Instituto Nacional de Astrofísica, Óptica y Electrónica

DESARROLLO DE UNA HERRAMIENTA DE ANÁLISIS DE CIRCUITOS PWL

Por:

María del Rocío De Jesús Ventura

Tesis sometida como requisito parcial para obtener el grado de Maestro en Ciencias en la especialidad de Electrónica en el Instituto Nacional de Astrofísica, Óptica y Electrónica

Supervisada por:

Dr. Luis Hernández Martínez, INAOE.

Dr. Librado Arturo Sarmiento Reyes, INAOE.

Tonantzintla, Pue.

29 de Junio de 2007

Agradecimientos

De manera muy especial, agradezco al Consejo Nacional de Ciencia y Tecnología por el apoyo económico para poder realizar mis estudios de maestría.

Mi más profundo agradecimiento, al Dr. Luis Hernández y al Dr. Arturo Sarmiento, por el soporte y la dedicación que dieron como resultado el buen término de este trabajo de tesis.

A quien ha significado algo muy especial en mi vida, Javier Hurtado, gracias por tu amor y todo el apoyo que me brindas. Junto a ti el mundo tiene otro sentido.

Por la fortaleza y ternura, que me han ofrecido a largo de todo el camino recorrido, gracias mamá Rosío Ventura y papá Lorenzo De Jesús. Todos mis logros son posibles gracias a ustedes.

A los responsables de darme tanta felicidad: a mis hermanos. Gracias Paty, Omar, Ulises, Irazema y sus retoñitos Karla y el próximo bebé. No hay meta que unidos no podamos alcanzar.

Su confiaza en mí y su apoyo han sido de gran valor, gracias por involucrarse en mi vida, Claudia Juárez, Dr. Roberto Murphy, Dr. Víctor Champac, Dr. Arturo Sarmiento, Luis Herández, Miguel A. García y Antonio Zenteno.

A todas las personas que de alguna u otra forma se hicieron partícipes de esta meta y sus acciones ayudaron a culminarla. Gracias por su grandeza humana.

Resumen

Las herramientas de software son un auxiliar imprescindible para la Electrónica de nuestros tiempos, cada vez surgen propuestas para tratar de dar solución a los nuevos problemas que se presentan o a los problemas clásicos que aquejan a esta área en particular.

Una de esas propuestas consiste en el uso de *Modelos Piecewise Linear*, los cuales se basan en la representación de segmentos de líneas rectas que manifiestan el comportamiento aproximado de los elementos no lineales dentro de un circuito. La ventaja que ofrece, es el cambiar un problema no lineal de cierta complejidad, por un conjunto de problemas lineales que pueden ser resueltos con mayor facilidad.

El trabajo aquí expuesto presenta algunos de los modelos piecewise linear más usados, divididos en explícitos e implícitos. Dentro del conjunto de representaciones piecewise linear explícitas, está la conocida como de Chua, la cual puede ser modelada en una sola ecuación conocida como canónica. La representación de Bokhoven pertenece al grupo de las implícitas y ésta se basa en expresiones de una red resistiva lineal con k puertos y usa diodos ideales . La Herramienta desarrollada para este trabajo utiliza estos dos modelos para la representación de los elementos no lineales, ya que el hecho de haber sido programada de tal manera que tuviera una estructura modular, la hace factible de agregar mas de una representación. Cabe mencionar que sólo la opción del modelo de Chua llega hasta la etapa de solución del circuito.

El simulador realizado para este trabajo de tesis ha sido implementado siguiendo las reglas de la ingeniería de software. Éste realiza su labor a través de un archivo de entrada que es revisado sintácticamente; internamente se detectan los elementos no lineales y se hace la conversión a la representación correspondiente, posteriormente se realiza el análisis en CD y se imprimen en pantalla los resultados obtenidos. La plataforma que utiliza el simulador para trabajar es MAPLE 9.5 y lo hace mediante una hoja de trabajo en la cual se

introduce el nombre del archivo con la descripción del circuito a simular.

Abstract

Software tools are strongly needed in modern electronic design. This need opens the field to new methodologies not only for tackling new problems but also for dealing or improving the solution procedure for classical problems.

One of these methodologies consists in the use of piece-wise linear (PWL) models in order to represent the nonlinearities of the circuit. The PWL representation yields a set of lines that approximates the original nonlinear characteristic of a given circuit component. As a result, the problem of finding the DC operating point is transformed from solving a nonlinear algebraic equation into solving a system of linear algebraic equation under certain validity conditions than can be solved with less difficulty.

In this work, several PWL representations, both explicit and implicit, are introduced and programmed in a circuit simulation package. Among the explicit representations, the canonical representation from Chua has been selected because it consists in a single canonical expression which is enough to represent a PWL model. Within the implicit representations, we have selected the model from Bokhoven because it has a circuit equivalent that can be regarded as a linear resistive k-port.

The simulation package has been programmed in a modular structure in order to handle both representations above and to permit future improvements with the addition of more representations. Besides, the design and conceptualization of the tool was done by resorting to software engineering rules, as far as the programming environment allowed. The package has been fully developed under the MAPLE environment, more specifically MAPLE 9.5 because it has the possibility to use symbolic manipulation, and the PWL representations are treated in semi-symbolic form. The package is able to find the DC solution and when the circuit possesses multiple operating points, it is also capable of asses more than one. Several well-known benchmark circuits are used to test the PWL simulation tool and the results have been

compared with continuous nonlinear simulations.

Prefacio

El comportamiento no lineal de los componentes de un circuito ha hecho que se busquen alternativas para poder representar de alguna manera la forma en que se desempeñan estos dispositivos. Es aquí donde los modelos y los simuladores tienen un rol muy importante en el análisis de circuitos.

La presente tesis se enfoca al desarrollo de un software que sea un auxiliar para el análisis en CD de circuitos que contengan elementos no lineales, dada también la importancia de este tipo de análisis ya que es el paso introductorio para realizar otros tipos de análisis como el de pequeña señal y el de CA. Por ello el análisis en CD es el paso de inicio para cualquier simulador.

La tesis lleva una secuencia tal que va desde la introducción al trabajo dando las justificaciones y la factibilidad de éste, pasando por la explicación de tópicos de modelado en la cual se describen los modelos piecewise linear más conocidos, para dar una idea general de ellos y tener los fundamentos teóricos necesarios para la comprensión de los algoritmos implementados. Continuo a esto se presenta las etapas y la estructura del simulador programado en lenguaje MAPLE, donde se plantean cada una de las fases con que cuenta el programa y se describen a manera de seudocódigo los algoritmos propuestos para cada etapa. También se muestran ejemplos los cuales siguen las reglas propuestas para el simulador (estas reglas son incluidas a manera de manual en la parte final del trabajo) y con los resultados obtenidos podemos constatar la válidez de los métodos de solución que se aplicaron. Estos ejemplos se simularon bajo el ambiente windows, aunque cabe mencionar que la herramienta es compatible con linux y unix, la computadora utilizada para dichas simulaciones cuenta con las siguientes características: procesador Intel pentium 4, velocidad de 3.20 GHz, 512 MB de RAM y disco duro de 80GB.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

En la actualidad la utilización de simuladores juega un papel muy importante en varias áreas de la Ciencia y la Tecnología. La Electrónica no es la excepción, ya que básicamente la simulación está presente durante todo el proceso de diseño de circuitos y también sistemas con diferentes niveles de abstracción. En el ámbito de la simulación a nivel circuital, esta disiplina tuvo sus inicios a mediados de los 60's, cuando debido al impacto de las computadoras, el quehacer de los diseñadores empezó a cambiar. Estos comenzaron a divisar las herramientas de análisis que se pudieran implementar en las computadoras digitales con el principal objetivo de predecir el comportamiento de un circuito dado, automatizando el análisis del mismo. Con la ayuda de los programas de computadoras los diseñadores estuvieron listos no sólo para implementar el circuito −una vez que su funcionamiento había sido validado−, sino también para llevar a cabo experimentos numéricos, a menudo más rápidos y económicos que los realizados con prototipos de banco. Tales programas de computadora son llamados simuladores de circuitos [19]. Este fue el inicio de una nueva disciplina en la Electrónica: Diseño Asistido por Computadora (Computer Aided Design, CAD), la cual juega en nuestros días un importante papel en el diseño de circuitos electrónicos, ya que la complejidad de los diseños se incrementa cada vez más a causa del incremento en la escala de integración de los circuitos, es decir que las cantidad de dispositivos aumentan y sus dimensiones disminuyen. De este modo

los simuladores empiezan a tener problemas principalmente en la convergencia de los métodos usados para resolver las ecuaciones del circuito, así como también en el incremento de tiempo necesario del CPU para la obtención de los resultados.

El propósito de la simulación de circuitos, es obtener información sobre el comportamiento que tiene una red circuital en términos de variables eléctricas. Dentro de ese comportamiento, un punto básico y de gran importancia tanto en el análisis como en el diseño, es hallar la solución en CD (corriente directa). Para realizar este tipo de análisis, el método más ampliamente utilizado a lo largo de los años es el método iterativo de Newton-Raphson (N-R). Este es el método utilizado por *SPICE*, el cual es uno de los simuladores con mayor demanda tanto en el ambiente industrial como en el académico. El método de N-R consiste básicamente en aproximar un conjunto de valores iniciales de las variables desconocidas a un valor final dicho valor puede ser considerado la solución cuando cumple con cierto criterio de tolerancia. Entre las desventajas que se pueden mencionar acerca de N-R, tenemos en primer lugar el alto costo computacional, especialmente en circuitos muy grandes, ya que una gran parte de los recursos computacionales son consumidos por la solución repetida del sistema de ecuaciones resultado de la linerización. En segundo lugar existe un problema originado por las limitadas capacidades de convergencia de N-R y en casos muy difíciles el programa aborta ya que depende de los valores iniciales. Finalmente, en el método de N-R hay una carencia de flexibilidad respecto a la introducción de nuevos modelos de componentes [26].

Una opción para realizar el análisis en CD, es linealizar en modo continuo a trozos, mediante técnicas conocidas como *piecewise linear* (PWL). Existen simuladores que han atacado los problemas que presenta N-R, utilizando métodos PWL alternativos para la resolución de las ecuaciones derivadas del circuito, entre los que se pueden mencionar: MOTIS [2], DIANA [9], SPLICE [18], PLANET [15] y PLATO [26]. Los dos últimos han explotado las características que ofrece la descripción de modelos usando funciones Piecewise Linear (PWL). En principio, la simulación PWL luce como la simulación de circuitos convencional, pero la primera conlleva unas importantes diferencias. Una de ellas es que en la simulación PWL, un conjunto de ecuaciones lineales piecewise linear debe ser resuelto, lo que implica que no hay necesidad de iteraciones. Otra de las diferencias, es que la solución del sistema de ecuaciones es posible realizarla usando métodos en los cuales se tiene mayor fortaleza en las propiedades de convergencia global, que la mostrada por N-R

al resolver las ecuaciones no lineales. Anuado a lo anterior, otra ventaja es que el modelado piecewise linear está abierto para realizar los modelos de una gran variedad de componentes de un modo uniforme, implementando un nivel-mixto de simulación de en una manera natural [15, 26].

Las características mencionadas anteriormente son las que hacen interesante la utilización de representaciones PWL, por lo que explorar la utilización de las descripciones PWL en un simulador es de gran importancia, ya que a partir de su implementación se puede generar nuevo conocimiento, además nos da otra opción para la obtención de la solución en CD.

El presente trabajo de tesis consiste en el desarrollo de un simulador donde se encuentra implementada la descripción PWL de Chua y Bokhoven, pero en sí cuenta con una estructura modular, de tal modo que se puede agregar más representaciones PWL y como trabajo a futuro poder hacer comparaciones entre ellas. La herramienta desarrollada, está enfocada al análisis en CD (el cual sólo se realiza con la representación de Chua), aunque está abierta a la implementación de otros tipos de análisis, gracias al ya mencionado enfoque de modularidad con que cuenta el programa y de este modo poder adicionar código que realice simulación convencional u otros métodos de interés.

Con lo anteriormente expuesto, el objetivo de tesis es: tener una herramienta de software que sea capaz de realizar simulación de circuitos basado en técnicas PWL, que sea la base para la implementación de más representaciones PWL, además de permitir simulación convencional o algún otro tipo de metodología de interés. Todo esto bajo una sintáxis a la *SPICE*, que resulte familiar al usuario final.

En lo que respecta a la organización del escrito de este trabajo se dividió en cinco capítulos y dos apéndices, los cuales se describen a continuación.

En el capítulo 2, se presentan las representaciones piecewise linear más usuales, mostrando su clasificación y la descripción matemática correspondiente a cada una de ellas. También se hace mención de algunos simuladores PWL.

El capítulo 3, muestra primeramente la estructura general del simulador, y posteriormente va detallando cada sección o bloque del diagrama representativo del simulador.

El capítulo 4 trata sobre los casos de estudio y aplicaciones del presente trabajo, así como también sobre la explicación de los resultados. Con lo que se puede obtener una idea de la potencialidad del uso de métodos PWL.

Las conclusiones acerca de la tesis y el trabajo futuro que se puede derivar de ésta, son los puntos discutidos en el capítulo 5.

Finalmente, en los apéndices A y B, se muestra respectivamente, un breve manual de usuario del simulador y una descripción de las variables utilizadas y de mayor relevancia dentro del código de la herramienta.

Capítulo 2

Modelos Piecewise Linear y Simuladores

En la literatura existen diferentes métodos para lograr la representación de funciones piecewise linear, cada una de ellas tiene sus ventajas y desventajas. En este capítulo se muestran las características principales de cuatro de las representaciones PWL más conocidas. Así mismo se mencionan algunos de los simuladores PWL existentes y se da una comparación entre ellos.

La organización del capítulo es desarrollada de la siguiente manera: en la primera sección se aborda una discusión a grandes rasgos acerca de modelado. En la segunda sección se presentan una serie de definiciones fundamentales de representaciones PWL. En seguida en la sección tres se discuten diferentes modelos, divididos en explícitos e implícitos. Respecto a los primeros se describen los métodos de Chua y el HL-CPWL, y en cuanto a los segundos, son las representaciones de Bokhoven1 y Bokhoven2 las que son mostradas. En la última sección, se ve cómo la facilidad para encontrar los parámetros de los modelos puede ser usada en la elaboración de simuladores.

2.1. Modelado no lineal

La gran mayoría de los sistemas utilizados en electrónica son no lineales, por lo que se tiene que seleccionar algún método de linealización que nos

conduzca de una mejor manera a la solución de dichos sistemas. La no linearidad de estos sistemas se debe a que los circuitos que los originan contienen elementos no lineales, los cuales pueden ser modelados de tal forma que sean factibles de ser incluidos en un algoritmo dando paso así a la creación de simuladores. Entre más se aproxime el modelo a las características reales, se obtendrán mejores resultados, pero el modelo puede resultar demasiado complejo. Generalmente en la fase inicial del modelado las propiedades de los componentes que se consideran de poca importancia son omitidas o simplificadas. La selección de la complejidad del modelo depende de la exactitud que requiera o necesite el diseñador.

Aproximar una función no lineal o un comportamiento a través de una función matemática conocida es una técnica que se usa frecuentemente. El propósito es obtener una mayor idea del comportamiento por medio de la aplicación de teoremas matemáticos conocidos, sobre una función matemática. Existen tres principales conceptos en la aproximación [1]:

- Funciones polinomiales. Sobre ellas se han realizado muchas investigaciones en lo que respecta al campo de las aproximaciones de funciones, usando polinomios así como también funciones trascendentales. La aplicación de estas técnicas finalmente derivó en simuladores como $SPICE$. Entre las más conocidas están la de Taylor, Chebyshev y Padé. En la aproximación trascendental, las formas sinusoidales son utilizadas para aproximar el comportamiento no lineal.

- Table-look-up. En ésta se lleva a cabo un muestreo de la función no lineal para obtener un conjunto de puntos. Estos puntos son almacenados en una tabla, de tal modo que si un valor de función es requerido la correspondiente entrada en la tabla es usada. Aunque si el valor solicitado de la función no fue directamente almacenado, se tienen que emplear técnicas de interpolación y extrapolación para obtener el valor. La ventaja de esta técnica es la velocidad, la desventaja es que requiere mucha memoria para almacenar los valores de la función.

- Funciones Piecewise Linear. La aproximación de la no linearidad por medio del uso de varias descripciones lineales afines, es un método frecuentemente utilizado. De esta manera el problema de una sola ecuación no lineal, es transformado en varias ecuaciones lineales. En matemáticas,

hay mucho más conocimiento y flexibilidad en la teoría lineal que en la teoría de las funciones no lineales.

El modelado PWL, que es el que interesa para este trabajo en particular, tiene distintas formas de descripción algunas de las más conocidas son discutidas en la siguiente sección, partiendo de sus fundamentos y su clasificación.

2.2. Modelos piecewise linear

En esta sección se aborda como primer punto, los fundamentos generales de las descripciones PWL, posteriormente se discute su clasificación y de ese modo se presentan cuatro modelos PWL del conjunto de representaciones existentes.

2.2.1. Definiciones fundamentales

Una función PWL es la aproximación de un comportamiento no lineal utilizando un mapeo de descripciones lineales por segmentos, donde cada segmento puede ser visto como una ecuación de recta. De este modo, el problema básico es transformado de una ecuación no lineal a un sistema de ecuaciones lineales. Una función PWL se define como sigue [4]:

Definición 1

Una función $\mathbf{f} : \mathbf{D} \subset \mathbf{R}^n \rightarrow \mathbf{R}^m$ *, donde* $\mathbf{D}$ *es un conjunto compacto, se dice que es piecewise-linear si y sólo si satisface las siguientes condiciones:*

1. *Sea el espacio dominio D, el cual está dividido en un número finito de regiones poliédricas*[1] *por un numero finito de fronteras de dimension* $(n-1)$*, de modo que cada frontera (o conjunto de fronteras) es un hiperplano caracterizado por:*

$$\boxed{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle - \beta = 0} \tag{2.1}$$

donde $\boldsymbol{\alpha}, \boldsymbol{x} \in \mathbf{R}^n$*,* $\beta \in \mathbf{R}^1$ *y "*$\langle , \rangle$*"denota el producto punto de dos vectores* $\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle = \boldsymbol{\alpha}^T \boldsymbol{x} = \boldsymbol{\alpha}^T \boldsymbol{x}^T$*.*

[1] Se designan por $R^{(1)}, R^{(2)}, ..., R^{(k)}$ (donde $\mathbf{D} = \bigcup_{i=1}^{k} R^{(i)}$)

2. $\boldsymbol{f}$ *está expresada en cada región por una representación afín de la forma*

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{J}^{(k)}\boldsymbol{x} + \boldsymbol{w}^{(k)} \tag{2.2}$$

para cualquier $\boldsymbol{x} \in R^{(k)}$*, donde* $R^{(k)}$ *es la region del espacio dominio definida como:*

$$R^{(k)} = \{x \in \mathbf{R}^n | \boldsymbol{\alpha}_i^T \boldsymbol{x} + \beta_i \geq 0, i \in k; \boldsymbol{\alpha}_i^T \boldsymbol{x} + \beta_i \leq 0, i \notin k \tag{2.3}$$

con $i = 1, 2, \ldots, k$*. Por otro lado,* $\boldsymbol{J}^{(k)} \in \mathbf{R}^{n \times n}$ *se conoce como la Matriz Jacobiana y* $\boldsymbol{w}^{(k)} \in R^n$*.*

3. *La función* $\boldsymbol{f}$ *es continua para cualquier frontera entre dos regiones. Es decir, si* $\boldsymbol{J}^{(n)}$*,* $\boldsymbol{J}^{(m)}$ *son las matrices jacobianas de dos regiones vecinas* $R^{(n)}$ *y* $R^{(m)}$ *que están separadas por el hiperplano* $\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle - \beta = 0$*, entonces*

$$\boldsymbol{J}^{(n)}\boldsymbol{x} + \boldsymbol{w}^{(n)} = \boldsymbol{J}^{(m)}\boldsymbol{x} + \boldsymbol{w}^{(m)} \tag{2.4}$$

para cualquier $\boldsymbol{x} \in R^{(n)} \cap R^{(m)}$*.*

Cuando una frontera o hiperplano se cruza con otra en algún punto se presenta una intersección. En la siguiente definición se introduce el concepto de intersecciones de fronteras [12].

Definición 2

En una partición del dominio $\mathbf{D} \subset \mathbf{R}^n$ *se dice que un hiperplano de dimensión* $(n-1)$ *es una intersección de primer orden, denotada como* S^1*. Una frontera lineal de dimensión* $(n-k)$ *en* $\mathbf{D}$ *es una intersección de orden* k *denotada como* S^k*, si es la intersección entre dos o más fronteras lineales de tipo* S^{k-1} *designadas por* $S^k = \cap_{i=1}^{\geq 2} S_i^{k-1}$

Está claro que una función PWL es lineal dentro de cada región $R^{(k)}$. Como consecuecia el comportamiento no lineal se observa en las fronteras que separan las regiones. Por otro lado, existe una relación especial entre dos expresiones afines de la forma 2.2 cuando éstas corresponden a regiones vecinas.

De hecho, la diferencia entre matrices Jacobianas y los vectores $\boldsymbol{w}^{(i)}$, que pertenecen a regiones adyacentes separadas por un hiperplano con un vector normal $\boldsymbol{\alpha}$ pueden ser expresadas como un producto $\mathbf{c}\boldsymbol{\alpha}^T$, es decir

$$\begin{array}{l} \Delta \boldsymbol{J}^{(i,j)} \doteq \boldsymbol{J}^{(i)} - \boldsymbol{J}^{(j)} = \boldsymbol{c}^{(i,j)}\boldsymbol{\alpha}^T \\ \Delta \boldsymbol{w}^{(i,j)} \doteq \boldsymbol{w}^{(i)} - \boldsymbol{w}^{(j)} = \boldsymbol{c}^{(i,j)}\beta \end{array} \tag{2.5}$$

donde $\boldsymbol{J}^{(i)}, \boldsymbol{J}^{(j)} \in \mathbf{R}^{m\times n}$, $\boldsymbol{w}^{(i)}, \boldsymbol{w}^{(j)}, \boldsymbol{c}^{(i,j)} \in \mathbf{R}^{(m)}$, $\boldsymbol{\alpha} \in \mathbf{R}^{(n)}$ y $\beta \in \mathbf{R}^1$.
Esta propiedad es una consecuencia directa de la continuidad de la función. Considere una frontera $H_q = \{\boldsymbol{x} \in \mathbf{R}^n : \boldsymbol{\alpha}_q^T\boldsymbol{x} + \beta_q = 0\}$ que separa las regiones $R^{(i)}$ y $R^{(j)}$, se debe cumplir la relación:

$$\Delta \boldsymbol{J}^{(i,j)}\boldsymbol{x} + \Delta \boldsymbol{w}^{(i,j)} = 0 \tag{2.6}$$

Si la ecuación 2.6 se describe componente a componente, entonces se obtiene la siguiente igualdad:

$$(\Delta \boldsymbol{J}^{(i,j)}\boldsymbol{x} + \Delta \boldsymbol{w}^{(i,j)}) - \boldsymbol{c}_k^{(i,j)}(\boldsymbol{\alpha}_q^T\boldsymbol{x} + \beta_q) = 0 \tag{2.7}$$

para $k = 1, 2, \ldots n$. Luego, si 2.7 es válida para cualquier $\mathbf{x} \in H$, es decir cualquier punto dentro de la frontera cumple con:

$$(\Delta \boldsymbol{J}_k^{(i,j)} = \boldsymbol{c}_k^{(i,j)}\boldsymbol{\alpha}_q^T, \Delta \boldsymbol{w}^{(i,j)}) = \boldsymbol{c}_k^{(i,j)}\beta_q) = 0 \tag{2.8}$$

que es precisamente la condición 2.5 con $\boldsymbol{c}^{(i,j)} = [c_1^{(ij)}, c_2^{(i,j)}, \ldots, c_n^{(i;j)}]^T$.

2.3. Representaciones para funciones piecewise-linear

El modo más simple de expresar un mapeo PWL es por medio de la representación convencional, la cual consiste en definir la función región por región

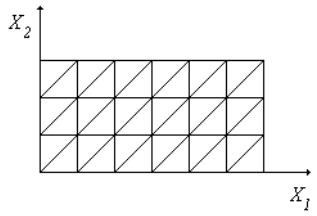


Figura 2.1: Partición simplex de un dominio en $\boldsymbol{R}^2$

como en 2.2. Desafortunadamente, esta descripción requiere un número excesivo de parámetros debido a las condiciones de continuidad de la función con respecto a las regiones vecinas. Por otro lado, para aquellos casos en que se tienen una gran cantidad de segmentos lineales no es factible realizar manipulaciones algebraicas usando esta metodología. Es por ello que se han desarrollado dos tipos de descripciones para representar funciones PWL, la representación explícita y la representación implícita.

2.4. Representación explícita

La representación explícita es conocida por generar la función PWL a partir de una expresión. Los modelos de Chua y la HL-CPLW son dos de las representaciones explícitas conocidas, las cuales se describen a continuación.

2.4.1. Modelo de Chua

La primera expresión canónica para representar funciones PWL $f : \mathbf{R}^1 \rightarrow \mathbf{R}^1$ fue introducita en [5]. Para el análisis de este caso, es conveniente considerar una función arbitraria PWL definida convencionalmente como:

$$f(x) = J^{(k)}x + w^{(k)}, \;\; si \;\; x \in R^{(k)} \tag{2.9}$$

donde

$$R^{(0)} = (-\infty, \beta_1), R^{(1)} = (\beta_1, \beta_2), \ldots, R^{(n)} = (\beta_n, \infty) \tag{2.10}$$

El siguiente teorema formula la expresión canónica que representa la función descrita por 2.9.

Definición 3

Cualquier función PWL univaluada con al menos σ puntos de quiebre $\beta_1 < \beta_2 \ldots < \beta\sigma$, puede ser representada únicamente por la expresión

$$f(x) = a + bx + \sum_{i=1}^{\sigma} c_i|x - \beta_i| \tag{2.11}$$

donde $a, b, c_i, \beta_i \in \mathbf{R}^1$ pueden ser calculados como sigue:

$$\begin{aligned} b &= (J^{(0)} + J^{(\sigma)})/2 \\ c_i &= (J^{(i)} - J^{(i-1)})/2, i = 1, 2, \ldots, \sigma \\ a &= f(0) - \sum_{i=1}^{\sigma} c_i|\beta_i| \end{aligned} \tag{2.12}$$

Observe que si se define la partición de $\mathbf{R}^1$ en $(\sigma + 1)$ regiones, entonces la representación convencional 2.9 requiere $2(\sigma + 1)$ parámetros, mientras que la expresión canónica 2.13 requiere tan sólo $(2 + \sigma)$ parámetros.
La existencia de la representación canónica en $\mathbf{R}^1$ motivó la extensión de la expresión 2.13 para dominios dimensionales arbitrarios, presentada también por Chua y Kang en 1978 [13].

Definición 4

Una función canónica PWL (CPWL) $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$, es una función que puede ser representada como

$$f(\boldsymbol{x}) = \boldsymbol{a} + \boldsymbol{B}\ \boldsymbol{x} + \sum_{i=1}^{\sigma} \boldsymbol{c}_i|\boldsymbol{\alpha}_i^T \boldsymbol{x} - \beta_i| \tag{2.13}$$

donde $\boldsymbol{a}, \boldsymbol{c}_i \in \mathbf{R}^m, \boldsymbol{B} \in \mathbf{R}^{m \times n}$ y $\beta_i \in \mathbf{R}^m$ para $i = 1, 2, \ldots \sigma$.

En la siguiente sección se muestra otra representación para una función arbitraria.

2.4.2. Modelo HL-CPWL

Encontrar una función analítica y los parámetros para la representación canónica de una función PWL arbitraria es un problema que existe dentro de las representaciones explícitas. En 1999 se presenta una aportación para coadyuvar este problema [12], pues se formula una representación de alto nivel (HL CPWL) y una metodología para obtener los coeficientes para todas las funciones PWL definidas en un dominio de $\mathbf{R}^n$ y dividido por una configuración *simplex* de fronteras como se muestra en la figura 2.1. Una de las razones de esta elección es que los hiperplanos que dividen el dominio en regiones y vértices pueden ser generados sistemáticamente. La definición de un simplex es la siguiente:

Definición 5

Sean $x_0, x_1, \ldots x_n$ puntos en el espacio de dimensión n. Un simplex $\Delta(x_0, \ldots x_n)$ se define como

$$\boxed{\Delta(x_0, \ldots, x_n) = \{x := x = \textstyle\sum_{i=0}^{n} \mu_i x_i\}} \tag{2.14}$$

donde $0 \leq \mu_i \leq 1, i \in 1, \ldots, n$ y $\sum_{i=1}^{n} \mu_i = 0$

La extensión de esta idea a un dominio n-dimensional requiere definir simplex de $n+1$ vértices. Entonces, con un valor de la función asociado a cada vertex (vértice), es posible determinar una función afín (local) única para cada simplex, de modo que la colección de todas las funciones lineales determinen una función PWL continua. Por otro lado, es necesario considerar una característica importante de un conjunto de funciones PWL que comparten la misma partición:

Definición 6

Considere un dominio compacto $D \subset R^n$ y un conjunto de hiperplanos H. Entonces, $PWL_H[D]$ está definido como el conjunto de todos los mapeos continuos PWL que toman valores en el dominio D particionado por la configuración de fronteras H.

Se considera el caso de una función PWL definida sobre un hipercubo con una intersección degenerada única centrada en el origen. Desde el punto de vista

conceptual, este análisis es de gran importancia y se puede extender para cubrir un dominio mas general dado por un conjunto rectangular compacto en R^n. En ambos casos, la metodología usada consiste de los siguientes pasos:

- Descripción del dominio y su partición
- Descripción de los vértices asociados
- Descripción de las funciones generadoras
- Formulación y prueba del resultado pricipal

2.5. Representación implícita

La representación implícita, a diferencia la explícita, necesita un algoritmo para ser resuelta. Las siguientes subsecciones describen dos de los modelos existentes: Bokhoven 1 (Bok1)y Bokhoven 2 (Bok2).

2.5.1. Modelo de Bokhoven 1

Dentro de la clasificación de modelos PWL, tenemos la representación implícita, propuesta en el año de 1981 por W.M.G. van Bokhoven [1]. Esta representación se basa en expresiones de una red resistiva lineal con k puertos y usando diodos ideales.

$$\begin{aligned} \boldsymbol{y} &= A\boldsymbol{x} + B\boldsymbol{u} + \boldsymbol{f} \\ \boldsymbol{j} &= C\boldsymbol{x} + D\boldsymbol{u} + \boldsymbol{g} \end{aligned} \tag{2.15}$$

donde $A \in \mathbf{R}^{m\times n}, B \in \mathbf{R}^{m\times k}, \boldsymbol{f} \in \mathbf{R}m, C \in \mathbf{R}^{k\times n}, D \in \mathbf{R}^{k\times k}, \boldsymbol{g} \in \mathbf{R}^k, \boldsymbol{u}, \boldsymbol{j} \in \mathbf{R}^k$ y las variables $\boldsymbol{u}, \boldsymbol{j} \in \mathbf{R}^k$ satisfacen el problema conocido como *lineal complementario* o LPC, por sus siglas en inglés (Linear Complementary Problem) [14], como se muestra a continuación:

$$\boldsymbol{u}^T\boldsymbol{j} \geq 0, \boldsymbol{u} \geq 0, \boldsymbol{j} = 0 \tag{2.16}$$

El sistema 2.15 puede ser descompuesto en dos partes diferentes: la ecuación de salida

$$\boxed{\boldsymbol{y} = A\boldsymbol{x} + B\boldsymbol{u} + \boldsymbol{f}} \tag{2.17}$$

y la ecuación de estados

$$\boxed{\boldsymbol{j} = D\boldsymbol{u} + \boldsymbol{q}\ (\boldsymbol{x})} \tag{2.18}$$

donde $\boldsymbol{q}\ (\boldsymbol{x}) = C\boldsymbol{x} + \boldsymbol{g}$.

Es evidente que la ecuación de salida 2.17 se puede usar para calcular $\boldsymbol{y}$ si se conocen $\boldsymbol{x}$ y $\boldsymbol{u}$. En efecto, cualquier estado dado para los diodos está relacionado a una configuración particular de las variables $\boldsymbol{u}$ y $\boldsymbol{j}$. Por ejemplo, suponga en una red con dos diodos, donde el primer diodo —caracterizado por (u_1, j_1)— está polarizado en directa y el segundo diodo —caracterizado por (u_2, j_2)— está polarizado en inversa, se tienen las relaciones $u_1 = 0, j_1 \geq 0$ y $u_2 \geq 0, j_2 = 0$. Cuando estas condiciones son sustituidas en la ecuación 2.18, se obtiene un conjunto de desigualdades lineales para el vector $\boldsymbol{x}$, el cual además determina una región del espacio dominio. El caso más sencillo es cuando:

$$\boxed{u_i = 0, \forall i = 1, 2, \ldots, k} \tag{2.19}$$

Aquí, la ecuación $\boldsymbol{j} = \boldsymbol{q}\ (\boldsymbol{x}) = C\boldsymbol{x} + \boldsymbol{g} \geq 0$ define una región caracterizada por

$$\boxed{\{\boldsymbol{x} \in \mathbf{R}^n : \boldsymbol{q}\ (\boldsymbol{x}) = C\boldsymbol{x} + \boldsymbol{g} \geq \boldsymbol{0}\}} \tag{2.20}$$

Observe que todas las regiones 2^k pueden obtenerse correspondientes a todas las posibles combinaciones de los estados del diodo. Sin embargo, para poder obtener la ecuación de la región del dominio correspondiente a la configuración de los diodos diferente de 2.19, son necesarias varias manipulaciones algebraicas sobre la ecuación de estados 2.18, conocidas como *operaciones de pivoteo diagonal* [14].

El análisis de todos los posibles estados de los diodos permite encontrar todas las regiones de los dominios también como sus expresiones correspondientes. Sin embargo, cuando el valor de la función debe calcularse para un

valor dado de $\boldsymbol{x}$, se vuelve evidente que primero es necesario encontrar las variables desconocidas $\boldsymbol{u}$ y $\boldsymbol{j}$. Este problema es conocido en la literatura como el Problema Lineal Complementario (LCP) [11]. Por consiguiente, su solución numérica es una tarea difícil la cual es equivalente a buscar la región del dominio donde $\boldsymbol{x}$ pertenecen a todas las posibles configuraciones de las variables $\boldsymbol{u}$ y $\boldsymbol{j}$ (es decir, todas las posibles operaciones de pivoteo sobre la matriz D).

El modelo presentado en esta sección es conocido como Bokhoven 1 (Bok1), ya que existe otra variante denominada Bokhoven 2 (Bok2), la cual es descrita en la siguiente sección.

2.5.2. Modelo de Bokhoven 2

La descripción implícita Bok2 presenta algunas diferencias con respecto a (Bok1). El sistema de ecuaciones cambia a la forma:

$$\begin{aligned} 0 &= Iy + Ax + Bu + f \\ j &= Dy + Cy + Iu + g \\ u &\geq 0,\ \ j \geq 0,\ \ u^T j = 0 \end{aligned} \tag{2.21}$$

Con $A \in \mathbf{R}^{m\times n}, B \in \mathbf{R}^{m\times k}, C \in \mathbf{R}^{k\times n}, D \in \mathbf{R}^{k\times m}, f \in \mathbf{R}^m$ y $g\mathbf{R}^k$

Además se observa que los hyperplanos en Bok2 son planos en un sentido estricto, ya que la ecuación $j = Dy + Cy + Iu + g$ cuando $u = j = 0$ toma la forma $0 = Dy + Cy + g$, la cual desde el punto de vista de geometría analítica es la ecuación de un plano.

Finalmente una diferencia que es importante destacar es la referente a la ubicación de hyperplanos en Bok1 y Bok2. En Bok1 los hyperplanos son generados por la misma característica no lineal, ya que su ubicación es regida por los puntos de quiebre, es decir, donde exista un punto de quiebre debe existir un hyperplano. En Bok2 la ubicación de hyperlanos es arbitraria lo que da como resultado múltiples opciones para llegar a una descripción Bok2.

2.6. Simuladores

A los simuladores de circuitos que utilizan funciones PWL para describir las no linearidades de los componentes son llamados Simuladores Piecewi-

se Linear. Con cualquiera de los modelos anteriores es posible construir un simulador PWL. En la literatura han sido reportados simuladores que han utilizado los modelos de Chua y Bokhoven, [7] y [16, 25, 26] respectivamente.

Simulador usando Chua

Chua y Ying reportaron en [7] que un simulador había sido desarrolado usando la representación de Chua. La restricción de este simulador consistía en que sólo podía manejar circuitos construidos a partir de fuentes controladas no lineales y resistores. En la práctica esto no parece ser una gran limitante para las redes simuladas a nivel circuito. Aunque, para sistemas que usan un alto nivel en macro modelos, esto representa una severa restricción. Por otro lado, la representación de Chua puede describir funciones multi-dimensionales. Esto significa que para dispositivos de dos terminales solamente se usa una parte del potencial de la representación. Usando uno de los métodos convencionales para análisis de circuitos (cortes, mallas, nodal modificado, etc.) las ecuaciones del circuito son transformadas a la forma canónica de Chua, donde x contiene un conjunto de variables de la red. El tamaño de $\mathbf{x}$ depende del método de análisis de circuitos seleccionado. Como en el análisis de redes normal las ecuaciones PWL $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ son resueltas usando un método introducido por Katznelson.

La propiedad de convergencia global de este método es mejor que la del método de Newton Raphson. Con extensiones dadas por Katznelson en [3], este método siempre encontrará una solución para $\mathbf{x}$, esto se debe también principalmente al hecho que la representación de Chua puede solamente modelar funciones en el estricto sentido. Si los hiperplanos del modelo equivalente del elemento no lineal tienen una estructura especial (es decir reticular) un método más efeciente llamado breakpoint-hopping, puede ser utilizado.

Simulador usando Bokhoven

Dos simuladores han sido reportados que han utilizado el método de Bokhoven en [25] y [26]. El método de Bokh1 es más poderoso que el de Chua y esto implica que este necesita de un algoritmo poderoso para encontrar una solución.

Bokh1 permite transformar un número de modelos conectados, en un modelo de la misma forma sin ninguna de las variables internas de la red circuital que se esté analizando. Esto abre la posibilidad de derivar modelos compactos de funciones complicadas, lo cual no es posible hacer con Chua. otra posibilidad es simplemente concatenar diferentes modelos para formar nuevamente

otro modelo de la forma Bokh1 y adicionarle ecuaciones topológicas. Este es el método utilizado en PLATO [26]: la red completa es transformada en un gran modelo PWL que es almacenado utilizando técnicas de matrices dispersas. Una solución es entonces obtenida usando el algoritmo de Van de Panne. En [25] se sigue una aproximación diferente: La jerarquía en la red (si es que está presente), es retenida por el almacenamiento en un jacobiano para cada nodo en el árbol jerarquico.

Ambos métodos sufren el hecho que implícitamente, el método del Tableau es el utilizado para resolver las ecuaciones de la red. Esto significa que todos los voltajes y corrientes del circuito son calculadas, lo cual es notorio en el tiempo de computo.

2.7. conclusiones

Existen varias representaciones PWL, de las cuales las expuestas en este capítulo, son las básicas para la comprensión del modelado PWL y todas pueden ser utilizadas para construir simuladores PWL. La representación canónica de Chua, fue la que se implementó en la herramienta de software, que se desarrolló para este trabajo de tesis, puesto que es un método base dentro del ámbito del piecewise linear y genera la función PWL apartir de una expresión.

Respecto a los simuladores descritos en los apartados anteriores, cabe mencionar que no son los únicos que utilizan técnicas PWL, existen otros como por ejemplo: SAMOC [10], que utiliza modelos PWL de MOS, Chatoyan [17], que es un simulador para dispositivos ópticos y Saber [23] que usa modelos PWL en el diodo y para cuestiones de control.

Capítulo 3

Estructura del Simulador

Los circuitos electrónicos en la actualidad cuentan con un grado de complejidad tal, que analizarlos por los medios antiguos es prácticamente imposible. Históricamente el análisis de circuitos electrónicos ha ido tomando caminos diferentes y desde diferentes puntos de vista, pero en sí, el objetivo primordial al que se desea llegar es el mismo: obtener el comportamiento cuantitativo de las variables eléctricas.

A mediados de los 60's y debido al impacto de las computadoras el quehacer de los diseñadores empezó a cambiar. Estos comezaron a divisar las herramientas de análisis que se pudieran implementar en las computadoras digitales con el principal objetivo de mejorar el análisis de circuitos. Con estas medidas, los experimentos numéricos se volvieron más rápidos y económicos que los realizados con los circuitos alambrados. Tales programas de computadora son los llamados simuladores de circuitos [21]. Algunos de los problemas a resolver típicos de simulación, son listados en la tabla 3.1 [6].

3.1. Anatomía de los programas de simulación

La estructura general para representar un simulador consta de cinco pasos principales:

Tipo de circuito	Descripción del problema
I. Resistivo lineal (sin capacitores e inductores) y dinámico lineal (con al menos un capacitor o un inductor).	1. Análisis en CD (encontrar la solución de una red resistiva lineal). 2. Análisis en CA (encontrar la respuesta transitoria de de una red dinámica lineal). 3. Análisis de ruido (análisis en CA o transitorio con fuentes de ruido como entrada). 4. Análisis de tolerancia (sensibilidad o análisis del peor de los casos). 5. Determinación de la localización de polos y ceros de las funciones de transferencia. 6. Generación de funciones de redes simbólicas.
II. Resistivo no lineal (sin capacitores e inductores).	1. Análisis del punto de operación (encontrar las soluciones en CD de una red resistiva no lineal). 2. Determinar la variante característica estática del circuito o graficar una variable eléctrica `vs` un barrido de voltaje o corriente. 3. Determinación de las características de transferencia (encontrar la relación de un voltaje o corriente de entrada `vs` un voltaje o corriente de salida). 4. Encontrar la forma de onda de salida debido a la función de la entrada con respecto al tiempo.
III. Dinámico no lineal (con al menos un capacitor o un inductor).	1. Condiciones iniciales, o análisis del estado de equilibrio (análisis del punto de operación con todos los inductores en corto circuito y los capacitores en circuito abierto). 2. Análisis transitorio (encontrar la forma de onda de salida dada una condición inicial, con o sin señal de excitación). 3. Análisis del estado continuo (encontrar la solución periódica del estado continuo con o sin señale de excitación). 4. Análisis de distorsión no lineal (encontrar distorsiones armónicas, de modulación e intermodulación).

Tabla 3.1: Problemas típicos de análisis de circuitos.

- Etapa de entrada
- Etapa de sustitución de modelos de dispositivos
- Etapa de la formulación de la ecuación de equilibrio
- Etapa de la solución numérica
- Etapa de salida

El diagrama de flujo de la figura 3.1 ilustra la relación de estos pasos.

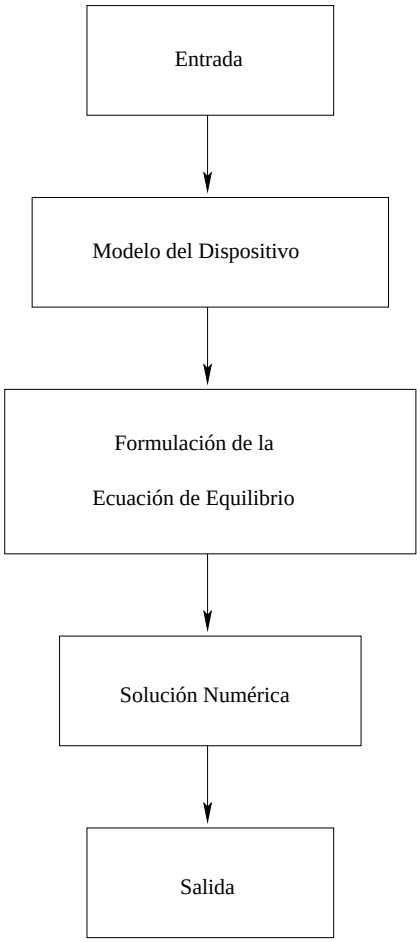


Figura 3.1: Principales etapas de los programas simuladores por computadora

- **Entrada** En la etapa entrada la computadora recibe la información del usuario con las consideraciones de la configuración del circuito, las características de los elementos y los tipos de análisis a realizar. Aquí se debe de realizar una verificación de la entrada dada por el usuario de acuerdo a unas reglas de sintaxis y gramática en búsqueda de errores.

- **Modelo del dispositivo** En éste se manejan los modelos de los dispositivos. Generalmente este paso no es necesario para pequeños programas de simulación o para programas escritos con uso de instrucciones, pero se vuelve muy importante en programas usados para diseñar circuitos electrónicos, en los que modelos dados por los fabricantes deben estar incluidos en forma de bibliotecas de modelos.

- **Formulación de la Ec. de equilibrio** En éste, el programa de simulación formula la ecuación de equilibrio para el circuito, en la cual la configuración y los valores de los elementos son completamente especificados. A continuación se enlistan los métodos más utilizados en programas de simulación computacional.

  - Métodos basados en incidencia. Dentro de esta categoría se pueden mencionar los métodos nodal y nodal modificado este último de reconocida utilidad en aplicaciones para simulación de circuitos. En estos métodos las variables de interés están fuertemente ligadas a detalles topológicos por lo que voltajes nodales, corrientes de lazos y voltajes de cortes, representan las variables fundamentales [20, 28].
  - Método de variable de estado. De aplicación restringida al dominio del tiempo, este método establece la ecucación de equilibrio en términos de estados o variables, usualmente voltajes de capacitores y corrientes de inductores [6].
  - Método híbrido. Este tipo de métodos, muy utilizados sobre todo en el análisis de circuitos para comunicaciones, se combinan corrientes y voltajes como variables de interés. Una forma restringida involucra la representación de un circuito por una red de dos puertos [22].

- **Solución Numérica** En el penúltimo paso, las ecuaciones de equilibrio son resueltas numéricamente. Para la solución de las ecuaciones

algebraicas lineales, que resultan de los métodos mencionados anteriormente, con coeficientes reales o complejos, se utiliza el método de eliminación gaussiana o el método de descomposición LU. Para la solución de ecuaciones algebraicas no lineales, que resultan del análisis nodal o híbrido, algunos métodos que se utilizan son el método de Newton Raphson y el lineal a segmentos (piecewise linear, PWL). Para la solución de ecuaciones lineales en el dominio del tiempo se pueden obtener con la ayuda de la matriz exponencial e^a. Para ecuaciones no lineales en el dominio del tiempo, una solución analítica, generalmente no es posible, por lo que se tienen que aplicar técnicas de integración numérica.

- **Salida** La última etapa es la salida, puesto que aquí es donde obtenemos las respuestas del análisis realizado. Dependiendo del tipo de análisis, serán los resultados obtenidos y el desplegado de estos variarán dependiendo del diseño del simulador, por lo que existe una gran variedad, así como también de opciones para visualizar e interpretar de una mejor manera dichos resultados.

3.2. Estructura del Simulador PWL (Sim-PWL)

En la seccion 3.1 se hizo referencia a las principales etapas con las que debe contar un simulador, el caso del simulador que se elaboró para este trabajo de tesis, no es la excepción.

El Sim-PWL (simulador Piecewise Linear) es la herramienta desarrollada para este trabajo, es un simulador enfocado a la solución de circuitos mediante modelos PWL. En esta fase inicial del simulador el tipo de análisis que se puede llevar a cabo es el de CD utilizando el modelo de Chua (el modelo de Bokhoven sólo llega hasta la representación del elemento). Aunque dada la estructura modular del Sim-PWL se pueden agregar otros tipos de análisis o cualquier otro método de solución adicional. Ver figura 3.2.

El esquema de la figura 3.3, donde se muestran las etapas del Sim-PWL, representa sólo la parte correspondiente al análisis en CD, por lo que la exposición de las características de cada una de las etapas es hecha en base a esta particularidad. Como resultado, en las siguientes subsecciones serán tratadas cada una de dichas etapas, con un enfoque hacia el trabajo de tesis.

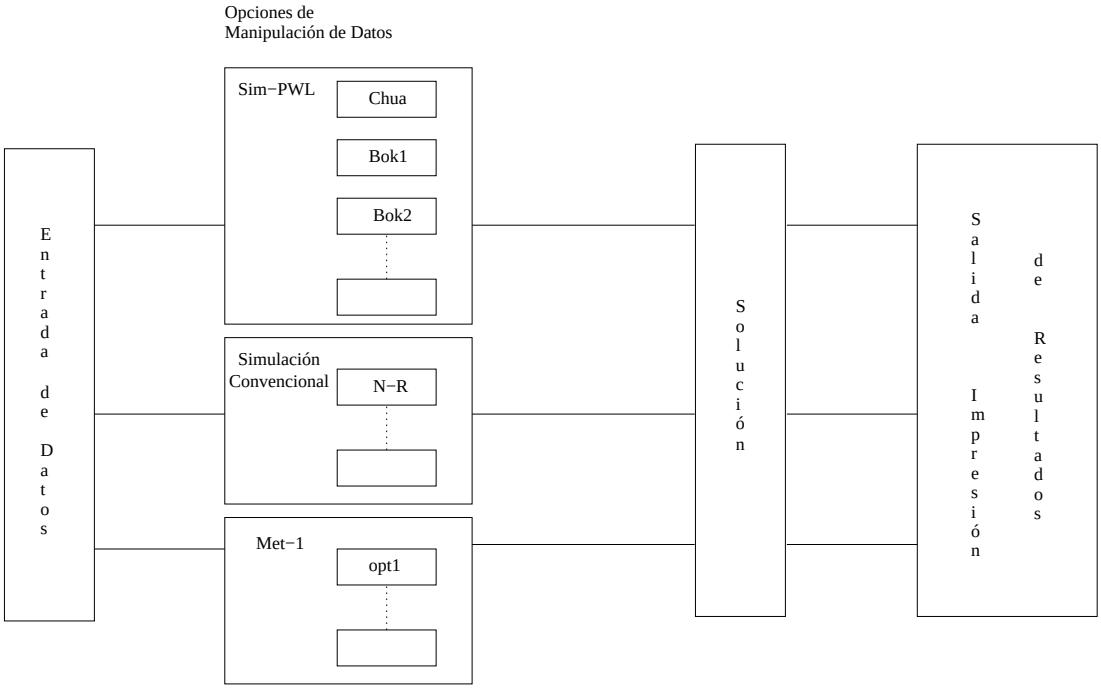


Figura 3.2: Estructura modular a bloques del Sim-PWL.

3.2.1. Etapa de entrada

La etapa de entrada comprende la gramática y la interpretación de los datos que se introducen al programa del simulador. La gramática utilizada es semejante a la utilizada en *SPICE* con algunas variantes. La lectura de las características del circuito se hace por medio de la plataforma MAPLE las cuales están contenidas en un archivo con extensión *cir*. La primera línea de entrada del archivo es considerada como el título del circuito a simular y la última línea debe ser la etiqueta *end*. Entre estas líneas se da la descripción del circuito, el tipo de análisis y modelos, todo esto puede ser escrito en orden indistinto. El archivo de entrada puede ser generado usando algún editor de texto, pero debe de cumplir con unas cuantas características que se enlistan a continuación.

1. Puede escribirse un enunciado y solamente un enunciado en cada línea.

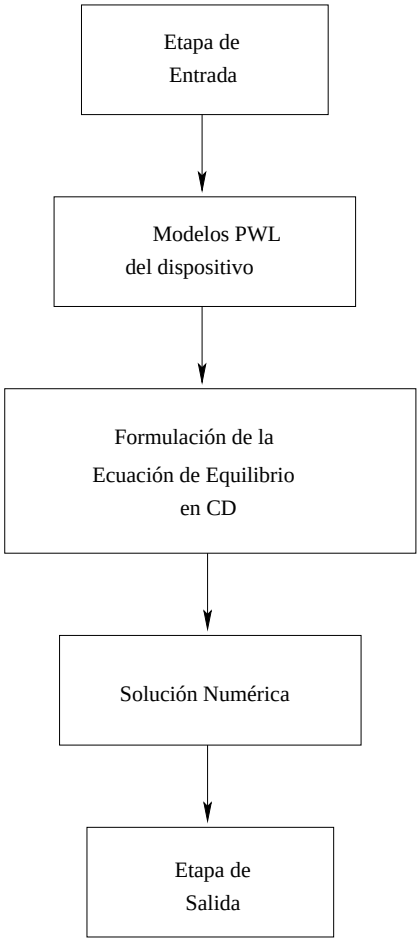


Figura 3.3: Etapas del simulador en DC

2. Entre cada enunciado puede haber cualquier número de líneas en blanco o líneas comentariadas. Las líneas con comentarios comienzan con el caracter especial #.

3. Cualquier enunciado debe comenzar en la primera columna.

4. Entre diferentes parámetros de un enunciado, se requiere mínimo de un espacio. (Es posible que tenga más de un espacio, pero el enunciado completo debe estar en la misma línea).

Gramática de entrada

La gramática de entrada está definida como la serie de reglas que establecen la forma correcta en que deben ser escritas cada una de las líneas que describen el circuito a simular y que se encuentran dentro del archivo de entrada. Las siguientes líneas muestran la gramática utilizada para el archivo de entrada, con el objeto de darles un significado semántico de acuerdo al contexto en que aparezcan dentro de estas líneas, se hacen ciertas consideraciones como lo son asumir un $< ret >$ en cada fin de línea y al caracter | que toma el significado de un ó y '␣' que implica un espacio.
Las líneas del archivo de entrada al simulador están definidas por la siguiente gramática general:

```
lines::=   line
         | lines <RET> line
line::=    elemdef
         | comment
         | endfile
```

Es decir, una línea puede contener la definición de un elemento, un comentario o la línea de de fin de archivo.

A continuación se da la gramática particular para cada uno de los diferentes *tokens* del simulador.

```
elemdef::=   elemname‘␣’nodelist‘␣’value
           | elemname‘␣’nodelist‘␣’paramlist
           | elemname‘␣’nodelist‘␣’paramlist initval
           | elemname‘␣’nodelist‘␣’modelname

comment::= ‘#’ text

endfile::= ‘end’ | ‘END’ <fin del archivo>

elemname::= elemINIT <cualquier identificador permitido>

nodelist::=   node
            | nodelist‘␣’node

value::= <cualquier valor numérico decimal de MAPLE>
```

```
paramlist::=   param
             | paramlist‘␣’param
             | PWLparam

initval::= IndepEV ‘=’ value

modelname::=   <nombre del modelo>

text::= <cualquier cadena de caracteres>

elemINIT::= | ‘v’ | ‘V’
            | ‘i’ | ‘I’
            | ‘d’ | ‘D’
            | ‘r’ | ‘R’
            | ‘c’ | ‘C’
            | ‘l’ | ‘L’
            | ‘e’ | ‘E’
            | ‘f’ | ‘F’
            | ‘g’ | ‘G’
            | ‘h’ | ‘H’
            | ‘n’ | ‘N’
            | ‘o’ | ‘O’
            | ‘p’ | ‘P’
            | ‘q’ | ‘Q’
            | ‘m’ | ‘M’

node::= <cualquier entero o string válido en MAPLE>

param::= paramID ‘=’ paramDEF

PWLparam::= <definición PWL permitida>

IndepEV::=   ‘i’   *ver NOTA
           | ‘u’

paramID::= <cualquier variable válida en MAPLE>
```

Identificador		*Nombre*
v	V	Fuente Independiente de Voltaje
i	I	Fuente Independiente de Corriente
r	R	Resistor
d	D	Diodo
l	L	Inductancia
c	C	Capacitor
e	E	Fuente de Voltaje controlada por Voltaje
f	F	Fuente de Corriente controlada por Corriente
g	G	Fuente de Corriente controlada por Voltaje
h	H	Fuente de Voltaje controlada por Corriente
n	N	Nullator
o	O	Norator
p	P	Nullor
q	Q	BJT
m	M	MOSFET

Tabla 3.2: Identificadores de los dispositivos

```
paraDEF::=   value
           | EVfun

EVfun::=   ufunction   *ver NOTA
         | ifunction

ufunction::= <cualquier expresión que use u>

ifunction::= <cualquier expresión que use i>

NOTA:Los elementos contemplados en la presente tesis son resistivos por
lo cual en el esquema sólo se toman en cueanta las variables i y u.
```

La tabla 3.2 muestra los identificadores elemINIT con su respectivo significado. Obsérvese que pueden usarse mayúsculas y minúsculas.

Los datos leídos son revisados gramaticalmente por el parser de entrada de manera tal que se cumplan las reglas. Los errores en que se puede incurrir

Error	*Ocurrencia del error*
RET omitido	line
Espacio omitido	elemdef
Elemento repetido	elemname
Nodo flotando	nodes
Etiqueta omitida	definition, modelname, endfile
Identificador incorrecto	elemINIT, paramID
Caracter especial omitido al inicio de la línea	(#) comment
Caracter especial omitido	(=) param, inival

Tabla 3.3: Lista de errores en que se puede incurrir.

al revisar los datos de entrada se enlistan en la tabla 3.2.1.

Los datos del circuito a simular son almacenados en un Netlist que es una lista de listas de formato MAPLE:

```
Netlist:= [[elemname
           [nodelist]
           [value]
           [control]]
```

De manera más desarrollada el Netlist toma la siguiente forma:

```
Netlist:= [[elemname-a, elemname-b, elemname-c, ...]
           [[node-a1,node-a2],[node-b3,node-b4], [node-c5,node-c6], ...]
           [paramlist-a, paramlist-b, paramlist-c, ...]
           [control]]
```

La lista control se forma por todas las fuentes controladas existentes en el circuito a simular.

Ventajas y desventajas del lenguaje de entrada

El lenguaje de entrada tiene la principal ventaja que es un lenguaje *à la SPICE*, el cual es bastante conocido en el medio de simulación de circuitos. La desventaja es que se depende de la plataforma MAPLE para ejecutar el simulador, y de los comandos que contiene.

Algoritmo de la etapa de entrada

1. Leer cada una de las líneas del archivo de entrada.
2. Almacenar los datos en diferentes variables.
3. Revisar los datos almacenados.
4. Si existe un error de gramática en alguna línea entonces detener el programa, si la gramática es correcta entonces continuar.
5. Formar el Netlist con los datos almacenados.

3.2.2. Modelos PWL del dispositivo

En esta segunda etapa, lo que procede es substituir internamente en la lista de valores (value), la descripción de los elementos resistivos no lineales, obtenida del archivo de entrada por su modelo PWL, ya sea la representación explícita de Chua o la implícita de Bokhoven. Para el caso de los transistores BJT, solo aplica la representación de Chua.

Algoritmo de la etapa de entrada

1. Mediante posiciones de listas identificar a los elementos resistivos no lineales y a los transistores BJT.
2. Cambiar su valor actual por su modelo en la posición correspondiente.
   - Resistores no lineales
     - Si la etiqueta del elemento es: PWL, usar la representación de Chua.
     - Si la etiqueta del elemento es: BPWL, usar la representación de Bokhoven.
   - BJTs
     - Si el nombre del elemento es: Q o q, usar la representación de Chua.
3. Actualizar la lista value.

3.2.3. Formulación de la ecuación de equlibrio en CD

Una vez que se ha obtenido el Netlist, la etapa siguiente es la formulación de la ecuación de equilibrio en CD, la cual se lleva a cabo a través de la manipulación de los datos proporcionados por éste, anuado a la utilización de la representación canónica de Chua.

El método base que se seleccionó para la formulación es el bien conocido MNA (Modified Nodal Analysis) [24], por sus características que lo hacen aplicable a la programación debido al uso de *stamps* y a que soporta a los elementos compatibles y no compatibles al análisis nodal, considerando las corrientes de rama de estos últimos como variables adiciopnales y sus correspondientes relaciones de rama como ecuaciones adicionales.

El Netlist contiene la información necesaria para crear la matriz MNA ($\boldsymbol{A}$) y colocar los *stamps* dentro de ésta, así como para generar el vector de estimulos ($\boldsymbol{d}$).

Lo anterior corresponde a la representación de la parte lineal, la parte no lineal, es decir los elementos no lineales, ya representados por la función canónica de Chua, son identificados respecto a la dependencia de corriente o voltaje y son agrupados en los vectores correspondientes ($\tilde{\boldsymbol{c}}(i)$ y $\tilde{\boldsymbol{b}}(u)$), que a la vez son sumados sumados a la matriz MNA. La multiplicación de la matriz MNA y el vector de variables desconocidas ($\boldsymbol{x}$) más la suma de los vectores no lineales y el vector de estimulos, todo igualado con cero, forman el sistema de ecuaciones total que representa a la ecuación de equilibrio 3.1.

$$\boldsymbol{A}\boldsymbol{x} + \tilde{\boldsymbol{b}}(u) + \tilde{\boldsymbol{c}}(i) + \boldsymbol{d} = \boldsymbol{0} \tag{3.1}$$

Algoritmo de la formulación de la ecuación de equilibrio en DC

1. Entran los datos del Netlist
2. Se identifica al elemento.
3. Si es resistor y es no lineal con etiqueta PWL, entonces:
   - *a*) Llama a las rutinas PWL
     - Traduce el valor a la forma canónica de Chua.
   - *b*) Coloca la función resultante en el vector no lineal correspondiente.

4. Si es diodo entonces:

   a) Identifica el valor y la posición del elemento.

   b) Llama a las rutinas PWL

      - Utiliza el modelo con forma canónica de Chua.

   c) Coloca la función resultante en el vector no lineal correspondiente.

5. Si es BJT entonces:

   a) Identifica el valor y la posición del elemento.

   b) Llama a las rutinas PWL

      - Utiliza el modelo con forma canónica de Chua.

   c) Coloca las funciones resultantes en el vector no lineal correspondiente.

6. Si es resistor y es lineal entonces:

   a) proceder normalmente realizando los *stamps* en la matriz MNA y el vector de estimulos.

7. Si es alguno de los otros elementos permitidos, entonces:

   a) proceder normalmente realizando los *stamps* en la matriz MNA y el vector de estimulos.

3.2.4. Proceso de solución

Después de la formulación de la ecuación de equilibrio, sigue la etapa de solución, en la cual el sistema de ecuaciones resultado de la ecuación 3.1, se resuelve utilizando las rutinas con que cuenta MAPLE para resolver sistemas de ecuaciones lineales.

Una vez que se ha resuelto para todas las variables, se obtienen el vector con los resultados, que representa el o los punto(s) de operación del circuito descrito en el archivo de entrada. Los resultados obtenidos son de tipo numérico.

Cabe mencionar que en el proceso de solución, el rango dado al elemento PWL dentro del archivo de entrada afecta el que se encuentre o no la solución del circuito, así como también la obtención de más de una solución es debida

Lista	*Contenido*
Anlist	Opción de análisis
Control	Fuentes controladas con su variable de control
Elements	Netlist
Names	Nombres de los elementos
Nodes	Nodos
Title	Título
Values	Valores de los elementos

Tabla 3.4: Listas generadas en la etapa de salida después del parser.

a las propiedades que implica el valor absoluto dentro de las ecuaciones que conforman el sistema.

Algoritmo de solución

1. Se lee el dato almacenado y se identifica la opción de análisis.
2. Si es DCPWL entonces.
   a) Se resuelven el sistema de ecuaciones.
   b) En caso de no tener solución en el rango seleccionado, manda un mensaje.

3.2.5. Desarrollo de la etapa de salida

La etapa de salida del simulador es mostrada mediante la plataforma de MAPLE. Los resultados que muestra el simulador en primera instancia son una serie de listas con los datos del circuito a simular obtenidos del archivo de entrada (ver tabla 3.2.5).

Después de las listas, se muestran los resultados del análisis, presentando la lista del circuito, los nombres de los dispositivos, el total de nodos, los valores de los dispositivos, la lista de análisis, la matriz lineal, el vector de variables desconocidas, el vector de excitación, el vector no lineal de funciones PWL $i = f(u)$, el vector no lineal de funciones PWL $u = f(i)$ y el vector de soluciones.

3.2.6. Conclusiones

El Sim-PWL esta preparado para soportar otras aplicaciones, ya sea de simulación convencional o alguna otra de interés especial como lo es el diseño estructurado, ya que cuenta con la aceptación de los elementos ideales nullator, norator y nullor [27]. También cuenta con la opción de implementación de modelos, lo cual es muy útil para el caso de los BJT y MOSFET.

En cuanto al planteamiento de la solución, se optó por utilizar MNA por sus ya conocidos beneficios de interpreatción circuital. En el caso de la inclusión de los elementos no lineales dentro del plateamiento de solución, se realizó la separación de parte lineal y no lineal, de manera tal que se respetarán los lineamientos de MNA en sus voltajes y corrientes.

Un punto importante de los resultados obtenidos, es el hecho de poder encontrar, en algunos casos, más de un punto de operación. Lo que nos da la pauta para trabajar en el tópico de múltiples soluciones en CD.

Capítulo 4

Aplicaciones y Casos de Estudio

En este capítulo se describen algunos ejemplos ilustrativos de circuitos con elementos no lineales, los cuales fueron resueltos por el simulador Sim-PWL. La lista de circuitos a presentar es:

- Circuito con diodo túnel
- Circuito simple con resistencia no lineal dependiente de voltaje
- Circuito con dos resistencias no lineales dependientes de voltaje
- Circuito con transistores
- Circuito con representación Bokohoven

En cada caso se presenta el diagrama respectivo, el archivo de entrada y el desplegado de resultados.

4.1. Circuito con diodo Túnel

La figura 4.1 muestra el diagrama del primer ejemplo, el cual está constituido por una fuente de voltaje, una resistencia lineal y una resistencia no

lineal, que tiene el comportamiento de un diodo túnel, modelado por una función polinomial controlada por voltaje, la que es internamente convertida a función PWL.

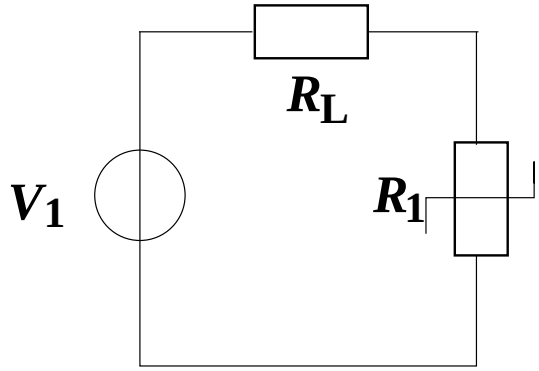


Figura 4.1: Circuito con un elemento no lineal.

El archivo de entrada correspondiente al diagrama de la figura 4.1 es el siguiente:

```
#1 funcion PWL i=f(u) serie
V1 1 0 2
RL 1 2 4
R1 2 0 i=PWL[83.72*u^5-226.31*u^4+229.62*u^3-103.79*u^2+17.76*u,0,1,6]
DCPWL
end
```

Un dato importante del archivo de entrada está en la sintáxis del elemeto no lineal R1, en el cual se puede apreciar que se tomó un rango de 0 a 1 y una aproximación de 6 segmentos.

Después de ejecutar el simulador, se obtuvieron los resultados que se presentan más adelante; los cuales constituyen código generado por MAPLE. Se presenta para este primer ejemplo todos los resultados generados por la simulación, incluyendo el despliege de resultados intermedios, la salida de la etapa de entrada del simulador, las listas más importantes, así como las matrices y vectores lineales y no lineales.

El primer bloque del impresión de resultados muestra la compilación del simulador y el análisis de la sintáxis del archivo de entrada. Si existiera algún error de gramática dentro del netlist del archivo de entrada, aquí es donde sería indicado, con un *mensaje de error* ó con un *warning* (ver apéndice A). Una vez terminada la revisión del netlist, se despliega la información obtenida en forma del listas.

> restart;

> read(‘D:/e/tesisMC/tesiscodigo-chuapj/55aver/simulplw/program/simulator0.txt‘);

```
Warning, the name changecoords has been redefined
Warning, the name GramSchmidt has been rebound
Warning, the protected names norm and trace have been redefined
and unprotected
Warning, the name CrossProduct has been rebound
Warning, the names DotProduct and Transpose have been rebound
Warning, the assigned name Group now has a global binding
Warning, the name Basis has been rebound
```

> checkfile(‘D:/e/tesisMC/tesiscodigo-chuapj/55aver/simulplw/cir/dc/pwlcir/ifu/ifutunnel.cir‘);

```
Reading file: #1 funcion PWL i=f(u) serie
   1 extra line(s) had been reading after "end" label.


   Simulator had read 6 lines.
Concluding reading of the Input File.

   ----------------------------------------------------------------

   PRINTING PARSED ITEMS.
```

$$Anlist = [[DC,\ PWL]]$$

$$Control = []$$

Elements=[[V1, RL, R1], [[1, 0], [1, 2], [2, 0]], [2, 4, i=PWL[83.72*u^5-226.31*u^4+229.62*u^3-103.79*u^2+17.76*u,0,1,6]], []]

$$Names = [V1,\ RL,\ R1]$$

$$Nodes = \{0,\ 1,\ 2\}$$

$$Outlist = []$$

$$Title = \text{“\#1 funcion PWL i=f(u) serie”}$$

Values=[2, 4, [i= - 2.163850480 + 5.592284121 u - 4.528472214 $\left|-\frac{1}{6}+u\right|$ +0,592268505 $\left|-\frac{1}{3}+u\right|$+0,8920833480 $\left|-\frac{1}{2}+u\right|$+0,2468979600 $\left|-\frac{2}{3}+u\right|$ + 2,532639240 $\left|-\frac{5}{6}+u\right|$]]

```
END PRINTING PARSED ITEMS.
STARTING  A N A L Y S E S.

------------------------------------------------------------
```

En el segundo bloque, se despliegan las listas que sirven para ir siguiendo el análisis del circuito, seguidas de la matriz lineal, el vector de variables desconocidas, el vector de excitación, el vector de funciones no lineales dependientes de voltaje, el vector de funciones no lineales dependientes de corriente y finalmente el vector de la o las soluciones del circuito.

```
Circuit list:
```

$$[V1,\ RL,\ R1]$$

$$[[1,\ 0],\ [1,\ 2],\ [2,\ 0]]$$

[2, 4, i=PWL[83.72*u^5-226.31*u^4+229.62*u^3-103.79*u^2+ 17.76*u,0,1,6]]

```
Device Names:
```

$$[V1,\ RL,\ R1]$$

Total nodes:

$$\{0,\ 1,\ 2\}$$

Device Values:

$$[2,\ 4,\ [\text{i= - } 2.163850480 + 5.592284121\,\text{u} - 4.528472214\ \left|-\frac{1}{6}+u\right| +0{,}592268505\ \left|-\frac{1}{3}+u\right| +0{,}8920833480\ \left|-\frac{1}{2}+u\right| +0{,}2468979600\ \left|-\frac{2}{3}+u\right| + 2{,}532639240\ \left|-\frac{5}{6}+u\right|]]$$

Analysis list:

$$[[DC,\ PWL]]$$

Linear matrix:

$$\begin{bmatrix} 1/4 & -1/4 & 1 \\ -1/4 & 1/4 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Vector of unknown variables:

$$\begin{bmatrix} v1 \\ v2 \\ i\_V1 \end{bmatrix}$$

Linear Vector of excitement:

$$\begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

Nonlinear Vector of PWL functions i=f(u):

$$[[0],\ \left[-2{,}163850480 + 5{,}592284121\ v2 - 4{,}528472214\left|-\frac{1}{6} + v2\right|\right.$$
$$+0{,}592268505\left|-\frac{1}{3} + v2\right| + 0{,}8920833480\left|-\frac{1}{2} + v2\right| + 0{,}2468979600\left|-\frac{2}{3} + v2\right|$$
$$\left. + 2{,}532639240\left|-\frac{5}{6} + v2\right|\right],\ [0]]$$

Nonlinear Vector of PWL functions u=f(i):

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Vector of solutions:

$$\begin{bmatrix} \{v1 = 2.,\ i\_V1 = -0{,}4129664266,\ v2 = 0{,}3481342936\} \\ \{v1 = 2.,\ i\_V1 = -0{,}4795312401,\ v2 = 0{,}08187503951\} \\ \{v1 = 2.,\ v2 = 0{,}8655359968,\ i\_V1 = -0{,}2836160008\} \end{bmatrix}$$

Solución Exacta	*Solución PWL*
v2, i_v1	*v2, i_v1*
0.03390742200, 0.4915231445	0.08187503951, 0.4795312401
0.3421963444, 0.4144509139	0.3481342936, 0.4129664266
0.9035366621, 0.2741158345	0.8655359968, 0.2836160008

Tabla 4.1: Comparación de las soluciones exactas y PWL. Circuito con diodo túnel.

Como se puede observar se encontraron tres soluciones para el circuito bajo prueba, esto lo podemos verificar y comparar, graficando las funciones que representan el comportamiento del diodo túnel, la función polinomial y la lineal a trozos, superpuestas con la recta de carga. Ver figura 4.2. La tabla 4.1 da una relación de la solución exacta *vs* la solución de la función PWL, de acuerdo a la gráfica de la figura 4.2.

4.2. Circuito simple con resistencia no lineal dependientes de voltaje

En este apartado se presenta un segundo ejemplo, el cual consta de un circuito ERD, en donde el elemento no lineal toma el comportamiento de un diodo con función exponencial. El diagrama es el mismo que el de la figura 4.1, la diferencia se aprecia a continuación en el archivo de entrada, en el que cambian los valores de los elementos.

```
#1 funcion PWL i=f(u) serie
V1 1 0 1.2
RL 1 2 10
R1 2 0 i=PWL[(exp(u/25E-3)-1)*1E-15,-1.1,1.1,6]
DCPWL
end
```

El rango que se maneja en el elemento PWL es de -1.1 a 1.1 y tiene una aproximación de 6 segmentos.

La impresión de los resultados de la simulación se muestran a continua-

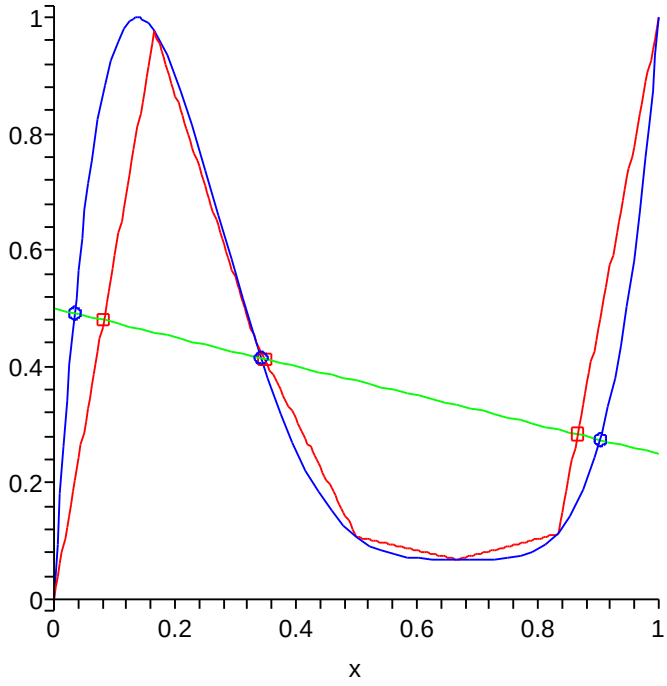


Figura 4.2: Comparción de resultados. Circuito con diodo túnel.

ción:

Listas con la información descrita por el archivo de entrada.

`Circuit list:`

$$[V1,\ RL,\ R1]$$

$$[[1,\ 0],\ [1,\ 2],\ [2,\ 0]]$$

$$[1,2,\ 10,\ i = PWL[(exp(u/25E-3)-1)*1E-15,-1,1,\ 1,1,\ 6]]$$

`Device Values:`

$[1.2,\ 10,\ [\mathrm{i}= -\ 12851.59191 + 17524.90178\,\mathrm{u}$
$+\ 0.5821363635\,10^{-21}\ |0{,}7333333333 + u|$
$+\ 0{,}1363635200\,10^{-14}\ |0{,}3666666666 + u| + 0{,}3194115451\,10^{-8}\ |u|$
$+0{,}007481746976\ |-0{,}366666667 + u| + 17524{,}89430\ |-0{,}733333334 + u|]]$

`Analysis list:`

$$[[DC,\ PWL]]$$

Matriz y vectores que conforman la ecuación de equilibrio.

`Linear matrix:`

$$\begin{bmatrix} \frac{1}{10} & \frac{-1}{10} & 1 \\ \frac{-1}{10} & \frac{1}{10} & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

`Vector of unknown variables:`

$$\begin{bmatrix} v1 \\ v2 \\ i\_V1 \end{bmatrix}$$

`Linear Vector of excitement:`

$$\begin{bmatrix} 0 \\ 0 \\ 1{,}2 \end{bmatrix}$$

`Nonlinear Vector of PWL functions i=f(u):`

$$[[0], [\text{- } 12851.59191 + 17524.90178\,\text{v2} + 0.5821363635\,10^{-21}\ |0{,}7333333333 + v2| + 0{,}1363635200\,10^{-14}\ |0{,}3666666666 + v2| + 0{,}3194115451\,10^{-8}\ |v2| + 0{,}007481746976\ |-0{,}366666667 + v2| + 17524{,}89430\ |-0{,}733333334 + v2|], [0]]$$

Solución Exacta	*Solución PWL*
v2, i_v1	*v2, i_v1*
0.7333345090, 0.04666654910	0.7839792774, 0.04160207226

Tabla 4.2: Comparación de las soluciones exactas y PWL. Circuito ERD.

`Nonlinear Vector of PWL functions u=f(i):`

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Valores numéricos de las variables desconocidas.

`Vector of solutions:`

$$\begin{bmatrix} v2 = 0{,}7333345090 \\ i\_V1 = -0{,}04666654910 \\ v1 = 1{,}200000000 \end{bmatrix}$$

La gráfica de la figura 4.3 ayuda a la comparación de los resultados, los cuales estan plasmados en la tabla 4.2.

4.3. Circuito con dos resistencias no lineales dependientes de voltaje

El ejemplo que se describe a continuación consta de dos elementos no lineales, los cuales tienen un comportamiento exponencial, son dependientes de voltaje, tienen un rango de -1.1 a 1.1 y una aproximación de 6 segmentos. El circuito ejemplo se muestra en la figura 4.4.

El archivo de entrada es el siguiente:

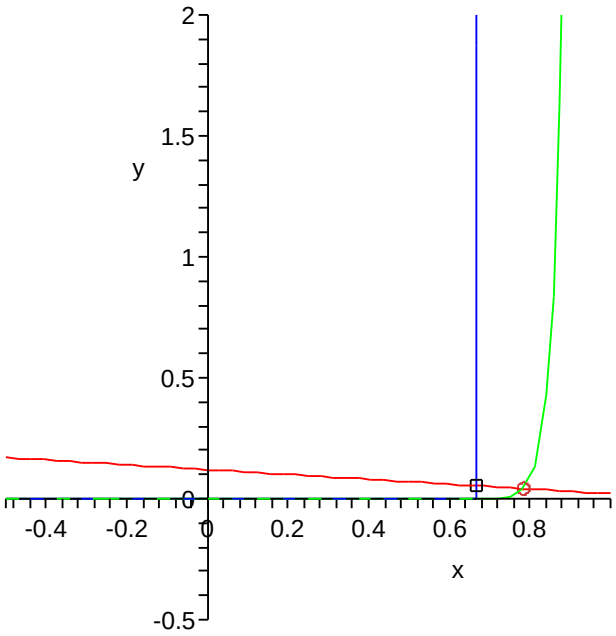


Figura 4.3: Comparción de resultados. Circuito ERD.

```
#2 funciones PWL i=f(u)
V1 1 0 3
RL 1 2 10
R1 2 3 i=PWL[(exp(u/25E-3)-1)*1E-15,-1.1,1.1,6]
R2 3 0 i=PWL[(exp(u/26E-3)-1)*1E-15,-1.1,1.1,6]
DCPWL
end
```

Los resultados obtenidos con la simulación son:

Listas con la información descrita por el archivo de entrada.

```
Circuit list:
```

$$[V1,\ RL,\ R1,\ R2]$$

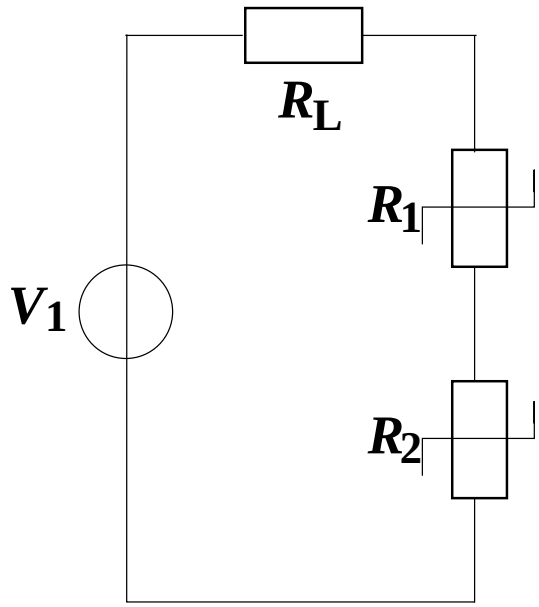


Figura 4.4: Circuito con dos elementos no lineales.

[[1, 0], [1, 2], [2, 3], [3, 0]]

[3, 10, i=PWL[(exp(u/25E-3)-1)*1E-15,-1.1,1.1,6],
i=PWL[(exp(u/26E-3)-1)*1E-15,-1.1,1.1,6]]

`Device Values:`

[3, 10, [i= - 12851.59191 + 17524.90178 u
+ 0.5821363635 10^{-21} $|u + 0{,}7333333333|$
+ 0,1363635200 10^{-14} $|u + 0{,}3666666666|$ + 0,3194115451 10^{-8} $|u|$
+0,007481746976 $|-0{,}366666667 + u|$+17524,89430 $|-0{,}733333334 + u|$],
i= - 2365.902399 + 3226.231752 u + 0.1023409091 10^{-20} $|u + 0{,}7333333333|$
+ 0,1363634317 10^{-14} $|u + 0{,}3666666666|$ + 0,1817036683 10^{-8} $|u|$
+0,002421193355 $|-0{,}366666667 + u|$ + 3226,229331 $|-0{,}733333334 + u|$
]

`Analysis list:`

$[[DC,\ PWL]]$

Matriz y vectores que conforman la ecuación de equilibrio.

`Linear matrix:`

$$\begin{bmatrix} \frac{1}{10} & \frac{-1}{10} & 0 & 0 \\ \frac{-1}{10} & \frac{1}{10} & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

`Vector of unknown variables:`

$$\begin{bmatrix} v2 \\ v1 \\ v3 \\ i\_V1 \end{bmatrix}$$

`Linear Vector of excitement:`

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$

`Nonlinear Vector of PWL functions i=f(u):`

$$[[\text{ - } 12851.59191 + 17524.90178\,\text{v2 - } 17524.90178\ \text{v3} + 0.5821363635\,10^{-21}\ |v2 - v3 + 0{,}7333333333| + 0{,}1363635200\,10^{-14}\ |v2 - v3 + 0{,}3666666666| + 0{,}3194115451\,10^{-8}\ |v2 - v3| + 0{,}007481746976\ |-0{,}366666667 + v2 - v3| + 17524{,}89430\ |-0{,}733333334 + v2 - v3|], [0],$$

$$
\begin{aligned}
&+ 20751.13353\,\mathrm{v3} + 0.1023409091\,10^{-20}\ |v3 + 0{,}7333333333| \\
&+ 0{,}1363634317\,10^{-14}\ |v3 + 0{,}3666666666| + 0{,}1817036683\,10^{-8}\ |v3| \\
&+0{,}002421193355\ |-0{,}366666667 + v3|+3226{,}229331\ |-0{,}733333334 + v3| \\
&- 17524{,}90178\,v2 - 0{,}5821363635\,10^{-21}\ |v2 - v3 + 0{,}7333333333| \\
&-0{,}1363635200\,10^{-14}\ |v2 - v3 + 0{,}3666666666|-0{,}3194115451\,10^{-8}\ |v2 - v3| \\
&- 0{,}007481746976\ |-0{,}366666667 + v2 - v3| \\
&- 17524{,}89430\ |-0{,}733333334 + v2 - v3| \\
&, [0]]
\end{aligned}
$$

`Nonlinear Vector of PWL functions u=f(i):`

$$
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
$$

Valores numéricos de las variables desconocidas.

`Vector of solutions:`

$$
\begin{bmatrix} v3 = 0{,}7333568223 \\ v2 = 1{,}466694374 \\ i\_V1 = -0{,}1533305626 \\ v1 = 3. \end{bmatrix}
$$

4.4. Circuito con transistores

Como cuarto ejemplo se muestra un circuito con transistores, en el simulador estos son substituidos internamente por un modelo PWL [8]. El circuito se presenta en la figura 4.5. El netlist de entrada se presenta a continuación.

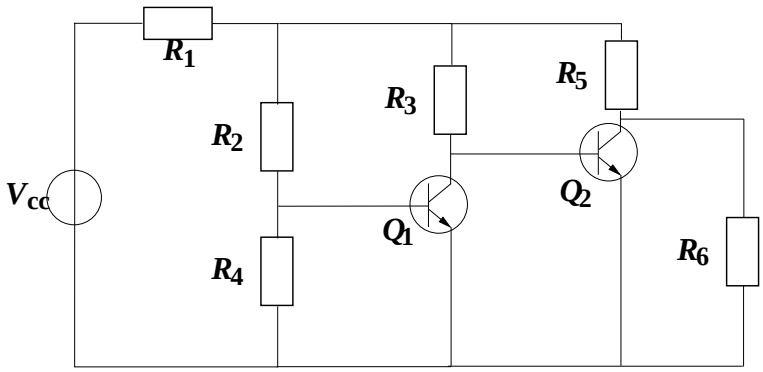


Figura 4.5: Circuito con transistores.

```
#PWL Transistor's Sample
Vcc 5 0 10
R1 1 5 2k
R2 1 2 60k
R4 2 0 3k
R3 1 3 100k
R5 1 4 5k
R6 4 0 50k
Q1 3 2 0 Modelch1
Q2 4 3 0 Modelch1
DCPWL
end
```

Para este caso en particular se aplicó un modelo sencillo, pero la herramienta esta abierta a la utilización de modelos más complejos y con visión a futuro para modelos de pequeña señal.
Los resultados obtenidos con Sim-PWL, en donde podemos observar dos soluciones, son los siguientes:

Listas con la información descrita por el archivo de entrada.

`Circuit list:`

$$[Vcc,\ R1,\ R2,\ R4,\ R3,\ R5,\ R6,\ Q1,\ Q2]$$

$$[[5, 0], [1, 5], [1, 2], [2, 0], [1, 3], [1, 4], [4, 0], [3, 2, 0], [4, 3, 0]]$$

$$[10,\ 2k,\ 60k,\ 3k,\ 100k,\ 5k,\ 50k,\ [Modelch1],\ [Modelch1]]$$

`Device Values:`

$$[10,\ 2000.,\ 60000.,\ 3000.,\ 100000.,\ 5000.,\ 50000.,\ [Modelch1],\ [Modelch1]]$$

`Analysis list:`

$$[[DC,\ PWL]]$$

Vector de variables de desconocidas y vector con los valores numéricos de dichas variables.

`Vector of unknown variables:`

$$\begin{bmatrix} v1 \\ v2 \\ v5 \\ v4 \\ v3 \\ i\_Vcc \\ ic\_FQ1 \\ ic\_FQ2 \end{bmatrix}$$

`Vector of solutions:`

$$\begin{bmatrix} \{v5 = 10., v3 = ,3152798268, ic_FQ1 = ,8548728910e-4, v1 = 6{,}979078111, \\ v2 = ,3152798268, ic_FQ2 = ,1326454060e-2, v4 = ,3152798268, i_Vcc = -,1510460945e-2\} \\ \{v5 = 10., v2 = ,2588412531, v3 = ,2588412531, ic_FQ1 = ,6696391707e-4, \\ v1 = 6{,}961473418, ic_FQ2 = ,1335349608e-2, v4 = ,2588412531, i_Vcc = -,1519263291e-2\} \end{bmatrix}$$

4.5. Circuito con representación Bokohoven

En esta última sección se presenta el netlist de un circuito como el de la figura 4.1, así como el despliege de resultados, que sólo llegan hasta la implementación de Bokhoven como representación del elemento y con opción a gráficar su función PWL.

```
Circuito representacion Bokhoven
V1 1 0 5
RL 1 2 10
R1 3 0 u=BPWL[[2*i+0,[0,1]],[-i+3,[1,2]],[(1/2)*i+0,[2,6]]]
DCPWLB
end
```

$$Anlist = [[DC,\ BPWL]]$$

$$Control = []$$

Elements=[[V1, RL, R1], [[1, 0], [1, 2], [2, 0]],
10, u=BPWL[[2*i+0,[0,1]],[-i+3,[1,2]],[(1/2)*i+0,[2,6]]]
, []]

$$Names = [V1,\ RL,\ R1]$$

$$Nodes = \{1,\ 2,\ 0\}$$

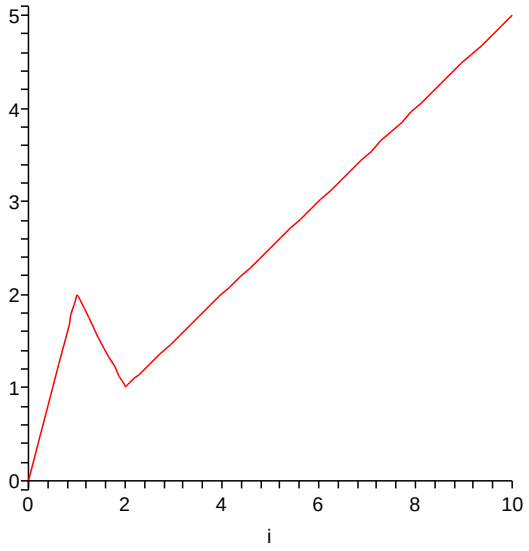
$$Outlist = []$$

$$Title = \text{“Circuito representacion Bokhoven”}$$

Values=

$$\left[5, 10, \left[\begin{array}{c} J_1 \\ J_2 \end{array}\right] = \left[\begin{array}{c} -3 \\ \frac{3}{2} \end{array}\right](x) + \left[\begin{array}{cc} -3 & 0 \\ 0 & \frac{3}{2} \end{array}\right]\left(\left[\begin{array}{c} u_1 \\ u_2 \end{array}\right]\right) + \left[\begin{array}{c} 3 \\ -3 \end{array}\right]\right]$$

```
> plotbok(R4);
```



4.6. Conclusiones

Sim-PWL tiene la ventaja de encontrar más de una solución, como se vió en algunos de los ejemplos anteriormente expuestos. Aunque tiene sus limitantes respecto a circuitos con demasiadas soluciones y en algunos casos, no encuentra todas.

En el último ejemplo se ve como la herramienta puede ser ampliada a más metódos de análisis y de solución, gracias a su modularidad.

Capítulo 5

Conclusiones

Este trabajo, presenta una herramienta de software alternativa para el análisis de circuitos, por medio de métodos PWL. Sim-PWL es un simulador que cuenta con las bondades de la modularidad, de la linearización a trozos y de la utilización de la paltaforma MAPLE. La modularidad del simulador se puede apreciar principalmente en las etapas de formulación de la ecuación de equilibrio en CD, en la de solución numérica y en la de salida, ya que es donde se pueden agregar rutinas para diferentes tipos de análisis, más opciones de modelado, diferentes métodos de solución e impresión de resultados. La etapa de entrada se puede manipular, pero tomando en cuenta las variables descritas en el apéndice B. Las propiedades de convergencia y el poder utilizar matemáticas para ecuaciones lineales en la solución, son factores determinantes para explorar los métodos PWL, así como también la obtención de más de una solución, lo cual es una característica importante del Sim-PWL, ya que por el método convencional de Newton Raphson sólo se obtiene una solución. Aunque las soluciones halladas por Sim-PWL no son exactas, obviamente por que provienen de un modelo aproximado, pero son un buen punto de partida de gran utilidad para los diseñadores.

El uso de MAPLE como plataforma de la herramienta desarrollada, nos da la ventaja de incluir bibliotecas ya existentes y de esa manera agilizar el proceso de programación, en el cual se consumiría mayor tiempo si se utilizara algún lenguaje de alto nivel. Si es cierto que con estos últimos se acelera el

proceso de solución, también es de considerar que un lenguaje con ambientes y blibliotecas prefabricadas, nos dan un rápido conocimiento respecto a la funcionalidad y factibilidad de los algoritmos propuestos.

El lenguaje de entrada del Sim-PWL presenta una gran ventaja al usuario al tener semejanza con el bien conocido *SPICE*. Además de que las diferentes opciones para introducir los valores de los elementos no lineales, da una flexibilidad en cuanto a la cantidad de información, que se tenga del comportamiento de dichos elementos.

Trabajo a futuro

Si bien los objetivos propuestos por la presente tesis han sido cumplidos a cabalidad, la herramienta generada es suceptible de mejoras y futuras ampliaciones.

Por un lado se requiere añadir más representaciones PWL, para tener más opciones en los métodos de solución y se puedan llevar a cabo comparaciones entre ellas.

Por otro lado se puede explotar aún más la característica de múltiples soluciones de la herramienta.

Algunas mejoras pueden ser el realizar barridos de parámetros a componentes y añadir más graficado.

Finalmente, se podrían agregar más tipos de análisis con los que se enriquecería a la herramienta.

Apéndice A

Breve Manual de Usuario

En este apéndice se presenta un breve manual de usuario en el que se explica de manera práctica sobre la forma de introducir los datos al simulador a través de un archivo de entrada con extensión *.cir*, y posteriormente se muestra el funcionamiento del simulador Sim-PWL, bajo la plataforma MAPLE, para llevar a cabo el análisis en CD.

A.1. Elementos Permitidos

Los elementos válidos utilizados para Sim-PWL junto con su respectiva representación son los siguientes:

- Resistor lineal

  Rx n+ n- Valor

- Resistor no lineal

Rx n+ n- u=f(i)
ó
Rx n+ n- u=f(i) i=Valor
ó
Rx n+ n- u=PWL[f(i), min, max, seg]
ó
Rx n+ n- i=PWL[f(u), min, max, seg]
ó
Rx n+ n- u=PWL[[recta-n, min-n, max-n],...]
ó
Rx n+ n- i=PWL[[recta-n, min-n,max-n],...]
ó
Rx n+ n- u=PWL[[x1-n, y1-n, x2-n, y2-n,min-n,max-n],...]
ó
Rx n+ n- i=PWL[[x1-n, y1-n, x2-n, y2-n,min-n,max-n],...]

- Diodo
  Dx n+ n- is=Valor ut=Valor

- Fuente de voltaje independiente
  Vx n+ n- Valor

- Fuente de corriente independiente
  Ix n+ n- Valor

- Fuente de voltaje controlada por voltaje
  Ex n+ n- V(nx,ny) Valor
  ó
  Ex n+ n- V(Xx) Valor

- Fuente de corriente controlada por corriente
  Fx n+ n- I(Xx) Valor
  ó
  Fx n+ n- I(Xx) Valor

- Fuente de corriente controlada por voltaje
  Gx n+ n- V(nx,ny) Valor
  ó
  Gx n+ n- V(Xx) Valor

- Fuente de voltaje controlada por corriente

  Hx n+ n- I(nx,ny) Valor
  ó
  Hx n+ n- I(Xx) Valor

Donde:

n+	⇝	Es el nodo positivo
n-	⇝	Es el nodo negativo
Valor	⇝	Es un valor numérico
u=f(i)	⇝	Es una función dependiente de corriente
i=f(u)	⇝	Es una función dependiente de voltaje
u=PWL[f(i), min, max, seg]	⇝	Es una definión PWL permitida
i=PWL[f(u), min, max, seg]	⇝	Es una definión PWL permitida
u=PWL[[recta-n, min-n, max-n],...]	⇝	Es una definión PWL permitida
i=PWL[[recta-n, min-n, max-n],...]	⇝	Es una definión PWL permitida
u=PWL[[xmin-n, ymin-n, xmax-n, ymax-n],...]	⇝	Es una definión PWL permitida
i=PWL[[xmin-n, ymin-n, xmax-n, ymax-n],...]	⇝	Es una definión PWL permitida
i=valor	⇝	Es el valor inicial de corriente
u=valor	⇝	Es el valor inicial de voltaje
is	⇝	Es la corriente de saturación inversa
ut	⇝	Es el voltaje térmico
V(nx,ny)	⇝	Son los nodos del elemento con el voltaje controlador
Xx	⇝	Es el elemento con el voltaje o corriente controlador(a)

A.1.1. Valores y funciones

Los valores numéricos reales se pueden dar en cifras, notación científica o usando sufijos. A continuación se presentan los sufijos válidos con su

respectiva equivalencia:

a	$1e-18$	E	$1e+18$
f	$1e-15$	P	$1e+15$
p	$1e-12$	T	$1e+12$
n	$1e-9$	G	$1e+9$
u	$1e-6$	M	$1e+6$
m	$1e-3$	K	$1e+3$

Las funciones que se pueden usar están restringidas por el teorema 2 presentado en el capítulo 3, que se resume en los siguientes dos puntos:

- La función debe tener partición lineal.
- Todas las regiones vecinas separadas por una frontera común que tienen una intersección degenarada con otras fronteras deben de cumplir la propiedad de variación consistente.

A.1.2. Archivo de entrada

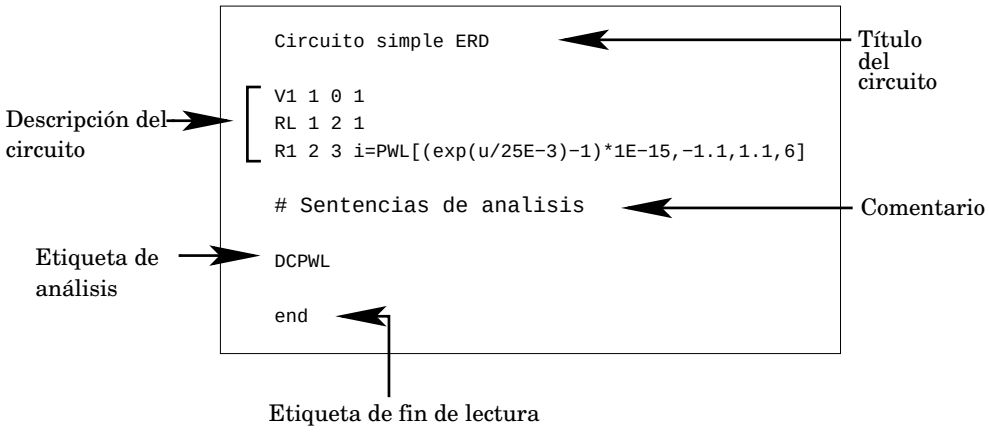


Figura A.1: Archivo de entrada.

Como se puede observar en la fig. A.1, la primera línea del archivo es considerada como el título del circuito a simular y la última línea es la etiqueta *end* que indica el fin de lectura del archivo, todo lo que se escriba después de ésta será ignorado en cuestión de formación de la lista de datos. Entre esas dos líneas se describe el circuito y se dan las sentencias de análisis, tomando en cuenta las siguientes reglas:

- Describir sólo una sentencia por línea.
- Cualquier sentencia debe comenzar en la primera columna.
- Entre diferentes parámetros de una sentencia debe existir por lo menos un espacio.
- Entre sentencias puede haber cualquier número de líneas en blanco o líneas de comentario.
- Las líneas de comentario comienzan con el carácter especial #.

Para realizar la simulación en DC la sentencia de análisis que se debe escribir es `DCPWL`, como se muestra en el ejemplo de la fig. A.1.

A.1.3. Simulación del Sim-PWL en MAPLE

El Simulador se ejecuta bajo la plataforma de MAPLE release 9.5. La estructura básica del archivo ejecutable se muestra en la fig. A.2.

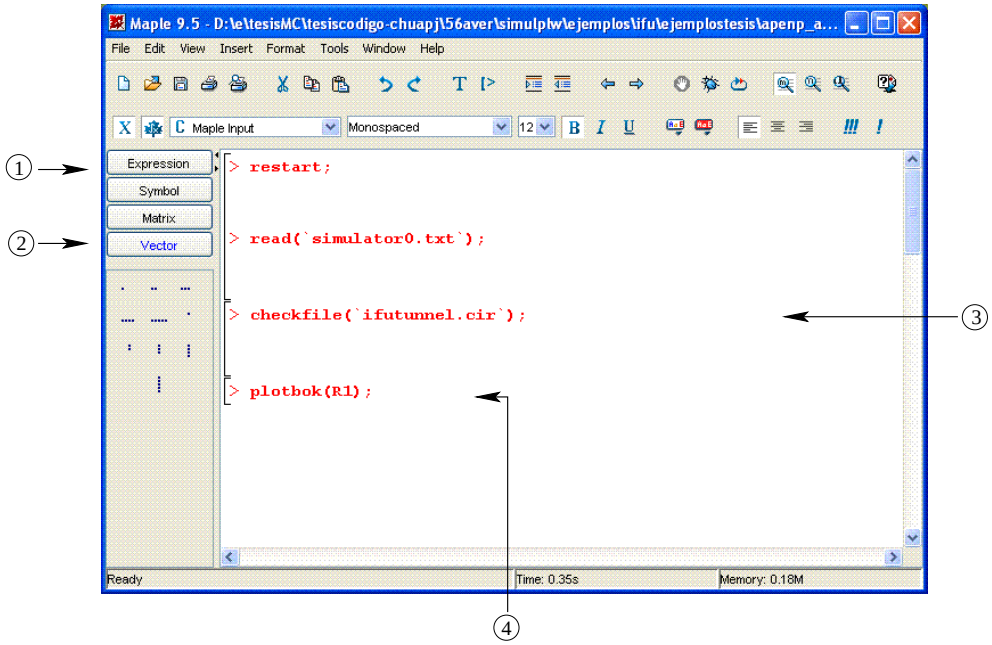


Figura A.2: Archivo de ejecución.

① Grupo de comando que inicializa y restablece al Simulador.

② Grupo de comando que lee los archivos de código del Simulador.

③ Grupo de comando donde se realiza la lectura y análisis del archivo de entrada.

④ Grupo de comando que contiene las opción de gaficado de para la representación Bokhoven.

En caso de que se desee simular otro circuito, no es necesario volver a cargar de nuevo el simulador, basta sólo con cambiar el nombre de archivo de entrada (*.cir*), que aparece en el grupo de comando ③.

Apéndice B

Variables

Este apéndice está dedicado a mostrar las variables que se considera resultarán de utilidad para el usuario así como para trabajos posteriores de mejoramiento de la herramienta.

Variable	tipo	descripción
Elements	*lista de listas*	*Guarda los datos del circuito*
Names	*lista*	*Guarda los nombres de los elementos*
Nodes	*lista de listas*	*Guarda los nodos de los elementos*
Values	*lista*	*Guarda los valores de los elementos*
Control	*lista*	*Guarda las fuentes controladas*
Anlist	*lista*	*Guarda las sentencias de análisis*
Ymna	*Matriz*	*Guarda la matriz MNA*
Jmna	*matriz*	*Guarda el vector estímulo*
Xmna	*matriz*	*Guarda el vector de variables desconocidas*
sol	*set*	*Guarda la ó las soluciones*

Variable	tipo	descripción
ZNLimna	*matriz*	*Guarda el vector de funciones no lineales dependientes de corrientes*
ZNLumna	*matriz*	*Guarda el vector de funciones no lineales dependientes de voltaje*
addtotal	*lista de listas*	*Guarda el sitema de ecuaciones a resolver*
cannonical	+	*Guarda el valor del elemento no lineal convertido a forma canónica de Chua*
ecbok1	+	*Guarda el valor del elmento no lineal de la ecuación de salida de la representación Bokhoven*
ecbok2	+	*Guarda el valor del elmento no lineal de la ecuación de estados de la representación Bokhoven*

Bibliografía

[1] V. M. G. Van Bokhoven. *Piecewise Linear Modelling and analysis*. Kluwer Academic Publisher, 1998.

[2] B. R. Chawla, H. K. Gummel, and P. Kozak. Motis - an mos timing simulator. *IEEE Trans. on Circuits and Systems*, CAS-22(12):910–910, December 1975.

[3] J. M. Chien and E. S. Kuh. Solving piecewise linear equations for resistive networks. *Int. J. Circuit Theory and Applications*, 4:3–24, 1976.

[4] L. O. Chua and A. C. Deng. Canonical piecewise-linear representation. *IEEE Trans. on Circuits and Systems*, 35:101–111, January 1988.

[5] L. O. Chua and S. M. Kang. Section-wise piecewise-linear functions: Canonical representation, properties and applications. *Proc. IEEE*, 65(6):915–929, January 1977.

[6] L. O. Chua and P. M. Lin. *Computater aided analysis of electronic circuits: algorithms y computational techniques*. Prentice–Hall, 1975.

[7] L. O. Chua and R. P. Ying. Canonical piecewise-linear analysis. *IEEE Trans. on Circuits and Systems*, 30(3):125–140, March 1983.

[8] Leon O. Chua and Robin L.P. Ying. Finding all solutions of piecewise-linear circuits. *Circuit Theory and Applications*, 10:201–209, July 1982.

[9] H. de Man, G. Arnout, and P. Reynaert. Mix-mode circuit simulation techniques and their implementation in diana. *Computer Design Aids for VLSI Circuits*, ed. H. de Man:113–174, 1980.

[10] Ying-Wei Jan and J.A. Starzyk. A simulation program emphasized on dc analysis of vlsi circuits:samoc. *Southwest Symposium on Mixed-Signal Design, 1999*, pages 174–179, April 1999.

[11] P. Julian, A. Desages, and O. Agamennoni. High level canonical piecewise-linear representation using a simplicial partition. *IEEE Trans. on Circuits and Systems - I*, 46(4):463–480, April 1999.

[12] P. M. Julián. *A High Level Canonical Representation: Theory and Applications*. PhD thesis, Universidad Nacional del Sur, 1999.

[13] S. M. Kang and L. O. Chua. A global representation of multidimensional piecewise-linear functions with linear partitions. *IEEE Trans. on Circuits and Systems*, 25(11):938–940, November 1978.

[14] T. Kevenaar and D. Leenaerts. A comparison of piecewise-linear model descriptions. *IEEE Trans. on Circuits and Systems - I*, 39(12):996–1004, December 1992.

[15] T. A. M. Kevenaar. *PLANET - A Hierarchial Network Simulator*. PhD thesis, Technische Universiteit Eindhoven, 1992.

[16] T. A. M. Kevenaar and D. M. W. Leenaerts. A flexible hierarchical piecewise linear simulator. *Integration, the VLSI Journal*, 12:211–235, 1991.

[17] Kahrs M., Levitan S.P., Chiarulli D.M., Kurzweg T.P., Martinez J.A., Boles J., Davare A.J., Jackson E., Windish C., Kiamilev F., Bhaduri A., Taufik M., Xingle Wang, Morris A.S., Kruchowski J., and Gilbert B.K. System-level modeling and simulation of the 10g optoelectronic interconnect. *Journal of Lightwave Technology*, 21(12):3244–3256, December 2003.

[18] A. R. Newton. Techniques for the simulation of large-scale integrated circuits. *IEEE Trans. on Circuits and Systems*, CAS-26(9):741–749, September 1979.

[19] Sarmiento Reyes L. A. *A partition method for the determination of multiple DC operating points*. PhD thesis, Delft University of Technology, May 1994.

[20] Sarmiento Reyes L. A. *A partition method for the determination of multiple DC operating points*. PhD thesis, Delft University of Technology, May 1994.

[21] Sarmiento Reyes L. A. Network theory: Part 1 (a preliminary version). Unpublished, 1995.

[22] A. Sedra and K. C. Smith. *Dispositivos eletrónicos y amplificación de señales*. Mc Graw Hill, 1989.

[23] Synopsys. Saber, library and model user guide. Technical report, 2006.

[24] G. van der Laan. On the existence and approximation of zeroes, mathematical programming. *Mathematical Programming*, (28):1–14, 1984.

[25] J. T. J. van Eijndhoven. *A Piecewise Linear Simulator for Large Scale Integrated Circuits*. PhD thesis, Technische Universiteit Eindhoven, 1984.

[26] M. T. van Stiphout. *PLATO - A Piecewise Linear Analysis Tool for Mixed Level Circuit Simulation*. PhD thesis, Technische Universiteit Eindhoven, 1990.

[27] Rocío De Jesús Ventura, Lizbeth Ruz Armas, Jonathan Valencia Santa Rosa, Luis Hernández-Martínez, and Arturo Sarmiento-Reyes. A symbolic frequency-domain simulation program for an analogue design environment. *Iberchip, 2006*, pages 254–259, March 2006.

[28] J. Vlach and K. Singhal. *Computer Methods for Circuit Analysis and Design*. Van Nostrand Reinhold Company, 1983.