)*pi/180)+LINE(comptador,8)*sin(LINE(c
omptador,9)*pi/180)*i;
%definimos la matriz D correspondiente a los param. R del line
Ad=d11*d22-d21*d12;
D=(1/d21)*[-Ad d11;-d22 1];
%definimos la matriz T=D*inv(TH)
T=D*inv(TH);
%escribimos ec. de segundo grado y buscamos a=x11/x22 i b=x12/x22=e11
pol1=[T(2,1) (T(2,2)-T(1,1)) (-T(1,2) )];
B=roots(pol1);
a=B(1);
b=B(2);
%miramos si el modulo de a es mayor que el de b, si lo es OK sino
%intercambiamos valores
if abs(a)< abs(b)
    p=b;
    b=a;
    a=p;
end
%Definimos la matriz N=inv(TH)*D
N=inv(TH)*D;
%buscamos c=y11/y12 i d=y21/y22=-f11 a partir de N
pol2=[N(1,2) (N(2,2)-N(1,1)) (-N(2,1) )];
C=roots(pol2);
c=C(1);
d=C(2);
%miramos si el modulo de c es mayor que el de d, si lo es OK sino
%intercambiamos valores
if abs(c)< abs(d)
    q=d;
    d=c;
    c=q;
end
%definimos coeficientes de reflexion a la entrada de los dos reflect
gmx=REFLECT(comptador,2)*cos(REFLECT(comptador,3)*pi/180)+REFLECT(comptador,2)*s
in(REFLECT(comptador,3)*pi/180)*i;
gmy=REFLECT(comptador,8)*cos(REFLECT(comptador,9)*pi/180)+REFLECT(comptador,8)*s
in(REFLECT(comptador,9)*pi/180)*i;
%definimos coeficiente de reflexion del thru
gml=th11;
%buscamos los dos posibles valores de e22
e22a= sqrt(((b-gmx)/(a-gmx))*((c+gmy)/(d+gmy))*((b-gml)/(a-gml)));
e22b=-sqrt(((b-gmx)/(a-gmx))*((c+gmy)/(d+gmy))*((b-gml)/(a-gml)));
%miramos cual de los dos es correcto
gra=(1/e22a)*((b-gmx)/(a-gmx))+1;
```

```
grb=(1/e22b)*((b-gmx)/(a-gmx))+1;
if abs(gra)< abs(grb)
    e22=e22a;
else
    e22=e22b;
end
%encontramos los valores de f22, e21*e12, f21*f12, e21*f12 i f21*e12
f22=(1/e22)*((b-gml)/(a-gml));
e21e12=(b-a)*e22;
f21f12=(c-d)*f22;
e21f12=th21*(1-(e22*f22));
f21e12=th12*(1-(e22*f22));
%definimos parametros s medidos de la muestra
rm11=SAMPLE(comptador,2)*cos(SAMPLE(comptador,3)*pi/180)+SAMPLE(comptador,2)*sin
(SAMPLE(comptador,3)*pi/180)*i;
rm12=SAMPLE(comptador,4)*cos(SAMPLE(comptador,5)*pi/180)+SAMPLE(comptador,4)*sin
(SAMPLE(comptador,5)*pi/180)*i;
rm21=SAMPLE(comptador,6)*cos(SAMPLE(comptador,7)*pi/180)+SAMPLE(comptador,6)*sin
(SAMPLE(comptador,7)*pi/180)*i;
rm22=SAMPLE(comptador,8)*cos(SAMPLE(comptador,9)*pi/180)+SAMPLE(comptador,8)*sin
(SAMPLE(comptador,9)*pi/180)*i;
%definimos la matriz de parametros R de la muestra
Arm=rm11*rm22-rm21*rm12;
RM=(1/rm21)*[-Arm rm11;-rm22 1];
%obtenemos L y M
L=[(-a*e22) b;-e22 1];
M=[(c*f22) f22;d 1];
%gracias a L y M obtenemos la matriz de parametros R correctos de muestra
RA=e21f12*inv(L)*RM/M;
%a partir de los parametros R buscamos los parametros S correctos
s21=1/RA(2,2);
s11=RA(1,2)/RA(2,2);
s22=-RA(2,1)/RA(2,2);
s12=(RA(1,1)*s21+s11*s22)/s21;
%guardamos en f el valor de la frecuencia a la que estamos trabajando
f=x(comptador,1);
%guardamos en la linea correspondiente de la matriz los datos correctos y
%la frecuencia a la que pertenecen
S(n,1)=f;
S(n,2)=s11;
S(n,3)=s12;
S(n,4)=s21;
S(n,5)=s22;
n=n+1;
comptador=comptador+1;
end
disp('El programa ha finalizado correctamente')
disp('Los parametros s obtenidos se encuentran guardados en la matriz S')
%hacemos unas graficas a modo de muestra
subplot(2,2,1)
H=plotyy(S(:,1),20*log(abs(S(:,2))),S(:,1),angle(S(:,2))*180/pi);
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S11|(dB)')
set(get(H(2),'Ylabel'),'String','<S11(grados)')
subplot(2,2,2)
H=plotyy(S(:,1),20*log(abs(S(:,3))),S(:,1),angle(S(:,3))*180/pi);
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S12|(dB)')
set(get(H(2),'Ylabel'),'String','<S12(grados)')
subplot(2,2,3)
H=plotyy(S(:,1),20*log(abs(S(:,4))),S(:,1),angle(S(:,4))*180/pi);
```

```
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S21|(dB)')
set(get(H(2),'Ylabel'),'String','<S21(grados)')
subplot(2,2,4)
H=plotyy(S(:,1),20*log(abs(S(:,5))),S(:,1),angle(S(:,5))*180/pi);
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S22|(dB)')
set(get(H(2),'Ylabel'),'String','<S22(grados)')
```

8.1.3 Programa del método LRM para puff

```
%miramos que se hayan introducido los datos correctamente
m=exist('LINE');
if m~=1
    disp('Error: no ha introducido la matriz LINE correspondiente a los
parametros s del line')
    return
end
m=exist('MATCH');
if m~=1
    disp('Error: no ha introducido la matriz MATCH correspondiente a los
parametros s del match')
    return
end
m=exist('REFLECT');
if m~=1
    disp('Error: no ha introducido la matriz REFLECT correspondiente a los
parametros s del reflect')
    return
end
m=exist('SAMPLE');
if m~=1
    disp('Error: no ha introducido la matriz SAMPLE correspondiente a los
parametros s del sample')
    return
end
comptador=2;
num=1;
while num<500
%introducimos las matrices A,B,C y X correspondientes al line, reflect, match y
sample que
%estamos calculando actualmente
A(1,1)=LINE(comptador,2)*cos(LINE(comptador,3)*pi/180)+LINE(comptador,2)*sin(LIN
E(comptador,3)*pi/180)*i;
A(2,2)=LINE(comptador,8)*cos(LINE(comptador,9)*pi/180)+LINE(comptador,8)*sin(LIN
E(comptador,9)*pi/180)*i;
A(1,2)=LINE(comptador,4)*cos(LINE(comptador,5)*pi/180)+LINE(comptador,4)*sin(LIN
E(comptador,5)*pi/180)*i;
A(2,1)=LINE(comptador,6)*cos(LINE(comptador,7)*pi/180)+LINE(comptador,6)*sin(LIN
E(comptador,7)*pi/180)*i;

B(1,1)=MATCH(comptador,2)*cos(MATCH(comptador,3)*pi/180)+MATCH(comptador,2)*sin(
MATCH(comptador,3)*pi/180)*i;
B(2,2)=MATCH(comptador,8)*cos(MATCH(comptador,9)*pi/180)+MATCH(comptador,8)*sin(
MATCH(comptador,9)*pi/180)*i;
B(1,2)=MATCH(comptador,4)*cos(MATCH(comptador,5)*pi/180)+MATCH(comptador,4)*sin(
MATCH(comptador,5)*pi/180)*i;
B(2,1)=MATCH(comptador,6)*cos(MATCH(comptador,7)*pi/180)+MATCH(comptador,6)*sin(
MATCH(comptador,7)*pi/180)*i;

C(1,1)=REFLECT(comptador,2)*cos(REFLECT(comptador,3)*pi/180)+REFLECT(comptador,2
```

```
)*sin(REFLECT(comptador,3)*pi/180)*i;
C(2,2)=REFLECT(comptador,8)*cos(REFLECT(comptador,9)*pi/180)+REFLECT(comptador,8
)*sin(REFLECT(comptador,9)*pi/180)*i;
C(1,2)=REFLECT(comptador,4)*cos(REFLECT(comptador,5)*pi/180)+REFLECT(comptador,4
)*sin(REFLECT(comptador,5)*pi/180)*i;
C(2,1)=REFLECT(comptador,6)*cos(REFLECT(comptador,7)*pi/180)+REFLECT(comptador,6
)*sin(REFLECT(comptador,7)*pi/180)*i;

X(1,1)=SAMPLE(comptador,2)*cos(SAMPLE(comptador,3)*pi/180)+SAMPLE(comptador,2)*s
in(SAMPLE(comptador,3)*pi/180)*i;
X(2,2)=SAMPLE(comptador,8)*cos(SAMPLE(comptador,9)*pi/180)+SAMPLE(comptador,8)*s
in(SAMPLE(comptador,9)*pi/180)*i;
X(1,2)=SAMPLE(comptador,4)*cos(SAMPLE(comptador,5)*pi/180)+SAMPLE(comptador,4)*s
in(SAMPLE(comptador,5)*pi/180)*i;
X(2,1)=SAMPLE(comptador,6)*cos(SAMPLE(comptador,7)*pi/180)+SAMPLE(comptador,6)*s
in(SAMPLE(comptador,7)*pi/180)*i;
%buscamos las matrices D,E,N,M,O,R,P y definimos t
D=[C(1,1) 0 ; 0 B(2,2)];
E=[B(1,1) 0 ; 0 C(2,2)];
t=-1;
N=inv(E-A)*(B-E);
M=(A-C)*N;
O=B-C;
R=inv(D-A)*(B-D);
P=(A-C)*R;
%buscamos los parámetros m,n,o y p
m=(P(2,1)+O(2,1))*M(2,2)-(P(2,2)+O(2,2))*M(2,1);
n=O(2,1)*P(1,2)-O(2,2)*P(1,1);
o=(M(1,2)+O(1,2))*P(1,1)-(M(1,1)+O(1,1))*P(1,2);
p=O(1,2)*M(2,1)-O(1,1)*M(2,2);
%buscamos los elementos de la matriz T4
t12=1;
t15=sqrt((m*p)/(n*o))*((P(1,1)*t12)/M(2,2));
T=-t*((p*P(1,1)*t12)/(o*M(2,2)*t15));
t13=(-P(1,2)/P(1,1))*t15;
t14=(-M(2,1)/M(2,2))*t12;
%buscamos los elementos de la matriz T5
t8=(R(1,1)*t12+R(1,2)*t14)*(1/t)-t13*(1/T);
t9=(N(1,1)*t13+N(1,2)*t15)*(1/t)-t12*(1/T);
t10=(R(2,1)*t12+R(2,2)*t14)*(1/t)-t15*(1/T);
t11=(N(2,1)*t13+N(2,2)*t15)*(1/t)-t14*(1/T);
%definimos las matrizes T1,T2,T3 y T4
T1=[C(1,1)*t8+C(1,2)*t10-(1/t)*(O(1,1)*t12+O(1,2)*t14) C(1,1)*t9+C(1,2)*t11-
(1/t)*(O(1,1)*t13+O(1,2)*t15);C(2,1)*t8+C(2,2)*t10-(1/t)*(O(2,1)*t12+O(2,2)*t14)
C(2,1)*t9+C(2,2)*t11-(1/t)*(O(2,1)*t13+O(2,2)*t15)];
T2=[B(1,1)*t12+B(1,2)*t14 B(1,1)*t13+B(1,2)*t15;B(2,1)*t12+B(2,2)*t14
B(2,1)*t13+B(2,2)*t15];
T3=[t8 t9;t10 t11];
T4=[t12 t13;t14 t15];
%buscamos los parametros S correctos y los almacenamos en su fila
%correspondiente
Y=inv(T1-X*T3)*(X*T4-T2);
f=SAMPLE(comptador,1);
S(num,1)=f;
S(num,2)=Y(1,1);
S(num,3)=Y(1,2);
S(num,4)=Y(2,1);
S(num,5)=Y(2,2);
num=num+1;
comptador=comptador+1;
end
```

```
disp('El programa ha finalizado correctamente')
disp('Los parametros s obtenidos se encuentran guardados en la matriz S')
%hacemos las gráficas de muestra
subplot(2,2,1)
H=plotyy(S(:,1),20*log(abs(S(:,2))),S(:,1),angle(S(:,2))*180/pi);
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S11|(dB)')
set(get(H(2),'Ylabel'),'String','<S11(grados)')
subplot(2,2,2)
H=plotyy(S(:,1),20*log(abs(S(:,3))),S(:,1),angle(S(:,3))*180/pi);
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S12|(dB)')
set(get(H(2),'Ylabel'),'String','<S12(grados)')
subplot(2,2,3)
H=plotyy(S(:,1),20*log(abs(S(:,4))),S(:,1),angle(S(:,4))*180/pi);
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S21|(dB)')
set(get(H(2),'Ylabel'),'String','<S21(grados)')
subplot(2,2,4)
H=plotyy(S(:,1),20*log(abs(S(:,5))),S(:,1),angle(S(:,5))*180/pi);
xlabel('frequencia (GHz)')
set(get(H(1),'Ylabel'),'String','|S22|(dB)')
set(get(H(2),'Ylabel'),'String','<S22(grados)')
```

8.2 Usando los programas de calibración TRL y LRM

Aquí explicaremos los pasos a seguir para obtener los resultados que buscamos al aplicar el programa en *matlab*. A continuación mostramos los pasos:

- Una vez abierto *matlab* abriremos el fichero .m correspondiente al programa a ejecutar (hay dos versiones para el programa TRL una si los parámetros S son del analizador vectorial, que deben estar guardados en una hoja de cálculo de *excel* en forma de parte real y parte imaginaria; otra por si los datos los hemos obtenido mediante una simulación con *puff* y que anteriormente debemos abrirlos con el bloc de notas, quitarles la cabecera y al parte de texto del final y guardarlos en formato .txt. Para el programa LRM solo existe la segunda versión)
- Antes de ejecutarlo debemos importar los datos de la siguiente manera. La parte correspondiente al *thru* como THRU, la parte correspondiente al *line* como LINE, la parte correspondiente al *reflect* como REFLECT, la parte del *match* como MATCH y, finalmente, la parte correspondiente a la muestra como SAMPLE. Si no lo hacemos así el programa nos indicará que dato no hemos importado y no se ejecutará. Cabe destacar que al importar los datos con *matlab* este automáticamente nos detectará la parte correspondiente a los datos y la parte correspondiente a las lineas de cabecera con lo que nos saldran dos secciones a elegir:*data* correspondiente a la primera y *textdata* correspondiente a la segunda. Nosotros debemos seleccionar la primera y canviarle el nombre según correponda a los datos que estamos importando.
- Ahora ya podemos ejecutar el programa, el cual encontrará la matriz en función de la frecuencia de los parámetros S correctos de la muestra y nos los representará en una gráfica. Para que el usuario pueda hacer más gráficas u operar con ellos esta matriz se guardará bajo el nombre de S y en ella la primera columna será la correpondiente a las frecuencias y las cuatro siguientes las correspondientes a los valores s11, s12, s21 y s22 en el orden indicado y expresados en forma de parte real y parte imaginaria.

8.3 Imágenes del analizador vectorial y de la muestra

En este apartado mostramos una serie de imágenes que nos muestran el analizador vectorial que usamos para hacer la comprobación del algoritmo TRL así como la muestra usada para ello.

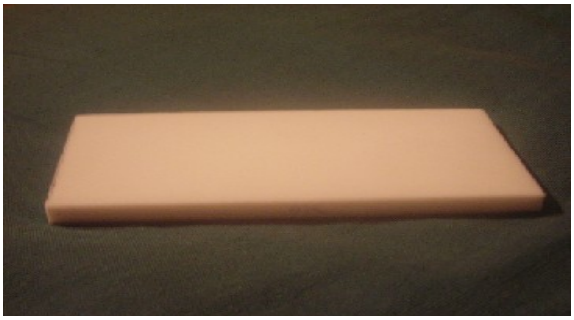


Figura 8.1 - Muestra PTFE usada

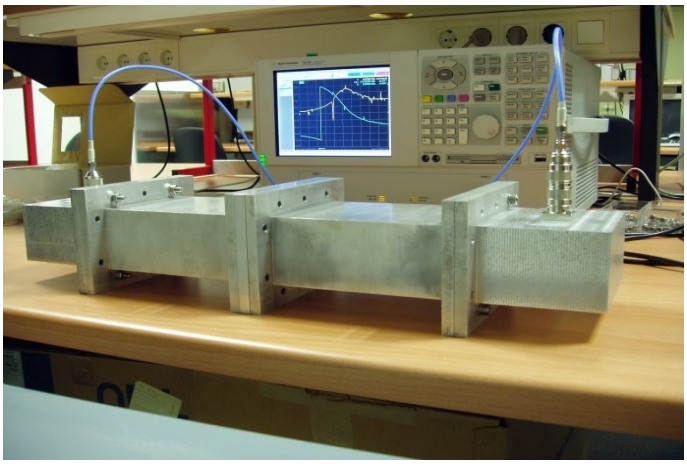


Figura 8.2 - Analizador Vectorial de Redes usado

Debemos comentar que, en el caso de la imagen correspondiente al analizador vectorial, ésta fue tomada al hacer unas mediciones que no se corresponden al proyecto, pero ha sido incluida porque lo que se pretende es mostrar como es el analizador en sí.

9 RESUMEN

En este proyecto, lo primero que hemos hecho ha sido desarrollar un algoritmo en *matlab* que implementara el método de calibración TRL, el funcionamiento del cual lo hemos comprobado en primera instancia mediante simulaciones y, posteriormente, mediante un ejemplo real. Posteriormente, hemos desarrollado otro algoritmo en *matlab* para implementar el método de calibración LRM. Este algoritmo sólo lo hemos podido comprobar a nivel de simulación. A continuación, mediante los dos algoritmos, hemos realizado una comparación entre ambos sistemas de calibración a través de simulaciones. Finalmente, analizando los resultados de varias simulaciones calibradas con nuestro programa del método TRL, hemos buscado cuales pueden ser los motivos para la apairición de picos indeseados y hemos encontrado uno de ellos.

En aquest projecte,el primer que hem fet ha sigut desenvolupar un algoritme en *matlab* que implementès el mètode de caibració TRL, el funcinament del qual hem pogut comprovar en primera instància mitjantçant simulacions i, posteriorment, mitjantçant un exemple real. Posteriorment, hem desenvolupat un altre algoritme en *matlab* per implementar el métode de calibració LRM. Aquest algoritme només l'hem pogut comprovar a nivell de simulació. A continuació, mitjantçant els dos algoritmes, hem realitzat una comparació entre ambdós sitemes de calibració a través de simulacions. Finalment, analitzant els resultats de varias simulacions calibrades amb el nostre programa del métode TRL, hem buscat quins poden ser els motius de l'aparició de pics indesitjats i hem trobat un d'ells.

In this project, the first thing that we have done has been to develop an algorithm in matlab that implement the method of calibration TRL, the operation that we have verified in first instance by means of simulations and, later, by means of a real example. Later, we have developed another algorithm in matlab to implement the method of calibration LRM. We only have been able to verify this algorithm at simulation level. Next, using both algorithms, we have made a comparison between both systems of calibration through simulations. Finally, analyzing the results of several simulations calibrated with our program of method TRL, we have looked for the reasons why undesired tips appear and we have found one of them.