ORDENAMIENTO ÓPTIMO USANDO EL ALGORITMO RECOCIDO SIMULADO Y ALMACENAMIENTO COMPACTO APLICADOS A LA SOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES

Jorge Mario Arias Palacio
Julián David González Hoyos

Universidad Tecnológica de Pereira
Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación
Programa de Ingeniería Eléctrica
Pereira
2008

ORDENAMIENTO ÓPTIMO USANDO EL ALGORITMO RECOCIDO SIMULADO Y ALMACENAMIENTO COMPACTO APLICADOS A LA SOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES

Proyecto de Grado para Optar al Título de
Ingeniero Electricista

Jorge Mario Arias Palacio
Julián David González Hoyos

Director
PhD.Ramón Alfonso Gallego Rendón

Universidad Tecnológica de Pereira
Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación
Programa de Ingeniería Eléctrica
Pereira
2008

Calificación:

MSc. Alejandro Garcés Ruiz
Jurado

PhD. Ramón Alfonso Gallego Rendón
Director

A mi familia por el gran amor y apoyo incondicional que me han brindado.

A mis padres, mi hermano, mi abuela y Angela.

Agradecimientos

A Dios, por haberme dado una hermosa familia, unos grandes maestros, unos importantes compañeros. Al apoyo incondicional del PhD. Ramón Alfonso Gallego, quien confió en el prospero desarrollo de este proyecto desde el primer instante, a sus valiosas explicaciones y correctos aportes. Deseo dar un agradecimiento muy especial al Mg. Alejandro Garcés, a sus grandes y acertadas recomendaciones. Agradezco también a mi maestro y amigo PhD. Juan José Mora Florez, por sus inolvidables enseñanzas, por todo su apoyo. Al Ing. Jorge Eduardo Calle, maestro de maestros, a sus excelentes cátedras, por hacer ver la ingeniería Eléctrica como un maravilloso mundo de posibilidades. A todos y cada uno de los partícipes de mi formación académica, a todos ellos mil gracias por todo lo aprendido, nunca los olvidaré.
Para finalizar, quiero agradecer a mis tres mejores y más grandes maestros y amigos, a quienes me han enseñado lo más importante de esta vida,a quienes nunca han dejado de creer en mi y de la mano me han acompañado de la forma más dulce en este mundo: a mi padre Henry Arias Patiño, a mi madre María del Socorro Palacio Floréz y a mi preciosa hemanita Angela María Arias Palacio, gracias Dios por tan maravillosa familia, los amo.

Primero que todo a Dios, por permitir que todo esto sea posible. Agradezco también a ese gran hombre que es mi padre: Jorge Iván González, pues gracias a su apoyo incondicional y confianza depositada supo formarme en grandes valores; a mi maravillosa madre María Edilia Hoyos, quien con sabios consejos y su abnegada labor en el hogar supo dirigir con sabiduría esa barca llamada familia; a mi muy querido hermano Oscar Iván González, quien a partir de su experiencia profesional en el sector eléctrico y al entrañable cariño que nos une, aportó en gran medida a la buena toma de decisiones trascendentales en este arduo ejercicio. Agradezco también al MSc. Alejandro Garcés por su desinteresada asesoría y colaboración en muchas áreas y por acompañar de cerca todo este proceso. En fin, mil gracias a todos aquellos que aportaron a mi formación académica.

Índice general

Capítulo Primero

Introducción

Gran número de fenómenos y procesos reales que se analizan en ingeniería son modelados a través de sistemas de ecuaciones lineales de gran tamaño, en cuyos casos, requieren de la intervención de un gran número de diferentes variables que, a su vez, se relacionan entre sí. En el caso específico del modelamiento de sistemas de potencia a gran escala, donde se analizan redes del orden de las centenas y hasta miles de barras, las matrices encargadas de almacenar los coeficientes que acompañan las variables del sistema, presentan en general, un gran número de elementos nulos. Esta característica es llamada *dispersidad*, la cual permite realizar cierto tipo de manipulaciones en cuanto al orden en que las variables serán procesadas y al almacenamiento computacional de la información.

Debido a la característica dispersa de las matrices que intervienen en procesos eléctricos, es de vital importancia aprovechar de manera adecuada las ventajas que trae el hecho de tener que trabajar con pocos elementos no nulos. Como es sabido, no es eficiente almacenar elementos con valor igual a cero, ya que requiere de un gran espacio en memoria y de un exigente esfuerzo computacional innecesario. Por ejemplo, considere que un sistema eléctrico de 1000 barras tiene una matriz de admitancia nodal con el 85 % de sus elementos iguales a cero, el cual es un valor normal de dispersidad. Si se deseara almacenar la totalidad de los elementos de la matriz, sería necesario guardar 1 millón de posiciones, lo cual contrasta con la aplicación de un esquema de almacenamiento, en donde sólo sería necesario almacenar 150 mil elementos más una información adicional que proporcione la ubicación de los mismos. Este tipo de manipulación ayuda no sólo a disminuir el espacio requerido en memoria, sino también a reducir el número de operaciones que deben ser realizadas, pues como se sabe, cualquier número multiplicado por cero es igual cero.

En segunda instancia, si lo que se desea es obtener el máximo beneficio computacional, es claro que paralelo al almacenamiento compacto de la matriz de coeficientes, se debe realizar un ordenamiento óptimo del sistema, el cual consiste en darle un nuevo orden a la matriz, de tal manera que los elementos de "relleno" que surgen del proceso de solución sean muy pocos, conservando así la dispersidad. En otras otras palabras, el ordenamiento óptimo se refiere a organizar las variables (columnas) en la forma adecuada para la cual, los elementos que inicialmente son cero se conserven así tras el proceso

de inversión de la matriz.

En la literatura especializada es común encontrar técnicas para el ordenamiento basadas en desarrollos matemáticos exactos, los cuales arrojan buenos resultados. Sin embargo, en el presente proyecto, la búsqueda del orden óptimo estará a cargo de una técnica de optimización combinatorial llamada Simulated Annealing. Tal herramienta se fundamenta en el proceso físico del recocido de sólidos, el cual consiste en alcanzar una temperatura máxima antes de la fundición del sólido, punto en el cual la temperatura debe ser disminuida paulatinamente pasando por estados de energía subóptimos hasta alcanzar una estructura cristalina perfecta, estado en el cual la energía libre es mínima. De acuerdo a lo anterior, el algoritmo básico del Simulated Annealing emplea una temperatura inicial que depende del valor que se le asigne a la longitud de la cadena e implementa un programa de enfriamiento donde se deben tomar algunas decisiones de forma aleatoria. Finalmente, existen varias formas de establecer el criterio de parada, lo cual depende del programador y del problema que debe ser solucionado.

Así mismo, se sabe de la existencia de diferentes formas para solucionar un sistema de ecuaciones lineales; unos requieren de invertir explícitamente la matriz de coeficientes, otros lo logran factorizando ésta. Computacionalmente, el hecho de invertir una matriz es bastante costoso, debido a ello, se prefieren métodos como la factorización **LU**, **LDU** y Cholesky. El método de solución seleccionado para este proyecto es el de la factorización **LDU**, no obstante, cabe aclarar que su aplicación dista de ser la convencional, pues como se verá a lo largo del trabajo, ya no se cuenta con un arreglo matricial —debido al almacenamiento compacto—, sino con una serie de vectores que contienen la información suficiente para llevar a cabo con éxito el proceso de solución.

Tal cual se desprende del título del proyecto, el problema consiste en hallar el orden óptimo de la matriz de coeficientes y almacenar sólamente la información necesaria para, finalmente, llevar a cabo el proceso de solución del sistema de ecuaciones lineales. Para ello, se ha desarrollado el presente documento que reune en cada capítulo las bases teóricas de cada una de las temáticas que se abordan en esta investigación.

El capítulo 2 introduce al lector a los sistemas de ecuaciones lineales, los conceptos fundamentales del álgebra matricial, la descomposición triangular y, por primera vez en el trabajo, se aborda el tema de las matrices dispersas. Después de la lectura del más extenso de los capítulos, se tendrán muy claras las ideas básicas y las motivaciones de la presente investigación.

En el capítulo 3 se realiza un recuento detallado, con ejemplos ilustrativos y de manera organizada de diferentes esquemas de almacenamiento nodal, resaltando las ventajas y desventajas de cada uno. El lector tendrá la posibilidad de ir observando la evolución de cada uno de estos métodos, los cuales van mejorando ciertas características que dan paso a nuevos esquemas más completos y eficientes desde el punto de vista computacional.

La presentación del capítulo 4 se realiza alrededor de la discusión de diferentes téc-

nicas de ordenamiento matricial. Se presentan los conceptos fundamentales de cada una de ellas, la motivación de encontrar un nuevo orden a una matriz y se analizan diversos casos de aplicación. Después de haber leído esta sección, se debe tener claro el esquema fundamental de la presente investigación y, obviamente, se comprenderá completamente el problema.

El capítulo 5 presenta una breve explicación de la motivación que se presenta actualmente en el ámbito académico sobre el uso de algoritmos heurísticos en problemas de optimización. Se plantea la base de muchos de estos algoritmos y sobre el que se fundamenta el Simulated Annealing: *la búsqueda local*. Además de ello —que en realidad es sólo el principio—, se explica el Simulated Annealing básico, la analogía con el proceso físico del recocido de sólidos, el algoritmo de Metropolis y la implementación del método.

Se ha destinado el capítulo 6 para exponer la metodología general que se propone en el presente proyecto, diseñándolo de tal manera que el lector pueda detallar cada una de la etapas que hacen parte de la investigación. En esta sección, además de ilustrar el modelo matemático, se aclara la selección de los valores de los parámetros que conforman el programa de optimización, también la forma en que se realiza el empaquetamiento de la matriz de coeficientes y, por último, el algoritmo de solución del sistema compacto. Además de ello, se ha destinado una parte de este capítulo a proponer tres ejemplos que ilustren el comportamiento del ordenamiento óptimo aplicado a un sistema radial, a un sistema lineal que se compara con la solución arrojada al no estar ordenado y en la solución de un problema específico y frecuente de corto circuito, el cual consiste en la obtención de los elementos de la matriz $\mathbf{Z}_{\mathbf{BUS}}$ a partir del almacenamiento de la $\mathbf{Y}_{\mathbf{BUS}}$.

Como en todo proyecto, se deben realizar pruebas que le den validez a la metodología que se propone y que permitan realizar un análisis detallado de cada uno de los resultados que se obtienen tras su aplicación. Es por eso que se destina el capítulo 7 a explicar la forma en que se realizaron las pruebas, al análisis de los resultados que se obtuvieron y a establecer los criterios con que se evalúa la validez de lo obtenido.

Finalmente, se incluye una sección destinada a la presentación de conclusiones y ciertas recomendaciones que apuntan a la posibilidad de continuar la temática abordada en este trabajo.

Capítulo Segundo

Sistemas De Ecuaciones Lineales

Desde el punto de vista de la ingeniería y a partir de una fundamentación matemática, el modelamiento de problemas reales conduce en la mayoría de los casos a solucionar sistemas de ecuaciones lineales que representan el fenómeno analizado. Por tanto, el planteamiento de estos sistemas de ecuaciones algebraicos deben respetar ciertas características ya establecidas en su forma teórica, para que la solución del mismo presente una respuesta que se acerque en lo posible a los niveles prácticos del problema, es decir, que la respuesta del sistema sea lo más confiable posible.

Dentro del proceso de darle solución a los sistema de ecuaciones, han surgido una gran variedad de técnicas, dentro de las cuales sobresale la **Regla de Cramer** y la cononocida **Eliminación de Gauss**; sin embargo, por características que se analizarán dentro del desarrollo de este capítulo, su implementación es costosa en materia computacional y la forma de solución del proceso debe derivarse en una metodología más económica, en lo que a *espacio de memoria* y *tiempo de cómputo* se refiere.

El objeto que tiene la inclusión de este capítulo dentro del desarrollo del proyecto, es precisamente brindar claridad al lector de la teoría básica de los sistemas de ecuaciones lineales, preparando un panorama para la correcta comprensión en el desarrollo de los posteriores capítulos. Para ello, este capítulo se ha dividido en 5 secciones. La primera sección describe las definiciones y conceptos básicos del álgebra matricial. La segunda sección, se encarga por su parte de la descripción de la eliminación de Gauss, concepto de vital importancia en el desarrollo de este proyecto. La tercera sección se ha destinado a la correcta comprensión de la factorización triangular, haciendo un desarrollo especial para el análisis de matrices dispersas. La sección cuarta se ha destinado para analizar el error relativo entre la respuesta ideal del sistema y la respuesta generada al aplicar una técnica apropiada de inversión. Para finalizar, se hace una breve introducción al análisis de matrices dispersas, enfocando su desarrollo al entendimiento general del problema.

2.1. Definiciones y Teoría Básica

2.1.1. Matriz

Una matriz **A** es un arreglo rectangular de números o funciones de la forma:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{2.1}$$

Si una matriz tiene m renglones y n columnas, su *tamaño* es de m por n (se escribe $m \times n$). Una matriz de $n \times n$ se llama matriz **cuadrada** de orden n.

El elemento del *i-ésimo* renglón y de la *j-ésima* columna de la matriz **A** de $m \times n$ se representa por a_{ij}. Con ello, una matriz **A** de $m \times n$ se representa en la forma $\mathbf{A} = (a_{ij})_{m\times n}$, o simplemente $\mathbf{A} = (a_{ij})$. Una matriz de *1* $\times$ *1* es sólo una constante o función.

2.1.2. Igualdad de Matrices

Dos matrices **A** y **B** de $m \times n$, son **iguales** si $a_{ij} = b_{ij}$ para toda i y j.

2.1.3. Matriz Columna

Una **matriz columna X** es cualquier matriz que tenga n renglones y una columna.

$$\mathbf{X} = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} = (b_{j1})_{n\times 1} \tag{2.2}$$

Una matriz columna también es llamada **vector columna** o simplemente vector.

2.1.4. Multiplos de Matrices

Un **multiplo** de una matriz **A** se define como sigue:

$$k\mathbf{A} = \begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix} = (ka_{ij})_{m\times n} \tag{2.3}$$

en donde k es una constante o función.

2.1.5. Suma de Matrices

La **suma** de dos matrices $\mathbf{A}$ y $\mathbf{B}$ de $m \times n$ se define como la matriz:

$$\mathbf{A} + \mathbf{B} = (a_{ij} + b_{ij})_{m \times n} \tag{2.4}$$

En otras palabras, para sumar dos matrices del mismo tamaño, se suman los elementos correspondientes.

La **diferencia** de dos matrices de $m \times n$ se define en la forma acostumbrada

$$\mathbf{A} - \mathbf{B} = \mathbf{A} + (-\mathbf{B}) \tag{2.5}$$

en donde $(-\mathbf{B}) = (-1) \cdot \mathbf{B}$.

2.1.6. Multiplicación de Matrices

Sea $\mathbf{A}$ una matriz con m renglones y n columnas, y $\mathbf{B}$ una matriz con n renglones y p columnas. El **producto** $\mathbf{A} \cdot \mathbf{B}$ se define como la matriz $m \times p$ dada por:

$$\mathbf{A} \cdot \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix} \tag{2.6}$$

$$\mathbf{A} \cdot \mathbf{B} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1n}b_{n1} & \cdots & a_{11}b_{1p} + a_{12}b_{2p} + \cdots + a_{1n}b_{np} \\ a_{21}b_{11} + a_{22}b_{21} + \cdots + a_{2n}b_{n1} & \cdots & a_{21}b_{1p} + a_{22}b_{2p} + \cdots + a_{2n}b_{np} \\ \vdots & & \vdots \\ a_{m1}b_{11} + a_{m2}b_{21} + \cdots + a_{mn}b_{n1} & \cdots & a_{m1}b_{1p} + a_{m2}b_{2p} + \cdots + a_{mn}b_{np} \end{bmatrix}$$

$$\mathbf{A} \cdot \mathbf{B} = \left(\sum_{k=1}^{n} a_{ik} b_{kj} \right)_{m \times p} \tag{2.7}$$

Obsérvese detenidamente en la definición 2.1.6, que el producto $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$ está definido sólo cuando el número de columnas en la matriz $\mathbf{A}$ es igual al número de renglones de la matriz $\mathbf{B}$. El tamaño del producto se determina con:

$$\mathbf{A}_{m \times n} \mathbf{B}_{n \times p} = \mathbf{C}_{m \times p} \tag{2.8}$$

También se reconocerá que los elementos del *i-ésimo* renglón de la matriz producto $\mathbf{A} \cdot \mathbf{B}$ se forma aplicando la definición en componentes de producto interior, o producto

punto, del *i-ésimo* renglón de $\mathbf{A}$ con cada una de las columnas de $\mathbf{B}$.

En general, la multiplicaciòn matricial no es conmutativa, esto es, $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$.

Identidad Multiplicativa. Para un entero positivo n, la matriz de $n \times n$

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2.9}$$

es la *matriz identidad multiplicativa*. Según la definición 2.1.6, para toda matriz $\mathbf{A}$ de $n \times n$

$$\mathbf{A} \cdot \mathbf{I} = \mathbf{I} \cdot \mathbf{A} = \mathbf{A} \tag{2.10}$$

También se comprueba con facilidad que, si $\mathbf{X}$ es una matriz columna, entonces $\mathbf{I} \cdot \mathbf{X} = \mathbf{X}$.

Matriz Cero. Una matriz formada sólo por elementos cero se conoce como matriz cero y se representa por $\mathbf{0}$. Si $\mathbf{A}$ y $\mathbf{0}$ son matrices de $m \times n$, entonces

$$\mathbf{A} + \mathbf{0} = \mathbf{0} + \mathbf{A} = \mathbf{A} \tag{2.11}$$

Propiedad Distributiva. Si todos los productos están definidos, la multiplicación matricial es distributiva respecto a la suma:

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC} \tag{2.12}$$

Propiedad Asociativa. Aunque no se demostrará, la multiplicación matricial es asociativa. Si $\mathbf{A}$ es una matriz de $n \times p$, $\mathbf{B}$ una matriz $p \times r$ y $\mathbf{C}$ una matriz de $r \times n$, entonces

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} \tag{2.13}$$

es una matriz de $m \times n$.

Determinante de una Matriz. Para toda matriz cuadrada $\mathbf{A}$ de constantes hay un número asociado, llamado determinante de la matriz, que se representa como $detA$.

Se demuestra que un determinante, $det\mathbf{A}$ se puede desarrollar por cofactores usando cualquier renglón o cualquier columna. Si $\mathbf{A}$ tiene un renglón o una columna con muchos elementos cero, el sentido común aconseja desarrollar el determinante por ese renglón o columna.

2.1.7. Transpuesta de una Matriz

La transpuesta de la matriz 2.1 de $m \times n$ es la matriz $\mathbf{A}^T$ de $n \times m$ representada por:

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix} \tag{2.14}$$

En otras palabras, los renglones de una matriz $\mathbf{A}$ se convierten en las columnas de su transpuesta $\mathbf{A}^T$.

2.1.8. Inversa Multiplicativa de una Matriz

Sea $\mathbf{A}$ una matriz de $n \times n$. Si existe una matriz $\mathbf{B}$ de $n \times n$, tal que:

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}$$

en donde $\mathbf{I}$ es la matriz identidad multiplicativa, se dice que $\mathbf{B}$ es la inversa multiplicativa de $\mathbf{A}$, la cual se representa como $\mathbf{B} = \mathbf{A}^{-1}$.

2.1.9. Matrices No Singulares y Singulares

Sea $\mathbf{A}$ una matriz de $n \times n$. Si $det\mathbf{A} \neq 0$, se dice que $\mathbf{A}$ es **no singular**. En caso contrario se dice que $\mathbf{A}$ es **singular**.

El siguiente teorema especifica una condición necesaria y suficiente para que una matriz cuadrada tenga inversa multiplicativa.

Teorema 1 *Una matriz* $\mathbf{A}$ *de* $n \times n$ *tiene una inversa multiplicativa* $\mathbf{A}^{-1}$ *si y sólo si* $\mathbf{A}$ *es no singular.*

El siguiente teorema describe un método para determinar la inversa multiplicativa de una matriz no singular.

Teorema 2 *Sea* $\mathbf{A}$ *una matriz no singular de* $n \times n$ *y* $C_{ij} = (-1)^{i+j} M_{ij}$*, donde* M_{ij} *es el determinante de la matriz* $(n-1) \times (n-1)$*, obtenido al eliminar el i-ésimo renglón y la j-ésima columna de* $\mathbf{A}$*, entonces*

$$\mathbf{A}^{-1} = \frac{1}{det\mathbf{A}} (C_{ij})^T \tag{2.15}$$

Cada C_{ij}, en el teorema 2, es tan sólo el **cofactor**, o con *menor signo*, del elemento a_{ij} correspondiente en $\mathbf{A}$. Obsérvese que en la ecuación 2.15 se utiliza la transpuesta.

La ecuación 2.15 pierde eficiencia cuando las matrices no singulares son mayores a 3×3. Por ejemplo, para aplicarla a una matriz 4×4, se necesitaría calcular **dieciseis** determinantes 3×3. Cuando una matriz es grande, hay métodos más eficientes para calcular $\mathbf{A}^{-1}$ de forma indirecta. Estos métodos se analizarán en la siguiente sección.

2.2. Eliminación de Gauss

Las matrices son una ayuda insustituible para resolver sistemas algebraicos de n ecuaciones lineales con n incognitas.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned} \tag{2.16}$$

Si **A** representa la matriz de los coeficientes en 2.16, se sabe que es posible usar la **regla de Cramer** o evaluar la inversa de **A** a través del teorema 2para resolver el sistema, siempre y cuando el $det\mathbf{A} \neq 0$. Sin embargo, ya se ha demostrado que el trabajo que se requiere al utilizar estas reglas es grande si la matriz **A** es mayor a 3×3. La metodología que se describe a continuación tiene la ventaja de no sólo ser un método eficiente para manejar sistemas grandes, sino tambien una forma de resolver sistemas inconsistentes.

El método que se ha mencionado es denominado **eliminación de Gauss**, el cual es uno de los mejores algoritmos conocidos y se basa en el hecho de que es posible transformar un sistema algebraico de ecuaciones en un sistema equivalente (es decir, un sistema que tiene la misma solución), multiplicando una ecuación por una constante distinta de cero, intercambiando el orden de dos ecuaciones cualesquiera del sistema y sumando un múltiplo constante de una ecuación a otra.

Considerese ahora el sistema presentado en 2.16, y denótese las componentes b_i del lado derecho de las ecuaciones como $a_{i,n+1}$. Esto simplifica la notación del desarrollo que se realizará. El sistema comienza

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= a_{1,n+1} \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= a_{2,n+1} \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= a_{n,n+1} \end{aligned} \tag{2.17}$$

Dividiendo la primera ecuación del sistema 2.17 por a_{11} se obtiene:

$$x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n = a_{1,n+1}^{(1)} \tag{2.18}$$

La notación significa que $a_{12}^{(1)} = a_{12}/a_{11}$, etc. Multiplicando la ecuación 2.18 por $-a_{21}$ y sumándola a la segunda ecuación del sistema en 2.17, se obtienen los nuevos coeficientes

de esta ecuación.

$$a_{2j}^{(1)} = a_{2j} - a_{21}a_{1j}^{(1)} \Rightarrow \forall j = 1, 2, ..., n+1 \tag{2.19}$$

La selección de este factor asegura que $a_{21}^{(1)}$ sea igual a cero. De manera similar para las ecuaciones posteriores se tiene:

$$a_{ij}^{(1)} = a_{ij} - a_{i1}a_{1j}^{(1)} \tag{2.20}$$

La cual es válida para $i = 2, 3, ..., n$ y $j = 2, 3, ..., n+1$ y hace que todos los elementos de la primera columna sean iguales a cero con la exepción de $a_{11}^{(1)}$, el cual es igual a 1.

La ejecución conscutiva de cada una de estas operaciones genera el siguiente sistema equivalente:

$$\begin{aligned} x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n &= a_{1,n+1}^{(1)} \\ a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n &= a_{2,n+1}^{(1)} \\ &\vdots \\ a_{n2}^{(1)}x_2 + \cdots + a_{nn}^{(1)}x_n &= a_{n,n+1}^{(1)} \end{aligned} \tag{2.21}$$

El siguiete paso excluye la primera línea y columna y aplica el mismo procedimiento a las ecuaciones restantes. La formulación continuaría con:

$$\begin{aligned} a_{kj}^{(k)} &= a_{kj}^{(k-1)} / a_{kk}^{(k-1)} \\ a_{ij}^{(k)} &= a_{ij}^{(k-1)} - a_{ik}^{(k-1)}a_{kj}^{(k)} \end{aligned} \tag{2.22}$$

Para $k = 1, 2, ..., n$, $i = k+1, ..., n$ y $j = 3, 4, ..., n+1$. Las ecuaciones ahora adquieren la siguiente forma:

$$\begin{aligned} x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n &= a_{1,n+1}^{(1)} \\ x_2 + a_{13}^{(2)}x_3 + \cdots + a_{2n}^{(2)}x_n &= a_{2,n+1}^{(2)} \\ x_3 + \cdots + a_{3n}^{(3)}x_n &= a_{3,n+1}^{(3)} \\ &\vdots \\ x_n &= a_{3,n+1}^{(n)} \end{aligned} \tag{2.23}$$

El proceso mediante el cual se obtiene la solución del sistema se denomina **sustitución regersiva**. La última variable, x_n, es utilizada en la ecuación $(n-1)$ para obtener el valor numérico de x_{n-1} y así sucesivamente. En general la sustitución regresiva es:

$$x_i = a_{i,n-1} - \sum_{j=i+1}^{n} a_{ij} x_j \tag{2.24}$$

donde $i = n-1, n-2, ..., 1$.

En términos computacionales, el costo de este algoritmo es calculado según el número de operaciones realizadas. Cada operación comienza con la combinación de una multiplicación seguida de una sustración. Es posible demostrar que la eliminación gausiana requier de $\approx n^3/3$ operaciones hasta llegar a plantear la matriz triangular superior y una vez en este punto, requiere de $\approx n^2/2$ operaciones en el proceso de sustitución regresiva, donde n representa el rango de la matriz. Es posible mejorar estos índices operativos haciendo uso de otra metodología que se fundamenta en la eliminación de Gauss, representando las operaciones elementales entre filas como productos de matrices elementales, esto es, la descomposición triangular.

2.3. Descomposición Triangular

En muchas aplicaciones, la mejor manera de desarrollar la solución de un sistema algebraico de ecuaciones es por la *descomposición triangular* o técnica de *factorización* **LU**. El algoritmo para la factorización triangular es derivado de la eliminación gausiana, aunque su forma computacional se desarrolla en una secuencia diferente.

La idea fundamental se basa en que es posible representar la matriz de coeficientes del sistema de ecuaciones presentado en 2.16 como:

$$\mathbf{A} = \mathbf{LU} \tag{2.25}$$

donde

$$\mathbf{L} = \begin{bmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ l_{31} & l_{32} & l_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \tag{2.26}$$

y

$$\mathbf{U} = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1n} \\ & 1 & u_{23} & \cdots & u_{2n} \\ & & 1 & \cdots & u_{3n} \\ & & & \ddots & \vdots \\ & & & & 1 \end{bmatrix} \tag{2.27}$$

Como se puede notar, $\mathbf{L}$ es una matriz *triangular inferior* y $\mathbf{U}$ es una matriz *triangular superior*. Obsérvese los elementos iguales a uno en la diagonal de la matriz $\mathbf{U}$. Esto significa que el determinante de $\mathbf{A}$ es obtenido como el producto de los elementos l_{ii} de la matriz $\mathbf{L}$.

Se derivará un algoritmo para encontrar $\mathbf{L}$ y $\mathbf{U}$ un poco más adelante, por el momento, se asumirá que esta descomposición es posible. El sistema de ecuaciones puede ser escrito de la siguiente forma:

$$\mathbf{LUx} = \mathbf{b} \tag{2.28}$$

Se define un vector auxiliar $\mathbf{z}$ como:

$$\mathbf{Ux} = \mathbf{z} \tag{2.29}$$

En este momento, $\mathbf{z}$ no puede ser calculado porque $\mathbf{x}$ es desconocido. Sin embargo, la sustitución de $\mathbf{z}$ en 2.28 genera:

$$\mathbf{Lz} = \mathbf{b} \tag{2.30}$$

Debido a la estructura que posee la matriz $\mathbf{L}$, el vector $\mathbf{z}$ puede ser calculado de una manera muy simple. Escribiendo el producto en forma de ecuaciones:

$$\begin{aligned} l_{11}z_1 &= b_1 \\ l_{21}z_1 + l_{22}z_2 &= b_2 \\ l_{31}z_1 + l_{32}z_2 + l_{33}z_3 &= b_3 \\ &\vdots \\ l_{n1}z_1 + l_{n2}z_2 + \cdots + l_{nn}z_n &= b_n \end{aligned}$$

Comenzando por la primera ecuación, es posible obtener la solución de la siguiente manera:

$$\begin{aligned} z_1 &= b_1/l_{11} \\ z_2 &= \left(b_2 - l_{21}z_1\right)/l_{22} \\ z_3 &= \left(b_3 - l_{31}z_1 - l_{32}z_2\right)/l_{33} \end{aligned}$$

y, en general,

$$z_1 = b_1/l_{11} \tag{2.31}$$

y los demas elementos se calculan como:

$$z_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij}z_j\right)/l_{ii} \tag{2.32}$$

Este es el denominado proceso de *sustitución hacia adelante*. La diagonal de los elementos en $\mathbf{L}$ deben ser diferentes de cero en concordancia con 2.31 y 2.32. Ahora, en el proceso de *sustitución regresiva* se dará solución a las variables desconocidas del vector $\mathbf{x}$ a través de la relación 2.29. Escribiendo la ecuación 2.29 en forma de ecuaciones se tiene:

$$\begin{aligned} x_1 + u_{12}x_2 + u_{13}x_3 + \cdots + u_{1n}x_n &= z_1 \\ x_2 + u_{23}x_3 + \cdots + u_{2n}x_n &= z_2 \\ &\vdots \\ x_{n-1} + u_{n-1,n}x_n &= z_{n-1} \\ x_n &= z_n \end{aligned}$$

Comenzando con la última ecuación y sustituyendo hacia arriba en el sistema anterior, se obtienen las siguientes dos relaciones:

$$x_n = z_n \tag{2.33}$$

La cual es la relacion de punto de partida. Para la solución de las demás variables del vector x se emplearía entonces:

$$x_i = z_i - \sum_{j=i+1}^{n} u_{ij} z_j; \qquad i = n-1, n-2, ..., 1. \tag{2.34}$$

El número total de operaciones requeridas para llevar a cabo este proceso de sustitución regresiva es aproximadamente de $n^2/2$, para un total de n^2 para la solución general. Un minucioso examen de la ecuación 2.32 revela que b_i es utilizado sólo para el cálculo de z_i y no es requerido después. Similarmente, en las ecuaciones 2.33 y 2.34, z_i no se requiere despues de que x_i ha sido calculado. Estas variables b, z y x, por sus características operativas dentro de la descomposición, pueden compartir la misma memoria en una implementación computacional. Nótese además la equivalencia en el proceso de sustitución regresiva existente en las ecuaciones 2.34) y 2.24.

Ahora se derivará el algoritmo para la factorización **LU**. Considere una matriz 4×4, y asuma que la factorización existe al igual que el producto de **L** y **U**:

$$\begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} & l_{11}u_{14} \\ l_{21} & l_{21}u_{12} + l_{22} & l_{21}u_{13} + l_{22}u_{23} & l_{21}u_{14} + l_{22}u_{24} \\ l_{31} & l_{31}u_{12} + l_{32} & l_{31}u_{13} + l_{32}u_{23} + l_{33} & l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} \\ l_{41} & l_{41}u_{12} + l_{42} & l_{41}u_{13} + l_{42}u_{23} + l_{43} & l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + l_{44} \end{bmatrix}$$

Al comparar este producto con la matriz **A** de 4×4, se puede observar con claridad que la primera columna de la descomposición permanece inalterable y que todo $l_{i1} = a_{i1}$, $i = 1, 2, 3, 4$. Se hace evidente que la primera fila del producto puede ser utilizada para obtener la primera fila de la matriz **U**, resolviendo $l_{11}u_{1j} = a_{1j}$ para las variables desconocidas u, $j = 2, 3, 4$. En el siguiente paso es posible notar que la segunda columna contiene únicamente a u_{12} y los valores conocidos l_{i1}, de modo que los valores de l_{i2}, $i = 2, 3, 4...$, pueden ser calculados de la forma:

$$l_{i2} = a_{i2} - l_{i1}u_{12}; \qquad i = 2, 3, 4.$$

Así, conocidos l_{21}, l_{22}, y u_{1j}, la segunda fila puede ser usada para calcular u_{2j}, $j = 3, 4$:

$$u_{2j} = \left(a_{2j} - l_{21}u_{1j}\right)/l_{22}; \qquad i = 3, 4.$$

El algoritmo procede de la misma forma alternando filas y columnas. En la *k-ésima* iteración de la factorización, las entradas de la fila k de **U** en la columna k de **L** es

calculada. Para derivar un algoritmo formalmente, considere nuevamente el producto de **L** con **U**. Se requiere que:

$$a_{ij} = \sum_{m=1}^{k} l_{im}u_{mj} = \sum_{m=1}^{min(i,j)} l_{im}u_{mj}$$

Donde el límite superior en la suma es cambiado para reflejar los zeros en **L** y **U**. Considerando un elemento dentro o por debajo de la diagonal, $i \geq j$, y utilizando el índice k en lugar de j:

$$a_{ik} = \sum_{m=1}^{k} l_{im}u_{mk} = l_{ik} + \sum_{m=1}^{k-1} l_{im}u_{mk}$$

como $u_{kk} = 1$. Reescribiendo se obtiene:

$$l_{ik} = a_{ik} - \sum_{m=1}^{k-1} l_{im}u_{mk}; \qquad i \geq k. \tag{2.35}$$

Similarmente, conside la entrada de un elemento por encima del triangulo, $i < j$, y defina el índice k en lugar a i:

$$a_{kj} = \sum_{m=1}^{k} l_{km}u_{mj} = l_{kk}u_{kj} + \sum_{m=1}^{k-1} l_{km}u_{mj}$$

Reorganizando términos se genera:

$$u_{kj} = \left(a_{kj} - \sum_{m=1}^{k-1} l_{km}u_{mj} \right) / l_{kk}; \qquad j > k. \tag{2.36}$$

Las ecuaciones 2.35 y 2.36 constituyen el *algoritmo de Crout* para la descomposición triangular. El algoritmo se desarrolla comenzando $k = 1, 2, ..., n$ y usando 2.35 y 2.36. Nótese que en estas ecuaciones las entradas de **L** y **U** son calculados en etapas previas del proceso. Además, cada entrada de a_{ij} es requerida solamente para el calculo del correspondiente **L** o **U**, dependiendo principalmente si $i \geq j$ o $i < j$.Como las entradas de elementos iguales a cero en **L** y **U** y las entradas de los elementos de la diagonal unitaria

de **U** no necesitan ser calculados, en implementaciones computacionales las matrices **L** y **U** describen a **A**, con **L** y **U** ocupando el triangulo inferior y superior de **A**, respectivamente. El algoritmo puede ser resumido de la siguiente manera:

1. Iniciar *k=1* e ir al paso 3.
2. Usar (2.35) para calcular la columna k de **L**. Si $k=n$, parar.
3. Usar (2.36) para calcular la fila k de **U**.
4. Hacer *k=k+1* y regresar al paso 2.

Otra presentación del algoritmo se puede obtener por observación de entradas de la matriz descompuesta en sus factores **L** y **U**. Para ello, escriba la descomposición en la forma de una matriz independiente, de la siguiente manera:

$$\begin{bmatrix} l_{11} & a_{12}l_{11} & a_{13}/l_{11} & a_{14}/l_{11} \\ l_{21} & a_{22}-l_{21}u_{12} & (a_{23}-l_{21}u_{13})/l_{22} & (a_{24}-l_{21}u_{14})/l_{22} \\ l_{31} & a_{32}-l_{31}u_{12} & a_{33}-l_{31}u_{13}-l_{32}u_{23}+l_{33} & (a_{34}-l_{31}u_{14}-l_{32}u_{24})/l_{33} \\ l_{41} & a_{42}-l_{41}u_{12} & a_{43}-l_{41}u_{13}-l_{42}u_{23} & a_{44}-l_{41}u_{14}-l_{42}u_{24}-l_{43}u_{34} \end{bmatrix} \tag{2.37}$$

La primera columna y la primera línea son calculadas como antes. En las restantes entradas se observa que la primera sustración es posible de realizar inmediatamente se conozcan todos los l_{i1} y u_{1j}. Suponga que se han realizado estas restas. Al hacerlo, la segunda columna se convierte en el resultado final para l_{i2}, $i = 2, 3, ..., n$. Similarmente, en la segunda fila todo lo que queda por hacer es dividir todas las entradas por l_{22}. Este paso corresponde a las operaciones que se realizan al inicio del proceso con la diferencia que el índice comienza en 2.De nuevo se nota que en la matriz restante todos los segundos términos pueden sustraersen y así sucesivamente. El algoritmo de descomposición por lo tanto, puede ser declarado como sigue:

1. Hacer *k=1.*
2. $l_{ik} = a_{ik}; \qquad i \geq k,$
3. $u_{kj} = a_{kj}/l_{kk}; \qquad j > k,$
4. $a_{ij} = a_{ij} - l_{ik}u_{kj}; \qquad i, j > k,$
5. Si $k=n$, pare; en caso contrario hacer *k=k+1* y regresar al paso 2.

Las características más importantes de la descomposición triangular son las siguien tes:

1. Es posible calcular de forma rápida el determinante de la matriz

$$det\mathbf{A} = \prod_{i=1}^{n} l_{ii} \tag{2.38}$$

2. Las matrices descompuestas **L**, **U** pueden sobreescribir los valores anteriores de **A** y ser almacenados en el mismo espacio (no hay necesidad de almacenar la diagonal unitaría de **U**).

3. Si solamente el vector del lado derecho del sistema b cambia, no es necesario recalcular la descomposición y solamente el proceso de sustitución regresiva variaría.

4. El sistema transpuesto de la forma $\mathbf{A}^T x = c$ puede ser resuelto usando los mismos factores triangulares. Esto requeriría de un análisis de sensibilidad del sistema.

El número de operaciones requeridas para la factorizació **LU** está dada por:

$$M = \sum_{j=1}^{n-1} \left[(n-j) + (n-j)^2 \right] = \frac{n^3}{3} - \frac{n}{3}.$$

En términos de cálculos necesarios, la descomposición **LU** es equivalente a la eliminación de Gauss; su ventaja radica en los puntos tercero y cuarto anteriormente mencionados.

Las redes que consisten exclusivamente de elementos pasivos y fuentes independientes en redes eléctricas, suelen dar lugar a sistemas de ecuaciones representados por una matriz simétrica. En este caso el algoritmo de la descomposición triangular puede ser modificada para generar la factorización $\mathbf{A} = \mathbf{D}^T\mathbf{DU}$, donde **U** es una matriz triangular unitaria y **D** es diagonal. El almacenamiento requerido y el costo de sus cálculos puede ser reducido casi a la mitad comparado con la descomposición **LU**.

2.4. Teoría de la Perturbación

La teroría de la perturbación estudia la magnitud de los cambios que suelen presentarsen en el proceso de solucionar un sistema de ecuaciones algebraicos, y es importante comprender que estas perturbaciones son generadas al aplicar operaciones elementales al sistema original como se hace en la *eliminación de Gauss* y en la *descompocición* **LU**.

El objetivo de esta sección no es la de ilustrar un desarrollo completo matemáticamente, sino por su parte, generar una visión de lo importante que es minimizar el número de operaciones realizadas al sistema de ecuaciones, para que la desviación de la

respuesta sea lo menos pronunciada posible.

Para lo anterior, cononsidere un sistema de ecuaciones lineales algebraicas de la forma:

$$\mathbf{A}x = b \tag{2.39}$$

se sabe que la solución general de las variables desconocidas x está determinada en su forma explicita por $x = \mathbf{A}^{-1}b$. Considerando ahora una perturbación en b, es decir, se variará b de su forma natural a $b + \Delta b$, se obtiene:

$$\mathbf{A}\left(x + \Delta x\right) = b + \Delta b \tag{2.40}$$

sustituyendo 2.39 en 2.40, se genera la siguiete relación:

$$\mathbf{A}\Delta x = \Delta b \tag{2.41}$$

lo que es igual a tener:

$$\Delta x = \mathbf{A}^{-1}\Delta b \tag{2.42}$$

aplicando propiedades básicas de normas de vectores y matrices se puede demostrar que:

$$\frac{\|\Delta x\|}{\|\Delta b\|} \leq \left\|\mathbf{A}^{-1}\right\| \tag{2.43}$$

Generalmente interesa obtener un error relativo en la respuesta, para ello se utiliza la siguiente expresión:

$$\frac{\|\Delta x\|}{\|x\|} \leq \left\|\mathbf{A}^{-1}\right\| \frac{\|\Delta b\|}{\|\mathbf{A}\|^{-1}\|b\|} = \|\mathbf{A}\|\left\|\mathbf{A}^{-1}\right\| \frac{\|\Delta b\|}{\|b\|} \tag{2.44}$$

La cantidad $k(\mathbf{A}) = \|\mathbf{A}\|\left\|\mathbf{A}^{-1}\right\|$ es denominada *número condicional* de $\mathbf{A}$ y para cualquier norma $k(\mathbf{A}) \geq 1$. Esta relación de amplificación mide los efectos generados en el vector solución, debido a perturbaciones en el lado derecho. Grandes valores de $k(\mathbf{A})$ implica que pequeños cambios en b, como los que se producen en implementaciones computacionales, puedan desviar de forma significativa la solución del sistema.

Considerando ahora una perturbación en la matriz de coeficientes $\mathbf{A}$ se genera el siguiente sistema:

$$(\mathbf{A} + \Delta\mathbf{A})(x + \Delta x) = b \tag{2.45}$$

Simplificando la ecuación 2.45 se obtiene:

$$(\mathbf{A} + \Delta\mathbf{A})\Delta x = -\Delta\mathbf{A} \cdot x \tag{2.46}$$

Asumiendo que $\mathbf{A}$ es no singular (que es lo que siempre se hace), $(\mathbf{A} + \Delta\mathbf{A})$ puede ser singular si $\Delta\mathbf{A}$ no se hace restricto. Escribiendo

$$(\mathbf{A} + \Delta\mathbf{A}) = \mathbf{A}\left(\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A}\right) \tag{2.47}$$

Se puede observar que $(\mathbf{A} + \Delta\mathbf{A})$ es sin duda no singular si

$$\left\|\mathbf{A}^{-1}\Delta\mathbf{A}\right\| \leq 1 \tag{2.48}$$

Asumiendo esta condición como satisfecha, se puede escribir la ecuación 2.46 como:

$$\Delta x = -\left(\mathbf{I} + \mathbf{A}^{-1}\Delta\mathbf{A}\right)^{-1}\mathbf{A}^{-1}\Delta\mathbf{A} \cdot x \tag{2.49}$$

Nuevamente, aplicando propiedades de norma de vectores y matrices se llega a la siguiente expresión:

$$\|\Delta x\| \leq \frac{\left\|\mathbf{A}^{-1}\Delta\mathbf{A}\right\| \|x\|}{1 - \|\mathbf{A}^{-1}\Delta\mathbf{A}\|} \leq \frac{\left\|\mathbf{A}^{-1}\right\| \|\Delta\mathbf{A}\| \|x\|}{1 - \|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\|} \tag{2.50}$$

El error relativo puede derivarse en forma sencilla de la ecuación 2.50 con lo cual se genera:

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\| \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}}{1 - \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}} \tag{2.51}$$

Nuevamente, la cantidad $k(\mathbf{A}) = \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|$ es un factor decisivo para determinar el error existente entre la respuesta ideal y la generada en el proceso de inversión.

Como se pudo observar, al manipular sistemas de ecuaciones algebraicos lineales, se genera siempre un error relativo entre la respuesta ideal del sistema y la que se obtiene a travéz de algún proceso de inversión. Este error puede disminuirse minimizando el número de operaciones aplicadas al sistema, como se ha mencionado con anterioridad. Para ello, es necesario aplicar técnicas especiales. Una de las recomendaciones más factibles es el *almacenamiento compacto de matrices* explicado en el *capítulo 3* y el *ordenamiento nodal* explicado en el *capítulo 4* del presente proyecto, en el cual, se hace un análisis de las más interesantes metodologías encontradas en la literatura especializada que servirán de base para explicar a profundidad la metodología que se propone.

2.5. Principios de Matrices Dispersas

Se ha recalcado que el número de operaciones necesarias para resolver un sistemas de ecuaciones por la *eliminación de Gauss* o por la *descomposición* **LU** es aproximadamente $n^3/3$. Esto es válido si todas las operaciones son realizadas, incluso aquellas operaciones en la que se ha multiplicado o sumado (restado) un cero. Dado a que es posible obtener el número de operaciones aplicadas a un sistema, no es posible por su parte, desviar el desarrollo de una de estas operaciones cuando se sume o multiplique por cero, por tanto, para superar esta dificultad, se hace necesario no almacenar estos elementos, reduciendo los *requerimientos de memoria*.

Considere una red eléctrica conpuesta por n barras y b ramas conformadas por elementos pasivos. La matriz $\mathbf{Y}$ contiene al menos $n + 2b$ entradas diferentes de cero. En una típica red eléctrica, el número de elementos es de dos a cuatro veces mayos que el número de nodos. Claramente, para este tipo de redes, el número total de elementos no nulos en la matriz es una pequeña fracción de n^2, el número total de elementos de la matriz.

Una matriz en la cual la gran mayoría de sus entradas (elementos) son iguales a cero es denominada *matriz dispersa*. Los algoritmos de las matrices dispersas no almacenan ni realizan las operaciones con ceros. El desarrollo de estos algoritmos son mucho más complicados que los sencillos métodos explicados en las secciones anteriores, pero el ahorro en tiempo computacional y memoria en el almacenamiento es sustancial.

Para motivar las estategias operativas de las matrices dispersas, se usará un ejemplo sencillo. Considere una matriz con dos entradas iguales a cero:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & 0 \\ a_{31} & 0 & a_{33} \end{bmatrix}$$

Asuma que se ha desarrollado la descomposición **LU**. Usando la forma explicita 2.37 se

obtiene:

$$\mathbf{LU} = \begin{bmatrix} l_{11} = a_{11} & u_{12} = a_{12}/l_{11} & u_{13} = a_{13}/l_{11} \\ l_{21} = a_{21} & l_{22} = a_{22} - l_{21}u_{12} & u_{23} = (0 - l_{21}u_{13})/l_{22} \\ l_{31} = a_{31} & l_{32} = 0 - l_{31}u_{12} & l_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} \end{bmatrix}$$

Note que en **LU** todas las entradas son diferentes de cero. Un *fill-in* es creado en una posición que originalmente era igual a cero pero que en **LU** es diferente de cero. Asuma ahora que el primer elemento al cual se le efecturá el picote es a_{33}, esto se logra intercambiando la fila y columna tres con la fila y columna uno:

$$\begin{bmatrix} a_{33} & 0 & a_{31} \\ 0 & a_{22} & a_{21} \\ a_{13} & a_{12} & a_{11} \end{bmatrix}$$

La descomposición **LU** es:

$$\mathbf{LU} = \begin{bmatrix} l_{11} = a_{33} & u_{12} = 0 & u_{13} = a_{31}/l_{11} \\ l_{21} = 0 & l_{22} = a_{22} & u_{23} = a_{21}/l_{22} \\ l_{31} = a_{31} & l_{32} = a_{12} & l_{33} = a_{11} - l_{31}u_{13} - l_{32}u_{23} \end{bmatrix}$$

Se puede observar que los ceros de la matriz han sido conservados. La descomposición de la matriz original requiere de 8 multiplicaciones (divisiones) y 5 sustracciones. La modificación de la matriz requiere solamente de 4 multiplicaciones (divisiones) y 2 sustracciones. Este ejemplo muestra claramente que *el número de operaciones puede ser reducido a través de la apropiada selección del elemento pivote*. En la operación con matrices dispersas, la selección del elemento pivote es denominada *ordenamiento matricial*. Este tema será tratado a profundidad en el *capítulo 4* como bien se aclaró en la sección anterior.

Capítulo Tercero

Almacenamiento Compacto de Matrices Dispersas

Gran número de las matrices empleadas en el área del Análisis de Sistemas de Potencia son de gran tamaño y presentan una característica específica: la mayoría de sus elementos son iguales a cero. Esta particularidad se conoce comunmente como *dispersidad*. Una matriz dispersa es aquella matriz para la cual es ventajosa la utilización de una técnica especial dado el hecho de que muchos de sus elementos son iguales a cero, para fines de economía de memoria y cálculo [1].

La utilización masiva de los ordenadores y el aumento constante de su capacidad y velocidad de cálculo, han permitido que los estudios científicos, tecnológicos y de ingeniería utilicen cada vez más modelos matemáticos para interpretar, simular y optimizar fenómenos de diversa complejidad [2]. Debido a ello, esos modelos crecen en magnitud y exactitud. Muchos de estos modelos conllevan enfrentarse con sistemas de ecuaciones de un tamaño tal —decenas o cientos de miles de variables— que hace sólo unos pocos años era casi inimaginable que se pudiesen tratar.

El almacenamiento compacto de matrices se refiere a la idea de almacenar los elementos no nulos de una matriz y de una información adicional, tal como: las posiciones (filas y columnas) de los elementos no nulos, encadenamiento entre los elementos almacenados y diversos apuntadores [3].

Todo esto hace parte de lo que se conoce como *técnicas de dispersidad*, las cuales se basan alrededor de tres principios básicos:

- Minimizar la cantidad de datos a almacenar
- Minimizar el número de operaciones a realizar
- Conservar la dispersidad

El tema que se va a desarrollar en el presente capítulo gira en torno al primer principio de las técnicas de dispersidad, es decir, el de la optimización de la información que debe ser almacenada en memoria.

3.1. Grado de Dispersidad

Como se dijo al principio de este capítulo, en el modelamiento de redes eléctricas es usual encontrar matrices con un alto *grado de dispersidad*. El grado de dispersidad corresponde al porcentaje de elementos nulos de una matriz y está dado por

$$GD = \frac{\text{Número de elementos nulos}}{\text{Número total de elementos}} \times 100\,\% \tag{3.1}$$

Para un sistema con NB barras y NR ramas, la matriz de admitancia nodal Y_{bus} presentará el siguiente grado de dispersidad

$$GD(Y_{bus}) = \frac{NB^2-(NB+2NR)}{NB^2} \times 100\,\% \tag{3.2}$$

y entre mayor sea el tamaño del sistema, el grado de dispersidad aumenta. Por ejemplo, en la tabla 3.1 se puede observar claramente esta característica.

Nº Barras	Número de Ramas	Grado de Dispersidad (%)
10	20	50
100	200	95
1000	2000	99,5

Cuadro 3.1: Valores típicos de dispersidad en Y_{bus}

Como se puede ver, el grado de dispersidad de la matriz de admitancia está íntimamente relacionado con el tamaño del sistema modelado.
En la siguiente sección se ilustran los esquemas de almacenamiento compacto más empleados en el análisis matricial de los sistemas de energía eléctrica.

3.2. Esquemas de Almacenamiento Compacto

La idea básica de estos esquemas consiste en almacenar sólamente los elementos no nulos de la matriz (más alguna información adicional) utilizando un conjunto de vectores y apuntadores, de tal forma que el espacio total de memoria utilizado sea menor que el requerido para almacenar toda la matriz.

Existen muchos esquemas propuestos para el almacenamiento de matrices dispersas:

- se han propuesto esquemas para matrices simétricas y asimétricas
- algunos presentan facilidad para alterar algunos de sus elementos
- otros presentan facilidad para realizar operaciones con matrices
- sus eficiencias son diferentes

La selección del esquema de almacenamiento a ser utilizado depende del problema que se quiera resolver. La eficiencia en la resolución del problema puede variar en función del esquema utilizado [4].

3.2.1. Esquema de almacenamiento por coordenadas

Es la forma más *intuitiva* de realizar un almacenamiento compacto y se compone de un conjunto ordenado o desordenado de tripletas $(a_{ij},\ i,\ j)$, donde $a_{ij} \neq 0$, que definen las coordenadas de cada elemento de la matriz.

Suponga que se desea almacenar la siguiente matriz

$\mathbf{A} =$

	1	**2**	**3**	**4**	**5**
1	x			x	
2	x		x		x
3		x			
4		x		x	
5	x		x		x

se definen entonces cuatro vectores:

Posición	1	2	3	4	5	6	7	8	9	10	11
AN	a_{11}	a_{14}	a_{21}	a_{23}	a_{25}	a_{32}	a_{42}	a_{44}	a_{51}	a_{53}	a_{55}
I	1	1	2	2	2	3	4	4	5	5	5
J	1	4	1	3	5	2	2	4	1	3	5

Como se puede ver, lo elementos no nulos se numeraron de forma organizada por filas, partiendo desde la primera. Estas son las funciones de cada uno de los vectores:

Posición $\longrightarrow$ *numeración elementos no nulos*
AN $\longrightarrow$ *elementos no nulos*
I $\longrightarrow$ *fila*
J $\longrightarrow$ *columna*

Este esquema apenas se usa. Su interés reside en que Matlab utiliza una estrategia similar para designar qué elementos son distintos de cero en una matriz de este tipo.

3.2.2. Esquema de almacenamiento por filas o columnas

Este es uno de los esquemas más difundidos para matrices sin ninguna estructura en particular y además es muy aplicado en programas especializados en matrices dispersas. Considere nuevamente la matriz de la sección anterior.

$\mathbf{A} =$

	1	**2**	**3**	**4**	**5**
1	x			x	
2	x		x		x
3		x			
4		x		x	
5	x		x		x

La variante por filas de este método (*Compressed Row Storage*) requiere de definir tres vectores: el primero, por ejemplo **AN**, debe contener todos los elementos de la matriz distintos de cero, agrupados por filas; el segundo **J** —de la misma dimensión de **AN**—, agrupa los subíndices columna de los elementos de **AN**; el tercero, un vector de punteros,

llamado **IA**, de dimensión $n+1$, contiene la posición del primer elemento no nulo de las filas que se corresponden con el orden de los elementos de **IA**.
Si a la matriz anterior se le aplicara el método por fila para almacenarla, sería necesario guardar la siguiente información:

Posición	1	2	3	4	5	6	7	8	9	10	11
AN	a_{11}	a_{14}	a_{21}	a_{23}	a_{25}	a_{32}	a_{42}	a_{44}	a_{51}	a_{53}	a_{55}
J	1	4	1	3	5	2	2	4	1	3	5
IA	1	3	6	7	9	12					

Observaciones:

- La dimensión de **IA** debe ser $n+1$, pues es necesario definir el número de elementos no nulos de la última fila n.
- El almacenamiento de un nuevo elemento no nulo que se crease a lo largo de un proceso de manipulación de la matriz sería complicado: habría que redefinir gran parte de la estructura.

3.2.3. Esquema de Knuth

El esquema de Knuth es uno de los esquemas más fáciles de aplicar debido a la forma simple de su desarrollo; sin embargo, también es uno de los que almacena mayor información, lo cual lo puede volver ineficiente.

Considere la siguiente matriz:

$\mathbf{A} =$

	1	**2**	**3**	**4**
1		x		
2	x	x		x
3	x			
4		x		x

En ella se puede observar que la mayoría de sus elementos son iguales a cero. Si se aplicara el esquema de Knuth para almacenar la anterior matriz, se obtendrían los siguientes vectores:

Posición	1	2	3	4	5	6	7
AN	a_{12}	a_{21}	a_{22}	a_{24}	a_{31}	a_{42}	a_{44}
I	1	2	2	2	3	4	4
J	2	1	2	4	1	2	4
NR	0	3	4	0	0	7	0
NC	3	5	6	7	0	0	0
JR	1	2	5	6			
JC	2	1	0	4			

Para obtener los anteriores vectores, se enumeraron los elementos diferentes a cero en orden descendente, iniciando por el elemento a_{12}. Esta numeración se puede realizar de manera aleatoria, ya que el método aplicado así lo permite. No obstante, en lo sucesivo

del capítulo, por facilidad, los elementos no nulos se numerarán de la misma forma.

A continuación se describen las funciones de cada uno de los vectores empleados por el esquema:

Posición	$\longrightarrow$	*numeración elementos no nulos*
AN	$\longrightarrow$	*elementos no nulos*
I	$\longrightarrow$	*fila*
J	$\longrightarrow$	*columna*
NR	$\longrightarrow$	*posición del próximo elemento de la fila*
NC	$\longrightarrow$	*posición del próximo elemento de la columna*
JR	$\longrightarrow$	*apuntador de inicio de fila*
JC	$\longrightarrow$	*apuntador de inicio de columna*

Observaciones:

- La posición de cada elemento del vector AN es almacenada en dos vectores I (fila) y J (columna).
- Para facilitar la obtención de elementos de una cierta fila o columna de la matriz, es necesario almacenar también:
  - Un par de apuntadores (**NR** y **NC**) que indican las posiciones de los próximos elementos no nulos en la fila y columna respectivamente.
  - Apuntadores de inicio de fila (**JR**) y columna (**JC**).
- Para cada elemento no nulo de **A** es necesario almacenar 5 valores, además de dos apuntadores de inicio de fila y columna.
- Ventajas del esquema de Knuth:
  - Se pueden agregar o eliminar elementos fácilmente.
  - Se pueden barrer las líneas y columnas fácilmnete.

3.2.4. Esquema Circular KRM *(Knuth-Rheinboldt-Mesztenyi)*

Éste método se basa en el esquema de Knuth y lo modifica de tal manera que la información a almacenar sea menor. Su nombre se debe a la forma en que elige el siguiente elemento de la fila o columna, la cual sugiere una trayectoria circular. A manera de ilustración, considere la matriz **A** de la sección anterior:

$\mathbf{A} =$

	1	**2**	**3**	**4**
1		x		
2	x	x		x
3	x			
4		x		x

Aplicando el esquema circular KRM a la anterior matriz se llega al siguiente conjunto de vectores a almacenar:

Posición	1	2	3	4	5	6	7
AN	a_{12}	a_{21}	a_{22}	a_{24}	a_{31}	a_{42}	a_{44}
NR	1	3	4	2	5	7	6
NC	3	5	6	7	2	1	4
JR	1	2	5	6			
JC	2	1	0	4			

Como se puede observar, ya no es necesario almacenar los vectores fila **I** y columna **J**, ya que esta información se encuentra embebida en la restante. Si se analiza detenidamente el esquema anterior, los vectores **NR** y **NC** ya no contienen ceros, debido a que el *último elemento* de una fila (columna) tiene como *próximo elemento* al primero de la fila (columna).

Observaciones:

- Al recorrer los elementos de una fila (columna), no se puede obtener directamente la información de la columna (fila).
- A menos que se recorra toda la matriz, es imposible obtener la fila y columna de un elemento almacenado en cierta posición.
- El esquema circular KRM requiere menor espacio de memoria con respecto al esquema de Knuth .

3.2.5. Esquema Circular KRM Modificado

Este esquema es igual al anterior, excepto por la forma en que define a los vectores **NR** y **NC**, lo cual hace que el esquema circular modificado tenga mayor eficiencia.

$\mathbf{A} =$

	1	**2**	**3**	**4**
1		x		
2	x	x		x
3	x			
4		x		x

A continuación se puede observar la forma de trabajar con el esquema circular KRM modificado:

Posición	1	2	3	4	5	6	7
AN	a_{12}	a_{21}	a_{22}	a_{24}	a_{31}	a_{42}	a_{44}
NR	-1	3	4	-2	-3	7	-4
NC	3	5	6	7	-1	-2	-4
JR	1	2	5	6			
JC	2	1	0	4			

Aunque se debe almacenar la misma cantidad de información, el proceso es más eficiente, ya que con el signo "−" y el número de la fila (columna) se está indicando que en ésta no hay más elementos diferentes a cero.

Observaciones:

- Similar a lo que ocurre con el método circular KRM al recorrer los elementos de una fila (columna), no se puede obtener directamente la información de la columna (fila).
- El esquema circular KRM modificado permite la obtención de la fila y columna de un elemento a partir de su posición sin recorrer toda la matriz.

3.2.6. Esquema de Zollenkopf

El esquema de Zollenkopf es ampliamente aplicado en el análisis matricial de redes eléctricas y constituye un método muy completo de almacenamiento, lo cual se puede observar fácilmente en la cantidad de información que requiere. Para comprender el funcionamiento de este esquema, considere la siguiente red de 6 nodos y 7 ramas.

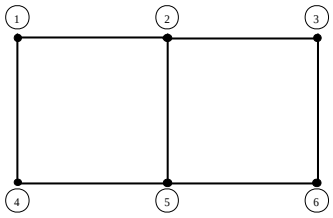


Figura 3.1: Red de 6 nodos y 7 ramas

Ahora, la estructura de la matriz de admitancia nodal $\mathbf{Y_{bus}}$ para el sistema anterior es:

$\mathbf{Y_{bus}} =$

	1	**2**	**3**	**4**	**5**	**6**
1	x	x		x		
2	x	x	x		x	
3		x	x			x
4	x			x	x	
5		x		x	x	x
6			x		x	x

Considere que la matriz sea:

- *Estructuralmente simétrica* $\longrightarrow$ si $y_{ij} \neq 0$ entonces $y_{ji} \neq 0$
- *Numéricamente asimétrica* $\longrightarrow$ en general $y_{ij} \neq y_{ji}$

A continuación, se puede observar la información que sería almacenada aplicando el esquema de Zollenkopf, se debe tener en cuenta que la numeración de los elementos no nulos se llevará a cabo de la misma forma que para los métodos anteriores; sin embargo, los elementos de la diagonal se numeran de una forma especial.

Posición	**LCOL**	**NOZE**	**ITAG**	**LNXT**	**DE**	**CE**	**RE**
1	1	3	2	2	y_{11}	y_{21}	y_{12}
2	3	4	4	0	y_{22}	y_{41}	y_{14}
3	6	3	1	4	y_{33}	y_{12}	y_{21}
4	8	3	3	5	y_{44}	y_{32}	y_{23}
5	10	4	5	0	y_{55}	y_{52}	y_{25}
6	13	3	2	7	y_{66}	y_{23}	y_{32}
7			6	0		y_{63}	y_{36}
8			1	9		y_{14}	y_{41}
9			5	0		y_{54}	y_{45}
10			2	11		y_{25}	y_{52}
11			4	12		y_{45}	y_{54}
12			6	0		y_{65}	y_{56}
13			3	14		y_{36}	y_{63}
14			5	0		y_{56}	y_{65}

A continuación se describen las funciones de cada uno de los vectores empleados por el esquema de Zollenkopf:

Posición $\longrightarrow$ *numeración elementos no nulos fuera de la diagonal*
LCOL $\longrightarrow$ *indicador de inicio de fila (columna)*
NOZE $\longrightarrow$ *número de elementos no nulos por fila (columna)*
ITAG $\longrightarrow$ *índice de fila (columna) de los elementos almacenados en RE (CE)*
LNXT $\longrightarrow$ *posición del próximo elemento no nulo de la fila (columna)*
DE $\longrightarrow$ *elementos de la diagonal de la matriz*
CE $\longrightarrow$ *elementos no nulos fuera de la diagonal almacenados por columna*
RE $\longrightarrow$ *elementos no nulos fuera de la diagonal almacenados por fila*

Observaciones:

- Es un esquema bastante utilizado en sistemas de potencia.
- No es el mejor esquema en términos de economía de almazenamiento, no obstante, presenta buena flexibilidad al momento de implementarlo.
- El esquema de Zollenkopf también puede ser utilizado para matrices estructural y numéricamente simétricas. En este caso, los vectores **RE** y **DE** no son necesarios.

Én el capítulo 6, se mostrará que si los elementos no nulos se numeran de forma organizada, la información a ser almacenada se puede reducir.

Capítulo Cuarto

Ordenamiento Matricial

Como se analizó en el capitulo 2, el error relativo en la respuesta de un sistema es función directa del número de operaciones que se realicen en el proceso de inversión. Igualmente, se mencionó que cuando la matriz de coeficientes posee una gran cantidad de entradas nulas, es decir, cuando el sistema presenta un alto grado de *dispersidad*, es necesario darle un tratamiento especial a dicha matriz para poder aprovechar al máximo su caracteristica dispersa. En el capítulo 3, se presentó una gran variedad de esquemas de almacenamiento compacto de matrices dispersas, y se mencionó que dentro de este empaquetamiento matricial era necesario almacenar unas posiciones de holgura si la matriz sería invertida, esto es debido a la generación de elementos ficticios (*fill-in*) en posiciones que originalmente eran nulas pero que despues de aplicar alguna técnica de inversión, bien sea eliminación de Gauss o descomposición LU, se genera un valor diferente de cero en dicha posición. Una de las conclusiones más importantes que provee el capítulo 2 es que el número de *fill-in* de un sistema puede ser reducido al mínimo si la matriz es ordenada de forma óptima.

El ordenamiento de una matriz, se logra mediante una permutación de filas, columnas o ambas. Una permutación puede analizarse como el producto de la matriz con una permutación de la matriz identidad. Si la matriz de permutación P pre-multiplica la matriz original, se intercambian las filas. Si se post-multiplica la matriz por la matriz de permutación, se obtiene una permutación de columnas. Si se realizan ambas multiplicaciones se logra intercambiar tanto las filas como las columnas, obteniendo así una *permutación simétrica*, que es la más utilizada en matrices dispersas. Hay que tener presente los efectos de dichas permutaciones sobre el sistema de ecuaciones, ya que el intercambio de columnas está cambiando realmente el orden de las incognitas del sistema.

Es precisamente el ordenamiento nodal el tema central de este capítulo, en cual se estudiarán las propuestas más interesantes presentes en la literatura especializada.

Finalizando el capítulo, se podrá concluir que el tema del ordenamiento nodal puede ser abordado desde un punto de vista combinatorial, haciendose una breve introducción al ordenamiento óptimo del sistema aplicando alguna técnica metaheurística, análisis que se desarrollará en profundidad en el siguiente capítulo, en el cuál se presenta una

novedosa metodología apoyada en el ***simulated annealing***.

4.1. Propósito del Ordenamiento Matricial

Para motivar la importancia del ordenamiento matricial en el proceso de dar solución a un sistema de ecuaciones algebraicas, es preciso considerar la siguiente matriz de coeficientes constantes que conforma el modelo de algún sistema físico:

$$\mathbf{A} = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \end{array} \\ \begin{bmatrix} \text{x} & \text{x} & \text{x} & \text{x} \\ \text{x} & \text{x} & & \\ \text{x} & & \text{x} & \\ \text{x} & & & \text{x} \end{bmatrix} \end{array} \tag{4.1}$$

Si se comenzara el proceso de inversión de esta matriz aplicando por ejemplo *eliminación de Gauss*, sería necesario llevar los elementos de la primera columna a un valor igual a cero, lo cual se logra con operaciones básicas o elementales de filas, obteniedo el siguiente sistema equivalente:

$$\mathbf{A}' = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \end{array} \\ \begin{bmatrix} \text{x} & \text{x} & \text{x} & \text{x} \\ 0 & \text{x} & \text{x'} & \text{x'} \\ 0 & \text{x'} & \text{x} & \text{x'} \\ 0 & \text{x'} & \text{x'} & \text{x} \end{bmatrix} \end{array} \tag{4.2}$$

Los elementos primados son valores diferentes de cero en posiciones que originalmente eran iguales a cero. A este tipo de elementos son a los que se denominan *fill-in*, un término anglosajon comunmente adaptado. Como puede observarse, se ha perdido la dispersidad de la matriz por completo, con esto, se genera a partir de este momento la realización de un número máximo de operaciones para poder dar solución al sistema, y como bien se mencionó en el capítulo 2, esto introduce un incremento en el error de la solución al igual que mayor tiempo de cómputo y espacio en memoria, ya que es necesario almacenar todos los elementos ficticios o *fill-in* que aparecieron.

Ahora, considerese que se realiza una permutación simétrica entre las variables 1 y 4, esto significa que la fila y columna correspondientes a la variable 1 ocuparán las posiciones correspondientes a la fila y columna de la variable 4 y viceversa. Realmente lo que se realizará es eliminar del sistema la variable 4 de primera y la variable 1 se eliminará de última. Así, se obtiene el siguiente sistema ordenado:

$$\mathbf{A}_{ord} = \begin{array}{c} \\ \mathbf{4} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{1} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{4} & \mathbf{2} & \mathbf{3} & \mathbf{1} \end{array} \\ \begin{bmatrix} \text{x} & & & \text{x} \\ & \text{x} & & \text{x} \\ & & \text{x} & \text{x} \\ \text{x} & \text{x} & \text{x} & \text{x} \end{bmatrix} \end{array} \tag{4.3}$$

Si ahora se inicia el proceso de eliminación, tal cual se hizo en el sistema anterior, se obtendría el siguiente sietema equivalente ordenado:

$$\mathbf{A}'_{ord} = \begin{array}{c} \\ \mathbf{4} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{1} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{4} & \mathbf{2} & \mathbf{3} & \mathbf{1} \end{array} \\ \begin{bmatrix} \text{x} & & & \text{x} \\ & \text{x} & & \text{x} \\ & & \text{x} & \text{x} \\ 0 & \text{x} & \text{x} & \text{x} \end{bmatrix} \end{array} \tag{4.4}$$

Claramente puede observarse que con este orden no se generó ningun elemento *fill-in*, lo que es igual a conservar la dispersidad de la matriz. Adicionalmente, el número de operaciones necesarias para llevar a cabo este propósito fue mínimo, con lo cual sólo se introduce un pequeño error al sistema en general, y es precisamente este el objetivo del ordenamiento matricial, tema que se desarrollará a continuación.

4.2. Esquemas de Ordenamiento Matricial

Básicamente todos los esquemas de ordenamiento matricial responden a los principios introducidos por **Tinney** y se explicarán a continuación:

4.2.1. Esquema de Ordenamiento de Tinney

W.F Tinney propuso tres esquemas de ordenamiento para minimizar fill-in:

4.2.1.1. Esquema I

El primer esquema desarrollado por **Tinney** es denominado *esquema estático* y básicamente suguiere que las columnas con menos elementos diferentes de cero deben ser procesadas primero, su eficiencia es baja aunque mejora los índices de *fill-in* en el proceso de inversión.

4.2.1.2. Esquema II

Mejorando los índices de eficiencia del primer esquema, este por su lado es denominado *esquema dinámico*, su principio de operación consiste en identificar en cada etapa

del proceso de factorización la columna que posea menor número de elementos no nulos. En su forma más simple se hace lo siguiente:

- Localizar la columna de la matriz con el menor número de elementos y eliminarla de primera.
- Al obtener el sistema equivalente despues de aplicar el item anterior, se debe observar la matriz generada de $(n-1) \times (n-1)$ donde no es tenida en cuenta ni la columna ni la fila de la variable procesada y se selecciona la columna con menor número de elementos y se prosigue su procesamiento.
- El proceso contínua de igual forma hasta que se llega a la matriz escalonada y el orden resultante de las variables procesadas es precisamente el orden sugerido con menor número de *fill-in*.

Realmente el orden que se obtiene reduce de manera considerable el número de elementos de relleno, más este no es precisamente el orden óptimo del sistema en el cual se presenta el número mínimo de *fill-in*.

4.2.1.3. Esquema III

Denominado *Esquema de Ordenamiento Óptimo*. En él se simula en cada etapa de factorización la salida de cada una de las variables del sistema, y se selecciona para salir aquella variable que genere en la presente etapa de factorización el menor número de elementos de relleno.

A pesar que este sistema es considerado como óptimo, su implementación es la más costosa en materia computacional, lo que la hace poco atractiva frente a los otros dos esquemas, más brinda una idea clara para generar propuestas con base en su principio operacional.

A continuación se realizará un ejemplo donde son aplicados cada uno de los esquemas de ordenamiento de Tinney, con el fín de obtener una idea clara de los niveles de eficiencia que surgen al ordenar una matriz con su respectivo esquema.

4.2.1.4. Ejemplo 4.1

Considere el circuito eléctrico de fase 90 ilustrado en la figura 7.1, tomado de [**?**]. Este tiene 17 ecuaciones nodales las cuales presentan 289 entradas. Se demostrará el impacto de la aplicación de las diferentes técnicas de ordenamiento de **Tinney** en el proceso de factorización **LU** para este ejemplo.

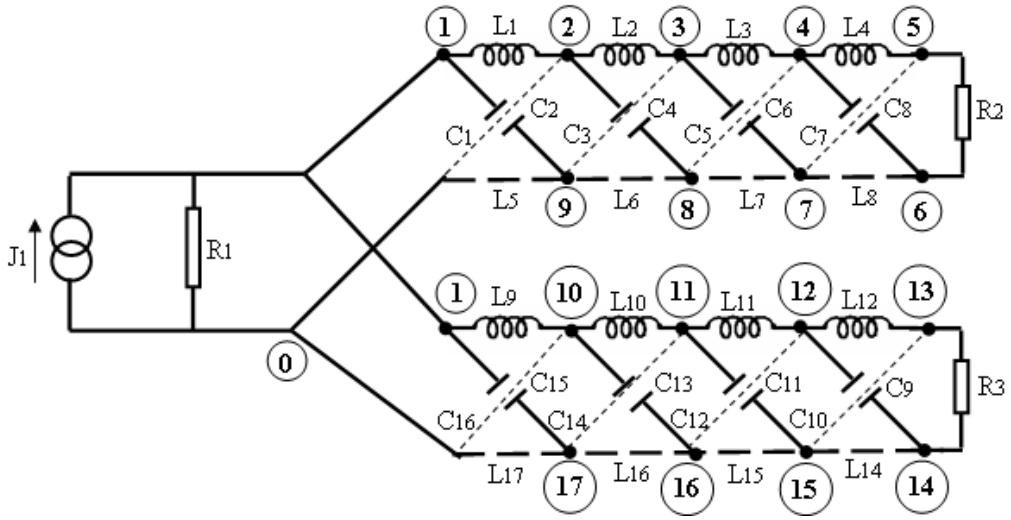


Figura 4.1: Todas las resistencias en ohms, inductancias en mH y capacitancias en uF

1. Los elementos iniciales diferentes de cero son indicados por cruces en la siguiente matriz, y los elementos *fill-in* que surgen si la matriz es factorizada con el orden indicado son representados por cruces encerradas en círculos. En este caso se presentan 68 fill-ins y son requeridas 377 operaciones (multiplicaciones y adiciones) comparadas con las apriximadamente 1550 operaciones si la matriz fuera densa.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
1	x	x							x	x							x
2	x	x	x					x	⊗	⊗							⊗
3		x	x	x			x	⊗	x	⊗							⊗
4			x	x	x	x	⊗	x	⊗	⊗							⊗
5				x	x	x	x	⊗	⊗	⊗							⊗
6				x	x	x	x	⊗	⊗	⊗							⊗
7			x	⊗	x	x	x	x	⊗	⊗							⊗
8		x	⊗	x	⊗	⊗	x	x	x	⊗							⊗
9	x	⊗	x	⊗	⊗	⊗	⊗	x	x	⊗							⊗
10	x	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	x	x					x	⊗
11										x	x	x			x	⊗	x
12											x	x	x		⊗	x	⊗
13												x	x	x	x	⊗	⊗
14												x	x	x	x	⊗	⊗
15											x	⊗	x	x	x	x	⊗
16										x	⊗	x	⊗	⊗	x	x	x
17	x	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	x	⊗	⊗	⊗	⊗	x	x

2. Cuando las filas y columnas son ordenadas según el **Esquema I**, se genera la matriz

representada a continuación. Para este ordenamiento, son generados 32 fill-ins y el número de operaciones requeridas para la descomposición **LU** es 209.

	2	**5**	**6**	**9**	**10**	**13**	**14**	**17**	**1**	**3**	**4**	**7**	**8**	**11**	**12**	**15**	**16**
2	x								x	x			x				
5		x	x								x	x					
6		x	x								x	x					
9				x					x	x			x				
10					x				x					x			x
13						x	x								x	x	
14						x	x								x	x	
17								x	x					x			x
1	x			x	x			x	x	⊗			⊗	⊗			⊗
3	x			x					⊗	x	x	x	⊗	⊗			⊗
4		x	x							x	x	⊗	x	⊗			⊗
17		x	x							x	⊗	x	x	⊗			⊗
8	x			x					⊗	⊗	x	x	x	⊗			⊗
11					x			x	⊗	⊗	⊗	⊗	⊗	x	x	x	⊗
12						x	x							x	x	⊗	x
15						x	x							x	⊗	x	x
16					x			x	⊗	⊗	⊗	⊗	⊗	⊗	x	x	x

3. Aplicando la metodología propuesta según el **Esquema II**, se genera la matriz representada a continuación. Con este orden son generados únicamente 14 fill-ins y el número de operaciones es reducido a 147.

	2	**5**	**6**	**4**	**7**	**3**	**8**	**9**	**1**	**10**	**7**	**11**	**16**	**12**	**13**	**14**	**15**
2	x					x	x		x								
5		x	x	x	x												
6		x	x	x	x												
4		x	x	x	⊗	x	x										
7		x	x	⊗	x	x	x										
3	x			x	x	x	⊗	x	⊗								
8	x			x	x	⊗	x	x	⊗								
9						x	x	x	x								
1	x					⊗	⊗	x	x	x	x						
10									x	x	⊗	x	x				
17									x	⊗	x	x	x				
11										x	x	x	⊗	x			x
16										x	x	⊗	x	x			x
12												x	x	x	x	x	⊗
13														x	x	x	x
14														x	x	x	x
15												x	x	⊗	x	x	x

4. Finalmente, aplicando la metodología del **Esquema III**, la estrategía de mínimo Fill-in local, representada en la siguiente matriz, reduce a 12 el número de fill-ins y el número de operaciones necesarias a 141.

	5	**6**	**13**	**14**	**4**	**7**	**12**	**15**	**3**	**8**	**11**	**16**	**2**	**9**	**10**	**17**	**1**
5	x	x			x	x											
6	x	x			x	x											
13			x	x			x	x									
14			x	x			x	x									
4	x	x			x	⊗			x	x							
7	x	x			⊗	x			x	x							
12			x	x			x	⊗			x	x					
15			x	x			⊗	x			x	x					
3					x	x			x	⊗			x	x			
8					x	x			⊗	x			x	x			
11							x	x			x	⊗			x	x	
16							x	x			⊗	x			x	x	
2									x	x			x	⊗			x
9									x	x			⊗	x			x
10											x	x			x	⊗	x
17											x	x			⊗	x	x
1													x	x	x	x	x

4.2.2. Ordenamiento Casi-Óptimo de Stevenson

Stevenson en [19], adopta el **Esquema de Ordenamiento II de Tinney**, y le adiciona un análisis gráfico para aquellos sistemas que representan redes eléctricas como es el caso de la matriz de admitancia nodal. Para ello, se generan básicamente dos reglas que conforman la metodología en general:

1. Remuévase el nodo k y todas sus ramas incidentes de la gráfica del sistema, En particular, remuévase la rama km si el nodo k se encuentra conectado *radialmente* a la red, esto es, si el nodo k está conectado solamente a otro nodo m.

2. Añádase una nueva rama entre cada par de nodos i y j si, y sólo si, están directamente conectados al nodo k pero no entre ellos antes de que el nodo k se elimine.

Este tipo de estrategías es aplicada solámente a matrices simétricas en estructuras, y se puede describir fácilmente realizando una representación del elemento ficticio (fill-in) generado en una red eléctrica física. Para ello es necesario considerar el siguiente sistema eléctrico de corriente contínua de seis nodos y dos mallas, por comodidad asumase que todas las resistencias de rama son iguales 1Ω. El sistema se ilustra en la figura 4.2.
La matriz de admitancia nodal que representa dicho sistema sería la siguiente:

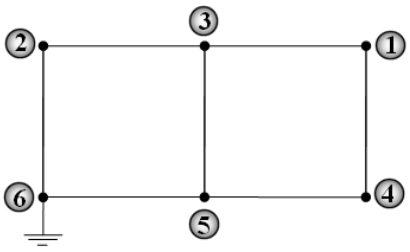


Figura 4.2: Sistema eléctrico de 6 nodos y 2 mallas

$$\mathbf{Y}_{barra} = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} \end{array} \\ \begin{bmatrix} 2 & 0 & -1 & -1 & 0 & 0 \\ 0 & 2 & -1 & 0 & 0 & -1 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & 0 & -1 & \infty \end{bmatrix} \end{array} \tag{4.5}$$

Al iniciar el proceso de eliminación Gaussiana, se obtiene el siguiente sistema equivalente:

$$\mathbf{Y}'_{barra} = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} \end{array} \\ \begin{bmatrix} 2 & 0 & -1 & -1 & 0 & 0 \\ 0 & 2 & -1 & 0 & 0 & -1 \\ 0 & -1 & 5/2 & -1/2 & -1 & 0 \\ 0 & 0 & -1/2 & 3/2 & -1 & 0 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & 0 & -1 & \infty \end{bmatrix} \end{array} \tag{4.6}$$

Como bien puede observarse, se han generado dos *fill-in* en las posiciones $\mathbf{Y}'(3,4)$ y $\mathbf{Y}'(4,3)$ esto es debido a la simetría de la matriz. Lo importante es la nueva topología del sistema, la cual se indica en la figura 4.3.

Claramente puede observarse que se ha generado una rama fictica en el sistema, su valor es correspondiente a la suma serie de las admitancias de rama que unen los nodos 3–1 y 1–4, en la figura 4.2, la cual es en efecto igual en valor a $y_{3-4} = (y_{3-1} \times y_{1-4})/(y_{3-1} + y_{1-4}) = 1/2$, el cual es el valor correspondiente del *fill-in* que surgue en la matriz $\mathbf{Y}'_{barra}$ al eliminar el nodo 1.

Gráficamente es a lo que se refiere Stevenson en su metodología y mediante sus reglas se encuentra que la gráfica de la red solamente tiene un nodo menos en cada etapa

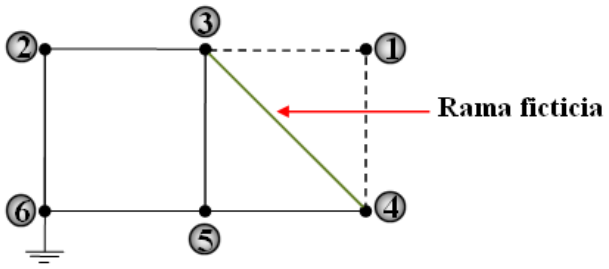


Figura 4.3: Sistema eléctrico resultante al eliminar el nodo 1 (ecuación 1)

del proceso de eliminación gaussiana y el número de ramas incidentes en el resto de los nodos se podría hacer variar. El propósito es el evitar añadir ramas en todo lo posible (recuerde que añadir ramas es equivalente a generar elementos *fill-in*).

Para fines del análisis y comparación de los resulatados que se pretenden desarrollar en los posteriores capítulos, se hará una breve explicación de las métodologías clásicas de solución de sistemas de ecuaciones líneales donde intervien de forma primordial el ordenamiento nodal del sistema.

4.3. Inversión de Matrices Dispersas

4.3.1. Inversión de Matrices Dispersas por Diagonalización

Esta metodología fue desarrollada por Y. Q. Wang y H. B. Gooi en [20]. Partiendo del hecho de que la matriz de coeficientes $\mathbf{A}$ del sistema es no singular y simétrica, entonces, es posible representarla como el siguiente producto:

$$\mathbf{A} = \mathbf{LDU} \tag{4.7}$$

Donde $\mathbf{L}$ y $\mathbf{U}$ son los factores triangulares inferiores y superiores de $\mathbf{A}$ con sus respectivas diagonales unitarias. $\mathbf{D}$ es una matriz diagonal.

Si esta representación es posible, entonces la inversa de $\mathbf{A}$ puede ser expresada como:

$$\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{D}^{-1}\mathbf{L}^{-1} = \mathbf{W}^{\mathbf{U}}\mathbf{D}^{-1}\mathbf{W}^{\mathbf{L}} \tag{4.8}$$

Claramente puede observarse que $\mathbf{W}^{\mathbf{U}}$ y $\mathbf{W}^{\mathbf{L}}$ se han definido como las inversas de $\mathbf{U}$ y de $\mathbf{L}$ respectivamente.

Cuando una matriz puede ser diagonalizada, entonces, es posible representarla como:

$$\mathbf{C}_R\mathbf{A}\mathbf{C}_C = \mathbf{D}_* \tag{4.9}$$

Las matrices $\mathbf{C}_R$ y $\mathbf{C}_C$ son las respectivas matrices de transformación de $\mathbf{A}$ en $\mathbf{D}_*$. La matriz $\mathbf{C}_R$ transforma a $\mathbf{A}$ en una matriz triangular superior y $\mathbf{C}_C$ transforma la matriz triangular superior en una matriz diagonal. Nuevamente, si la solución del sistema existe, entonces

$$\mathbf{A}^{-1} = \mathbf{C}_C\mathbf{D}_*^{-1}\mathbf{C}_R \tag{4.10}$$

Es preciso aclarar que diferente orden en el cual se ejecuten las operaciones para obtener la inversa de la matriz $\mathbf{A}$ generará diferentes matrices de $\mathbf{D}_*$, $\mathbf{C}_C$ y $\mathbf{C}_R$. Igualmente, es interesante observar la similitud entre las ecuaciones 4.8 y 4.10, lo cual sirve de base para el siguiente desarrollo.

Si se satisface la condición de simetría, la siguiente propiedad es válida:

$$\mathbf{C}_C = \mathbf{C}_R^T$$

con lo cual se genera la siguiente expresión.

$$\mathbf{A}^{-1} = \mathbf{C}\mathbf{D}_*^{-1}\mathbf{C}^T \tag{4.11}$$

La metodología propuesta consiste en obtener la matriz $\mathbf{C}$ con el menor número de elementos *fill-in*, para ello, se procede a colocar una matriz identidad justo debajo de la matriz original $\mathbf{A}$, es decir, se opera con un sistema modificado de dimención $2n \times n$, y se procede a diagonalizar la matriz original aplicando el esquema II propuesto por Tinney, para ilustrar un poco esta metodología considere el siguiente sistema de 20 nodos representado en la figura 4.4

El siguiente sistema representa el proceso de diagonalización en la etapa 19 de la factorización:

La filosofía que se adopta para minimizar el número de elementos de relleno en la matriz $\mathbf{C}$ es la misma desarrollada por el esquema II de Tinney como bien se habia mencionado previamente.

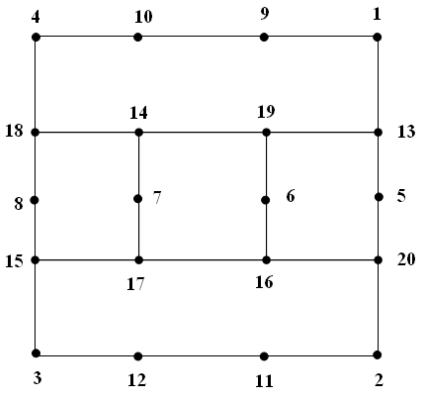


Figura 4.4: Sistema de 20-nodos

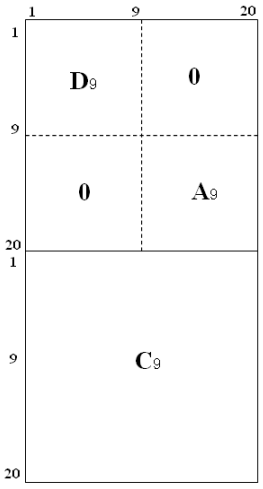


Figura 4.5: Esquema de la matriz aumentada en la etapa 19 del proceso de diagonalización

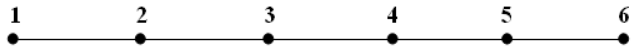
Gracias a la estrecha relación entre las ecuaciones 4.8 y 4.10, el número de *fill-in* que se adquieren con el orden resultante al aplicar la estrategía de minimización para obtenr la matriz C de la ecuación 4.10, será el mismo que se generariá si se obtuviera la matriz $\mathbf{W}$ de la ecuación 4.8 con la misma conbinación de variables, a pesar que estas matrices sean numéricamente diferentes.

La ventaja que tiene esta metodología es el cálculo rápido de la matriz $\mathbf{C}$ y $\mathbf{D}_*$ aplicando el esquema II de Tinney, con lo cual se han obtenido grandes índices en lo que a la reducción de elementos de relleno se refiere.

4.3.2. Inversa de Matrices Dispersas por Particiones

Esta metodología fue desarrollada por Alvarado, David y Betancourt en [3]. Se resuelve el clásico problema $\mathbf{A}x = b$ introduciendo el mismo concepto de las matrices $\mathbf{W}^L$

y $\mathbf{W}^U$ de la sección anterior, y se adiciona el concepto de particiones matriciales, lo cual consisten en dividir la matriz original en multiplos equivalentes de ella misma con lo cual se consigue operar con sistemas más fáciles de manipular. Para aclarar esta metodología de una manera más sencilla, considere el siguiente sistema puramente radial de 6 nodos que se ilustra en la figura 4.3.2.



La topología de la matriz dispersa asociada con este sistema es:

$$\mathbf{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} x & x & & & & \\ x & x & x & & & \\ & x & x & x & & \\ & & x & x & x & \\ & & & x & x & x \\ & & & & x & x \end{bmatrix} \end{array} \tag{4.12}$$

La descomposición ordinaría de esta matriz resulta en una matriz triangular inferir $\mathbf{L}$, en una matriz diagonal $\mathbf{D}$ y una matriz triangular superior $\mathbf{L}^T$. La matriz triangular inferior $\mathbf{L}$ puede ser expresada como el producto de las matrices triangulares inferiores elementales $\mathbf{L}_i$. La estructura de las matrices triangulares inferiores elementales $\mathbf{L}_i$ *y la de su inversa* $\mathbf{W}_i = \mathbf{L}_i^{-1}$ es la misma y esta dada por:

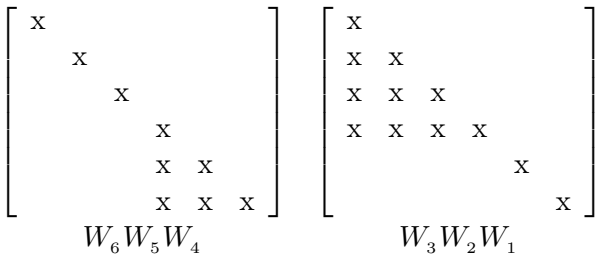
$$\begin{bmatrix} x & & & & & \\ x & x & & & & \\ & x & & & & \\ & & x & & & \\ & & & x & & \\ & & & & & x \end{bmatrix} \quad \begin{bmatrix} x & & & & & \\ & x & & & & \\ & x & x & & & \\ & & & x & & \\ & & & & x & \\ & & & & & x \end{bmatrix} \quad \begin{bmatrix} x & & & & & \\ & x & & & & \\ & & x & & & \\ & & x & x & & \\ & & & & x & \\ & & & & & x \end{bmatrix}$$

$$L_1; W_1 \qquad L_2; W_2 \qquad L_3; W_3$$

$$\begin{bmatrix} x & & & & & \\ & x & & & & \\ & & x & & & \\ & & & x & & \\ & & & x & x & \\ & & & & & x \end{bmatrix} \quad \begin{bmatrix} x & & & & & \\ & x & & & & \\ & & x & & & \\ & & & x & & \\ & & & & x & \\ & & & & x & x \end{bmatrix} \quad \begin{bmatrix} x & & & & & \\ & x & & & & \\ & & x & & & \\ & & & x & & \\ & & & & x & \\ & & & & & x \end{bmatrix}$$

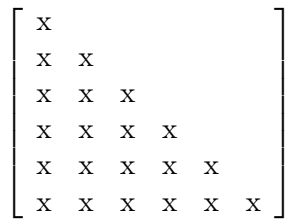
$$L_4; W_4 \qquad L_5; W_5 \qquad L_6; W_6$$

Los elementos fuera de la diagonal de $\mathbf{L}_i$ y de $\mathbf{W}_i$ tienen signos opuestos pero poseen la misma magnitud. $\mathbf{L}_6$ y $\mathbf{W}_6$ son matrices identidad y pueden ser omitidas. Considere ahora dos matrices formadas por el producto de las tres últimas matrices elementales y el producto de las tres primeras respectivamente.



$W_6 W_5 W_4$ $W_3 W_2 W_1$

Estos productos por separado retienen un número de elementos diferentes de cero. Si se agrupan todas las matrices elementales en un mismo grupo se obtiene la topología de L^{-1}.



Ahora bien, el proceso de optimización de fill-in consiste en obtener un orden en el producto de las matrices elementales $\mathbf{L}_i$, este orden debe generar un mínimo número de matrices elementales para así garantizar un mínimo número de elementos de relleno. Este proceso es llevado a cabo por diferentes metodologías dentro de las cuales sobresale de manera notable el análisis hecho a un árbol o grafo, donde cada uno de los nodos de este grafo representa una operación elemental en la matriz original.

Capítulo Quinto

Simulated Annealing

Gran número de problemas en ingeniería pueden ser modelados como un proceso de minimización o maximización de una función de costos sujeta a un conjunto de restricciones, donde las variables de decisión pueden o no ser discretas. Esta clase de problemas suelen ser llamados de optimización combinatorial y han recibido gran atención desde hace unas décadas. Dentro de los logros alcanzados por estas metodologías, se encuentra el hecho de haber permitido la separación de esta clase en dos sub-clases. La primera de ellas contiene los problemas que pueden ser resueltos eficientemente, por ejemplo, aquellos para los cuales se conocen algoritmos que alcanzan el óptimo en un tiempo polinomial, tales como la programación lineal y flujo en redes. La segunda sub-clase reune los problemas que son notablemente difíciles *"hard"*, formalmente conocidos como *"NP-hard"*. Para un problema NP-hard es reconocido que no existen algoritmos con esfuerzo computacional de tipo polinomial para encontrar su solución óptima [18]. Muchos problemas conocidos pertenecen a esta categoría y probablemente el ejemplo más conocido es el problema *traveling salesman*. La mencionada distinción está respaldada por un marco general en ciencias de la computación llamado teoría de la complejidad [6], [7].

Es evidente que problemas NP-hard deben ser manejados en la práctica. En términos generales, esto puede hacerse mediante dos tipos de algoritmos de naturaleza intrínsecamente diferentes: o bien se pueden utilizar *algoritmos de optimización exacta* que alcancen soluciones óptimas, posiblemente utilizando grandes cantidades de tiempo de cómputo o se pueden usar *algoritmos heurísticos* que encuentren soluciones de buena calidad en un tiempo de cálculo mucho menor. Los algoritmos de búsqueda local son los de este último tipo [8]. Simulated Annealing, el tema de este capítulo, se encuentra entre los más conocidos algoritmos de búsqueda local, ya que es bastante eficiente y es ampliamente aplicable. En este capítulo se presentan los conceptos básicos de Simulated Annealing. En primer lugar, se introducen algunos conceptos elementales de búsqueda local. Así mismo se presenta un enfoque correspondiente a la fuerte analogía con el proceso físico del recocido de sólidos. A continuación, se presentan algunos programas de enfriamiento que permiten un tiempo finito de ejecución. Finalmente, se discuten algunas cuestiones relacionadas con el uso práctico del Simulated Annealing.

5.1. Búsqueda Local

Los algoritmos de búsqueda local constituyen un campo ampliamente aplicado, corresponden a un enfoque general para poblemas de optimización combinatorial tipo hard. Estos problemas son casos específicos de varios esquemas generales de búsqueda local, pero todos comparten la misma característica de basarse en una función de vecindad, que es usada para guiar la búsqueda de una buena solución.

Un ejemplo de un problema de optimización combinatorial consiste en un conjunto S de soluciones factibles y un valor positivo de la función de costo f. El problema es encontrar la solución óptima global $i^* \in S$, es decir, una solución con costo óptimo f^*. Una función de vecindad es un mapeo $N : S \to 2^S$ que define para cada solución $i \in S$ un conjunto $N(i) \subseteq S$ de soluciones que son, en cierto sentido, cercanas a i. El conjunto $N(i)$ representa la *vecindad* de la solución i y cada $j \in N(i)$ corresponde a un *vecino* de i. La forma más simple de búsqueda local es llamado *mejoramiento iterativo*. Un algoritmo de mejoramiento iterativo comienza con una solución inicial y, a continuación, explora las configuraciones vecinas para encontrar una con menor costo. Una vez es encontrada, la solución actual se reemplaza por esta última. El procedimiento se repite hasta que no se pueda encontrar una mejor solución en la vecindad de la configuración actual. Por definición, el mejoramiento iterativo finaliza en un *óptimo local*, es decir, una solución $i \in S$ que como mínimo es tan buena (en su función de costo) como sus vecinos. Nótese que el concepto de optimalidad local depende de la función de vecindad implementada.

Para muchos problemas de optimización combinatorial se pueden representar soluciones como secuencias de subconjuntos de elementos que constituyen las soluciones; por ejemplo las rutas en el problema del agente viajero (TSP —por sus siglas en inglés—). Este tipo de representaciones permite el uso de k cambios entre soluciones vecinas, es decir, vecinos que se obtienen mediante la definición de k intercambios de elementos en una secuencia dada de subconjuntos. Los llamados k cambios dentro de la vecindad constituyen una clase de algoritmos básicos de búsqueda local que son ampliamente aplicados ([13], [14]).

5.2. Simulated Annealing Básico

A principios de la década de los 80's Kirkpatrick et al. [9] (1983) e independientemente Černý [10] (1985) introdujeron los conceptos del recocido en optimización combinatoria. Originalmente, estos conceptos fueron fuertemente inspirados por una analogía entre el proceso físico del recocido de sólidos y el problema de resolver grandes problemas de optimización combinatorial. Ya que esta analogía es bastante atractiva, se usa aquí como base para la introducción del tema.

En física de la materia condensada, el recocido se conoce como un proceso térmico para obtener estados de baja energía en un sólido sometido a un gradiente de temperatura. El poceso consiste en los siguientes dos pasos [9]:

- Aumentar la temperatura hasta el máximo valor antes de que el sólido empiece a fundirse.
- Disminuir la temperatura cuidadosamente hasta que las partículas del sólido alcancen un estado de estructura cristalina perfecta.

En el estado líquido, todas las partículas se organizan aleatoriamente, mientras que en el estado sólido, las partículas están organizados en una red fuertemente estructurada, a la que la energía libre es mínima. La estructura perfecta del sólido se obtiene sólo si el valor máximo de la temperatura es suficientemente alta y el enfriamiento se realiza suficientemente despacio. De lo contrario, el sólido se va a congelar en un estado metaestable y no en el verdadero.

Metropolis et al. [11] presentó un algoritmo sencillo para simular la evolución de un sólido somtido a un gradiente de temperatura para el equilibrio térmico. Su algoritmo se basa en las técnicas de Monte Carlo [12] y genera una secuencia de estados del sólido en la siguiente forma. Dado el estado actual i del sólido con energía E_i, seguido de un estado posterior j que es generado a partir de la aplicación de un mecanismo de perturbación que transforma el estado actual en un estado próximo por medio de una pequeña perturbación, por ejemplo mediante el desplazamiento de una partícula. La energía del siguiente estado es E_j. Si la diferencia de energía, $E_j - E_i$, es menor o igual a cero, el estado j se acepta como el estado actual. Si la diferencia de energía es mayor que cero, entonces el estado j se acepta con una probabilidad dada por

$$e^{\frac{E_i - E_j}{k_B T}} \tag{5.1}$$

donde T denota la temperatura a la que se somete el sólido y k_B es una constante física llamada constante de Boltzmann. La regla de aceptación descrita anteriormente es conocida como el *criterio de Metropolis* y el algortimo *algoritmo de Metropolis*. Se sabe que, si la reducción de la temperatura se hace lo suficientemente despacio, el sólido puede llegar al equilibrio térmico en cada nivel de temperatura. En el algoritmo de Metropolis esto se logra mediante la generación de un gran número de transiciones en un determinado valor de temperatura. El equilibrio térmico se caracteriza por la distribución de Boltzmann, la cual calcula la probabilidad de que un sólido esté en un estado i con energía E_i, a una temperatura T, y que está dada por

$$P_T\{X = i\} = \frac{e^{\frac{-E_i}{k_B T}}}{\sum\limits_j e^{\frac{-E_j}{k_B T}}} \tag{5.2}$$

donde X es una variable aleatoria que denota el estado actual del sólido y la sumatoria recorre todos los posibles estados. Como se indicó anteriormente, la distribución de Boltzmann desempeña un papel esencial en el análisis de la convergencia del Simulated Annealing.

Volviendo al Simulated Annealing, el algoritmo de Metropolis puede ser utilizado para generar una secuencia de soluciones de un problema de optimización combinatorial

asumiendo las siguientes equivalencias entre un sistema físico compuesto de un gran número de partículas y un problema de optimización combinatorial:

- las soluciones en el problema de optimización combinatorial son equivalentes a los estados del sistema físico
- el costo de una solución es equivalente a la energía de un estado

Por otra parte, se introduce un parámetro de control que desempeña el papel de la temperatura. El Simulated Annealing, por lo tanto, puede considerarse como una iteración del algoritmo de Metropolis, ejecutado como la disminución de los valores del parámetro de control.

Ahora, se puede pasar de la analogía física y formular el Simulated Annealing en términos de un algotirmo de búsqueda local. Para simplificar la presentación, se asume, en lo que resta de este capítulo, que se está tratando con un problema de minimización. La discusión se puede trasladar fácilmente a problemas de maximización. Para una instancia (S, f) de un problema de optimización combinatorial y una función vecina N, el siguiente pseudo-código describe el proceso de Simulated Annealing.

Algorithm 1 ALGORITMO SIMULATED_ANNEALING

```
begin
INICIALIZAR (i_start, c_0, L_0);
k = 0;
i = i_start;
repeat
for l = 1 to L_k do
  begin
  GENERAR (j from S_i);
  if f(j) ≤ f(i) then
    i = j
  else
    if e^((f(i)−f(j))/c_k) > random[0,1] then
      i = j
    end if
  end if
end for
k = k + 1;
CALCULE_LONGITUD (L_k);
CALCULE_CONTROL (c_k);
until stopcriterion
end
```

El significado de las cuatro funciones en el algoritmo SIMULATED_ANNEALING es bastante sencillo: INICIALIZAR calcula una solución inicial y un valor inicial para los parámet-

ros c y L; GENERAR selecciona una solución vecina a la configuración actual; CALCULE_LONGITUD y CALCULE_CONTROL calculan nuevos valores de los parámetros L y c, respectivamente.

Como ya se ha mencionado, una característica del Simulated Annealing es que, además de aceptar mejoras en la función objetivo, también acepta empeoramientos de forma limitada. Inicialmente, para valores grandes de c, serán aceptados grandes empeoramientos; a medida que c disminuye, sólo serán aceptados deterioros pequeños y, finalmente, cuando c tiende a cero, sólo se aceptan mejoramientos de la función objetivo. Además, no hay limitante en el tamaño de un deterioro con respecto a su aceptación. En Simulated Annealing, son aceptados arbitrariamente grandes empeoramientos con probabilidad positiva, sin embargo, la probabilidad de aceptación de de estos deterioros es pequeña. Esta característica significa que el Simulated Annealing, en contraste con el mejoramiento iterativo, puede escapar de óptimos locales, sin renunciar a las características favorables de este último, es decir, la sencillez y la aplicabilidad general.

Nótese que la probabilidad de aceptar empeoramientos en la función objetivo se aplica comparando el valor de $e^{(f(i)-f(j))/c}$ con un número aleatorio generado a partir de una distribución uniforme en el intervalo $[0, 1]$. Además, debería ser obvio que la velocidad de convergencia del algoritmo está determinada por la selección de los parámetros L_k y c_k con $k = 1, 2, \ldots$; donde L_k y c_k denotan los valores de los parámetros L y c en la k-ésima iteración del algoritmo.

Comparando al Simulated Annealing con el Mejoramiento Iterativo, es evidente que el primero puede ser visto como una generalización. En el caso de que el parámetro de control c tome el valor de cero, el Simulated Annealing presentaría un comportamiento idéntico al del Mejoramiento Iterativo. Con respecto al desempeño que presentan los dos algoritmos, cabe mencionar que para la mayoría de problemas, el Simulated Annealing ofrece un mejor rendimiento [5].

5.3. Implementación del Método

La implementación del Simulated Annealing es obtenida mediante la generación de una secuencia homogénea de cadenas de Markov, de longitud finita y con valores descendentes del parámetro de control. Para esto, se debe especificar un conjunto de parámetros que gobiernan la convergencia del algoritmo. Estos parámetros son combinados en lo que es llamado un *programa de enfriamiento*. Un *programa de enfriamiento* especifica una secuencia finita de valores de los parámetros de control y un número finito de transiciones para cada valor del parámetro de control. Más precisamente, se especifica por

- un *valor inicial* del parámetro de control c_0,
- una *función decremental* para reducir el valor del parámetro de control,
- un *valor final* del parámetro de control especificado como *criterio de parada*, y
- una *longitud* finita para cada cadena de Markov.

La búsqueda de un programa de enfriamiento adecuado ha sido el objetivo de numerosos estudios en los últimos años. Dentro de los estudios más importantes se encuentran los hechos por Van Laarhoven y Aarts [15] en 1987, Collins et al. [16] en 1988, y Romeo y Sangiovanni-Vincentelli [17] en 1991.

La mayoria de los trabajos sobre programas de enfriamiento presentados en la literatura proponen algoritmos heurísticos. Se distinguen dos grandes clases: estáticos y dinámicos. En un programa de enfriamiento estático los parámetros son fijos, no pueden ser cambiados durante la ejecución del algoritmo. En un programa de enfriamiento dinámico son variados adaptativamente a medida que avanza la ejecución del algoritmo.

5.3.1. Programas de Enfriamiento Estáticos

El siguiente programa es conocido como *"the geometric schedule"*. Se origina a partir de los primeros trabajos en programas de enfriamiento de Kirkpatrick et al. [9] (1983) y aún se utiliza en muchas situaciones prácticas.

Valor inicial del parámetro de control. Para asegurar un valor lo suficientemente grande de c_0, se puede hacer $c_0 = \Delta f_{max}$, donde Δf_{max} es la máxima diferencia en el valor de la función objetivo entre dos soluciones vecinas cualquiera. En la mayoría de los casos, el cálculo exacto de Δf_{max} requiere de bastante tiempo. Sin embargo, a menudo se pueden dar simples estimaciones de su valor.

Disminución del parámetro de control. Una función de disminución usada frecuentemente está dada por

$$c_{k+1} = \alpha \cdot c_k \qquad k = 0, 1, 2, \ldots \tag{5.3}$$

donde α es una constante positiva, pequeña y cercana a 1. Los valores típicos se encuentran entre 0.8 y 0.99.

Valor final del parámetro de control. El valor final es fijado en algún número pequeño, que puede estar relacionado con la menor diferencia posible emtre los valores de las funciones objetivo de dos soluciones vecinas.

Longitud de la cadena de Markov. La longitud de las cadenas de Markov se fija en un número que puede estar relacionado con el tamaño de la vecindad en el problema a manejar.

5.3.2. Programas de Enfriamiento Dinámicos

Existen muchas extensiones del simple programa de enfriamiento estático presentado anteriormente que dan lugar a un programa dinámico. Por ejemplo, un valor lo suficientemente grande de c_0 puede ser obtenido mediante el requisito de que el rango de aceptación $\omega(c_0)$ sea cercano a 1. Esto se puede lograr iniciando desde un valor pequeño y mayor que cero de c_0, que será multiplicado por un factor constante mayor que 1, hasta que el correspondiente valor de $\omega(c_0)$, que es calculado del número de transiciones generadas, sea cercano a 1. Valores típicos para $\omega(c_0)$ están entre 0.9 y 0.99. Se puede

obtener un cálculo adaptivo del valor final del parámetro de control mediante el requisito de terminar la ejecución del algoritmo en un valor c_k, para el cual el valor de la función objetivo de la solución obtenida en la última evaluación de una cadena de Markov se mantenga igual para un número consecutivo de cadenas. Es evidente que existe un valor para cada mínimo local que se encuentre. La longitud de las cadenas de Markov se puede determinar haciendo que para cada valor c_k, se acepte un número mínimo de transiciones. Sin embargo, dado que las transiciones son aceptadas con probabilidad decreciente, se puede llegar a que $L_k \to \infty$ para $c_k \downarrow 0$.

En cuanto a programas de enfriamiento dinámicos, en la literatura se presenta un buen número de algoritmos más elaborados. La mayoría de éstos están basados en un análisis estadístico del Simulated Annealing.

Capítulo Six

Metodología Propuesta

En la solución de sistemas de ecuaciones algebraicas lineales se genera un error que es función, como bien se ha analizado en los capítulos anteriores, del número de operaciones que se realizan en el proceso. Igualmente, se ha demostrado que este error puede ser minimizado si el sistema es ordenado previamente y almacenado de forma compacta.

El objetivo de este capítulo es plantear la aplicación de técnicas meta-heurísticas en la determinación del orden óptimo de un sistema. Aunque las técnicas meta-heurísticas de solución a problemas de optimización tienen un sustento matemático que garantiza un acercamiento al óptimo en un número infinito de iteraciones, en la práctica su implementación llega a óptimos locales. Además de ello, se propone una metodología que incluye el almacenamiento compacto basado en el esquema de Zollenkopf, a través del cual se da solución al sistema de ecuaciones en tres etapas secuenciales: la primera de ellas consite en desarrollar la descomposición triangular de la matriz almacenada; posteriormente, se soluciona el sistema $\mathbf{LD}z = b$, que lógicamente también está almacenado de forma compacta y así, finalmente, obtener la solución del sistema en general a través del conjunto de ecuaciones $\mathbf{U}x = b$.

Para validar la aplicabilidad de esta metodología se presentará en el desarrollo de este capítulo una variedad de ejemplos de interés que proporcionan la capacidad operativa del método que se presenta, esto es con el objetivo de evidenciar la eficiencia tanto que presenta el ordenamiento matricial así como el almacenamiento compacto del sisitma, de igual forma, se desarrolla un análisis general donde es almacenada de forma compacta la matriz de admitancia nodal ($\mathbf{Y_{BUS}}$) de determinado sistema eléctrico, para fines de este análisis, se pide determinar a partir de este almacenamiento compacto las impedancias de transferencia y de punto de operación en determinado nodo, estudio importante que valida la metodología para el estudio de corto circuito.

6.1. Modelo Matemático del Problema

El modelo matemático del problema que se aborda, como bien se ha mencionado en capítulos anteriores, consiste en la minimización del número de fill-in del sistema, el cual es una función directa del orden en que se proceasan las variables del mismo. Así, es posible representar el problema de la siguiente manera:

$$\begin{cases} \min \quad f(\mathbf{x}) \\ \text{s.a.} \\ \qquad \mathbf{x} \in \Re^n \\ \qquad \mathbf{x}_i \in Z \quad \forall i = 1, 2, ..., n \end{cases} \tag{6.1}$$

Donde:

- $f(\mathbf{x})$ Representa el número de ***fill-in*** generados a partir de cierta combinación de variables.
- $\mathbf{x}$ Es un vector de orden n, el cual contiene a todas las variables del sistema.
- $\mathbf{x}_i$ Representa la i-ésima componente (i-ésima variable) del vector $\mathbf{x}$, las cuales lógicamente deben ser variables enteras.

6.2. Algoritmo de Ordenamiento Matricial

6.2.1. Simulated Annealing aplicado al ordenamiento óptimo

Como se ha visto, el ordenamiento óptimo de un sistema es una combinación estratégica en el procesamiento de variables, es decir, de la secuencia en que se eliminan incógnitas del sistema. Con ello se logra minimizar el número de elementos de relleno que aparecen en el proceso de inversión de la matriz de coeficientes. Básicamente, el problema consiste en obtener dicho orden, lo cual requiere llevar a cabo un búsqueda exhaustiva en el espacio de soluciones que, dependiendo del orden del sistema, puede llegar a ser infinito, pues la dimensión del espacio de solución es exactamente $n!$, siendo n el tamaño del problema a analizar.

Dada la naturaleza combinatorial del problema, es posible adaptarla al enfoque metodológico en el que se basa el Simulated Annealing. Para ello, es necesario definir la función objetivo que se pretende minimizar y algunos parámetros necesarios para llevar a cabo dicho proceso. A continuación, se describe la manera en que es posible realizar dicho adaptamiento.

6.2.1.1. Función Objetivo

La función objetivo de este problema evalua el número de *fill-in* generados a partir de una combinación pre-establecida de variables. Esto se logra simulando la inversa de la matriz de coeficientes y cuantificando el número de elementos de relleno generados.

6.2.1.2. Determinación de la población

Se parte del hecho que es conocida una configuración inicial a partir de la cual es posible generar configuraciones futuras a través de una metodología aleatoria fundamentada en el criterio de vecindad que se analizará más adelante en detalle.

6.2.1.3. Longitud de la cadena

Este importante parámetro tiene una relación estrecha con el tamaño del problema, por lo cual se define la longitud de la cadena como un múltiplo entero de dicha magnitud. Así:

$$N_0 = kn \tag{6.2}$$

donde:

- N_0: Longitud de la cadena inicial.
- k: Constante de proporcionalidad[1].
- n: Tamaño del problema a ser resuelto.

6.2.1.4. Tasas de variación de parámetros

Tal como se analizó en el capítulo 3 existe una relación inversa entre la variación de la temperatura y la longitud de la cadena, así, se ha hecho un estudio de estos dos parámetros llegando a resultados satisfactorios con los cuales la tasa de enfriamiento se ha definido al 4 % y una tasa de incremento de la cadena del 5 %. Es necesario aclarar que estos parámetros varían de acuerdo al tipo y al tamaño del problema a analizar.

6.2.2. Descripción de la metodología de ordenamiento

Partiendo del hecho de que los sistemas a analizar presentan la característica de simetría estructural, ayuda en gran medida a plantear de forma rápida la función objetivo del problema, esta es, por supuesto una cuantificación del número de elementos de relleno que surgen del proceso de inversión de acuerdo a determinada configuración o secuencia de variables.

La idea que surge para la evaluación de dicha cantidad es determinar el número de elementos nulos en la matriz original, éste valor se denomina f_1. Posteriormente, se lleva la matriz a su forma escalonada aplicando eliminación Gaussiana y se cuantifica el número de elementos nulos que existen en la matriz triangular superior. A este valor se le denomina f_2. Claramente, puede analizarse que el número de elementos de relleno es igual a la diferencia entre f_1 y $2f_2$. Así, la función objetivo se puede plantear como:

$$f_i = f_1 - 2f_2 \tag{6.3}$$

Es importante aclarar que f_1 es independiente del orden que se le de a la matriz, contrario a f_2 que es función del orden en que se procesen las variables.Además, es necesario

[1]Para fines de la presente metodología se define la constante $k = 1$.

duplicar f_2, ya que la información obtenida de la matriz escalonada es únicamente de un sólo factor de la matriz original y por la propiedad de simetría de la matriz este valor es igual al número de ceros existente en la matriz triangular inferior.

El orden en que se procesan las variables es una decisión aleatoria basada en el criterio de vecindad, tal como se vio en el capítulo 3. Partiendo de una configuración inicial, es necesario determinar la temperatura inicial del problema, para ello se procede como se indica en el flujograma presentado en la figura 6.1.

Una vez obtenido este valor, es posible dar inicio al algoritmo del ***simulated annealing***, el cual se representa en la figura 6.2. Es necesario describir las diferentes funciones que cumplen algunas de las variables que intervien en el proceso anterior, a decir: el criterio de parada del proceso se ha definido a través de la temperatura del sistema, la cual debe ser mayor a 0.5. Dado que la metodología no tiene la posibilidad de identificar la mejor función objetivo, ya que una de sus características es salirse de óptimos locales aceptando soluciones de peor calidad, entonces ésta debe ser filtrada a través de la incumbente ($x_q \Rightarrow q$). Además de lo anterior, es preciso aclarar la importancia de la selección de los parámetros de decremento de la temperatura y de incremento de la cadena (α y β), los cuales difieren según el tamaño del problema a analizar. De la acertada selección que se haga de estos dos parámetros depende en buena medida la eficiencia de la metodología.

Al obtener el orden de variables que genere una función objetivo mínima, es posible comenzar con el almacenamiento de la función, tema que se discute a continuación.

6.3. Algoritmo de Almacenamiento

Básicamente se adopta el esquema de Zollenkopf descrito en el capítulo 3, para ello es necesario ordenar la matriz de coeficientes según lo indique el orden óptimo de las variables en el proceso anterior. Adicional al número de elementos de relleno que se genera a partir de esta combinación de variables, es necesario identificar la posición en la que surgen estos elementos a medida que se desarrolla el proceso de inversión de la matriz, para de igual forma guardarles un espacio en el esquema de almacenamiento.

Como el propósito de este almacenamiento es tambien generar la solución del sistema operando desde un esquema compacto, es necesario acoplar ciertas metodologías de resolución de sistemas de ecuaciones lineales como la descomposición triangular para tal fin. Para ello, se diseñaron tres algoritmos complentarios que satisfacen a cabalidad estas necesidades. Es decir, un algoritmo que almacene el sistema de forma compacta, un algoritmo que se encargue de desarrollar la descomposición triangular bajo este esquema de almacenamiento y finalmente, un algoritmo que de solución al sistema. Estas metodologías se describen de forma general en los diagramas de flujo correspondiente a las firguras 6.3– 6.5.

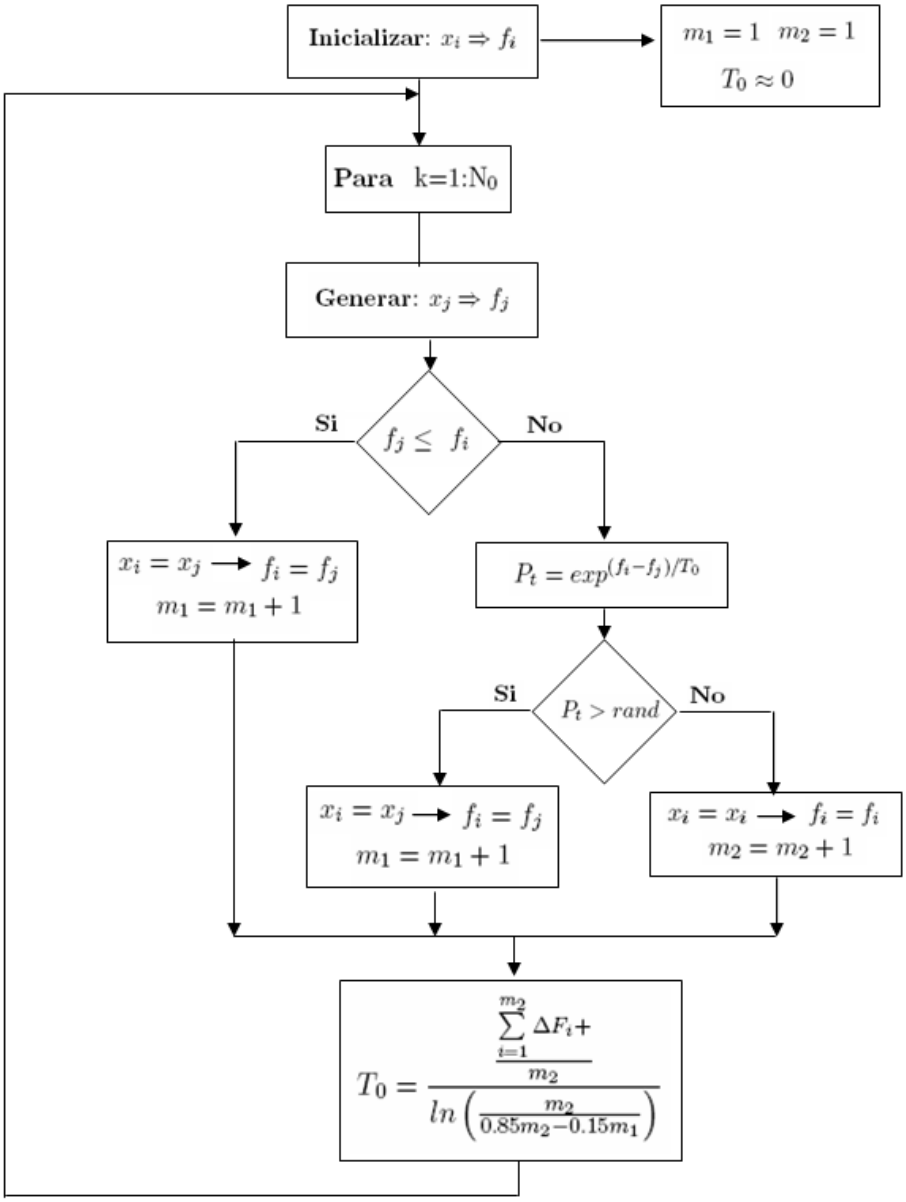


Figura 6.1: Flujograma representativo para determinar la temperatura inicial del proceso

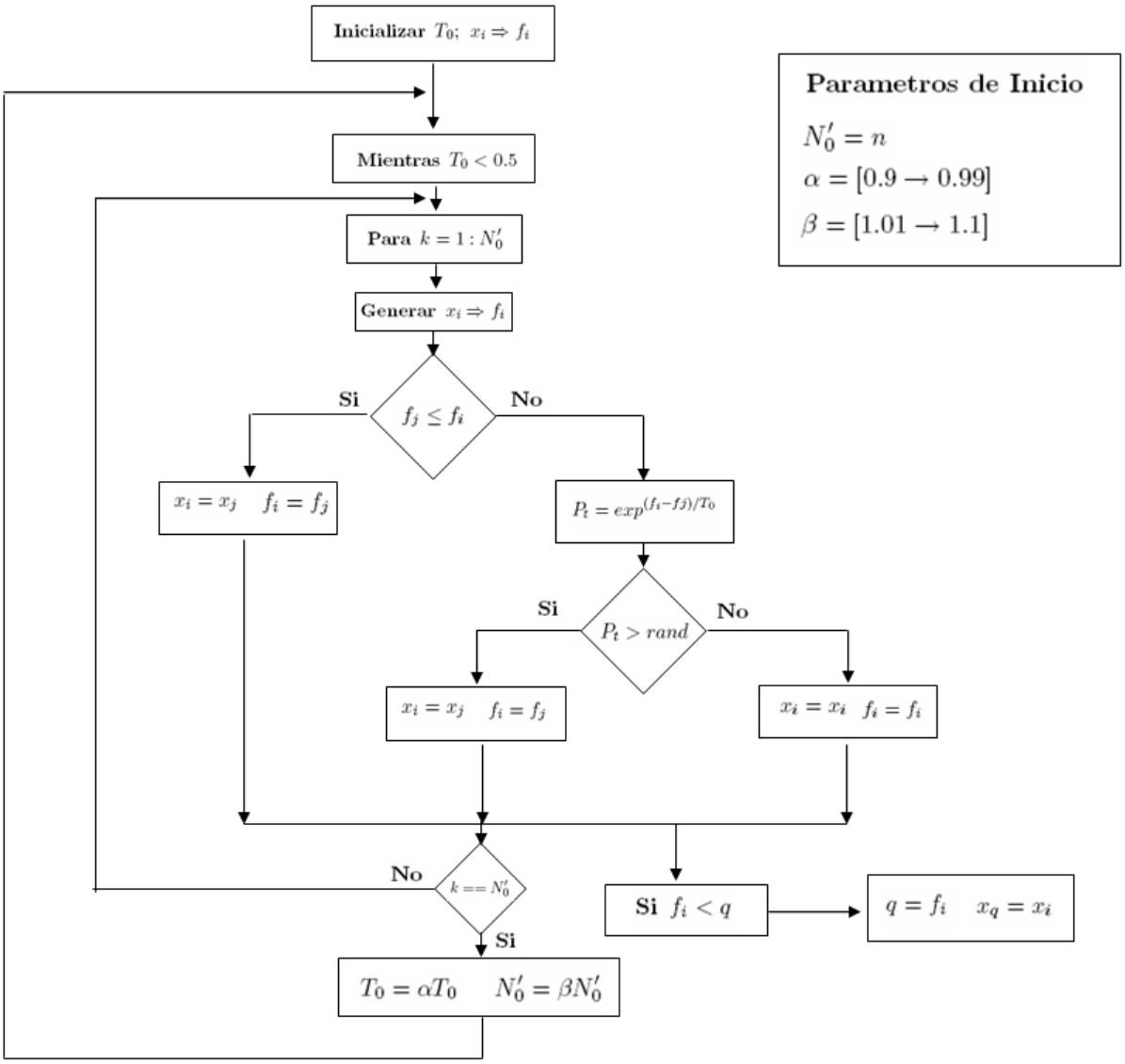


Figura 6.2: Flujograma representativo de la metodología de optimización.

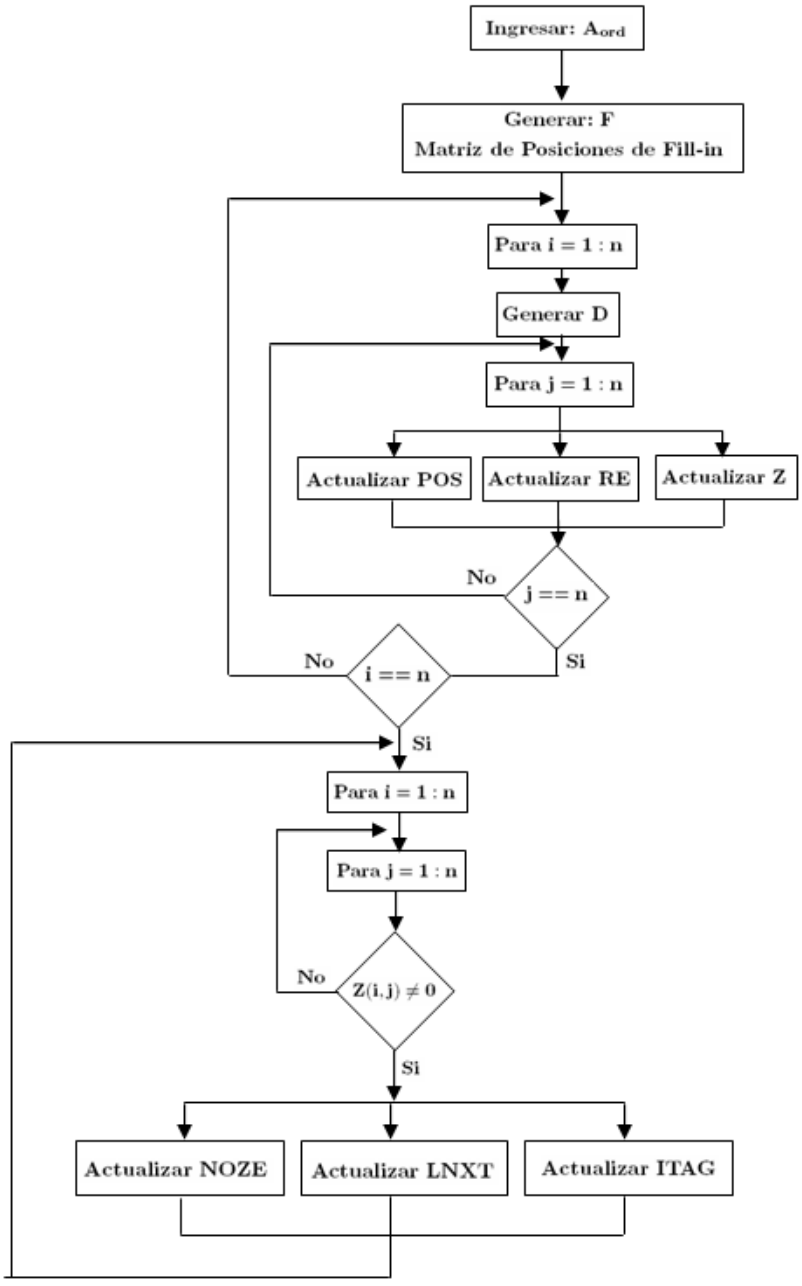


Figura 6.3: Flujograma representativo del almacenamiento del sistema

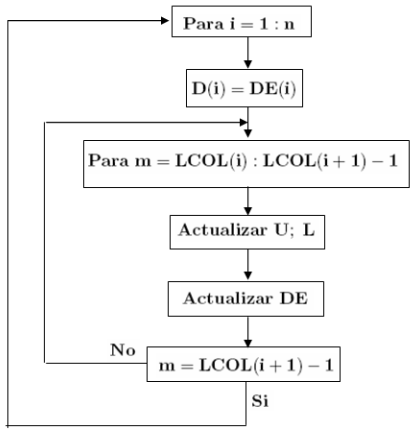


Figura 6.4: Flujograma representativo de la descomposición LDU

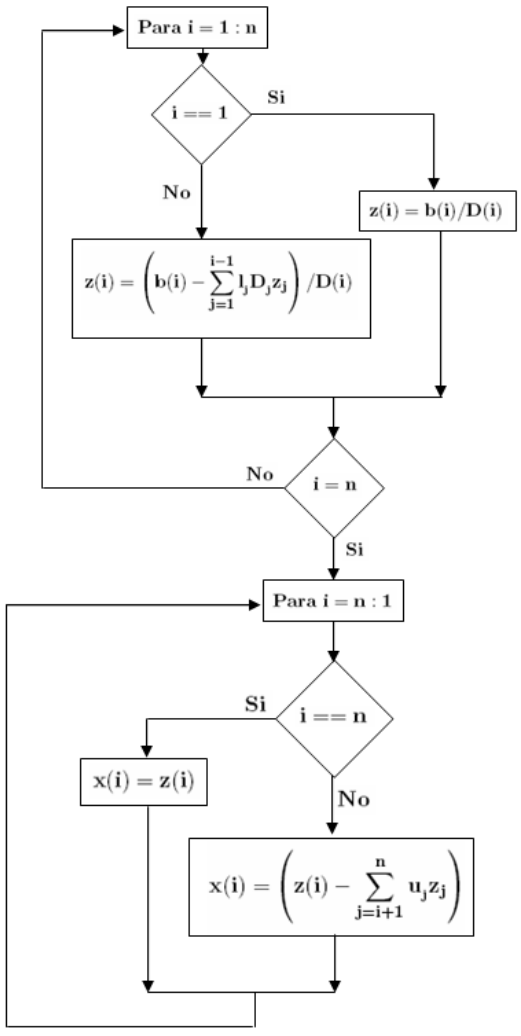


Figura 6.5: Flujograma representativo de solución del sistema

Los algoritmos se programaron en la plataforma de MatLab y es necesario hacer notar que son aplicados a problemas simetricos estructuralmente, esto es debido principalmente al programa de optimización, el cual sólo evalua el número de fill-in en matrices que presenten esta característica, todas la variables que aparecen en estos flujogramas han sido explicadas con anterioridad a diferencia de la matriz Z, la cual almacena las posiciones de todos los elementos del sistma incluyendo los elementos de relleno que suegun.

6.4. Ejemplos de Aplicación

6.4.1. Ordenamiento Óptimo de un Sistema Eléctrico Radial

Considere el siguiente sistema eléctrico radial conformado por 10 nodos y 9 ramas.

Figura 6.6: Sistema eléctrico de 10 nodos y 9 ramas

La estructura de la matriz de coeficientes de este sistema, aplicando el orden indicado en la figura 6.6 es la siguiente:

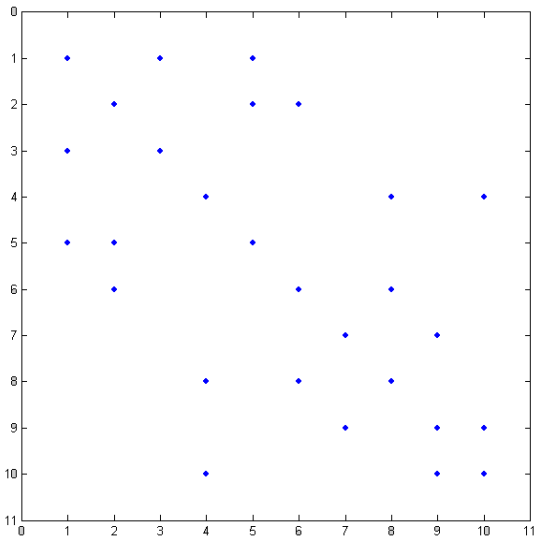


Figura 6.7: Estructura del sistema con el orden indicado.

Es necesario ingresar este orden al programa para dar inicio al proceso de óptimización, donde:

$$\mathbf{x_i} = [9\ 7\ 10\ 4\ 8\ 6\ 1\ 5\ 2\ 3]$$

y

$$\mathbf{f}(\mathbf{x_i}) = 6$$

El valor de la función es obtenido fácilmente simulando la inversa de la matriz por eliminación de Gauss. Al introducir estas variable de iniciación en el programa del ***simulated Annealing***, este obtiene una respuesta óptima en tan sólo 0.016 segundos, generando el siguiente orden:

$$\mathbf{x_i} = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$$

con una función objetivo igual a:

$$\mathbf{f}(\mathbf{x_i}) = 0$$

La estructura del sistema para este orden es el siguiente:

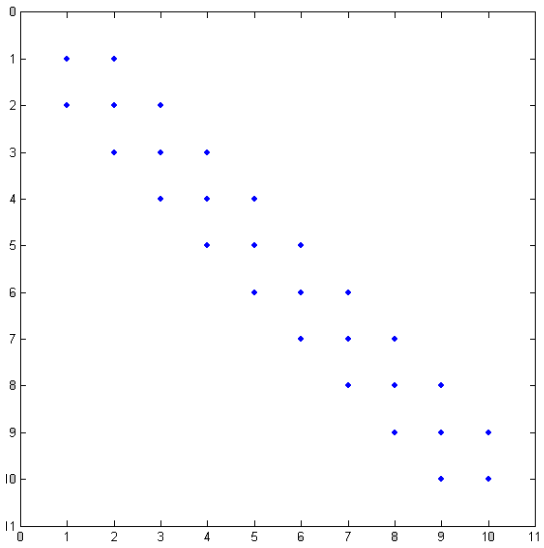


Figura 6.8: Estructura del Sistema radial con el orden óptimo.

6.4.2. Solución de un Sistema de Ecuaciones, Ordenado y Almacenado de Forma Compacta

El objetivo que tiene la presentación de este ejemplo es enseñar la eficiencia computacional que presenta un ordenamiento a priori a la solución del sistema. Para ello, considérese el siguiente sistema de ecuaciones lineales:

$$\begin{bmatrix} 4 & -1 & & & -1 & & & -1 & & & \\ -1 & 4 & -1 & & & & & & & & \\ & -1 & 4 & -1 & & & & & & & \\ & & -1 & 1 & & & & & & & \\ -1 & & & & 3 & -1 & & -1 & & & \\ & & & & -1 & 2 & -1 & & & & \\ & & & & & -1 & 1 & & & & \\ -1 & & & & -1 & & & 3 & -1 & & \\ & & & & & & & -1 & 2 & -1 & \\ & & & & & & & & -1 & 1 & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 6 \\ 3 \\ 7 \\ 2 \\ 5 \\ 8 \\ 1 \\ 3 \end{bmatrix} \tag{6.4}$$

Si se comezara el proceso de solución de este sistema por eliminación gaussiana o factorización LDU, se evidencia con facilidad que aparecen 16 elementos de relleno, esto puede observarse en el vector **CE** del siguiente esquema compacto.

LCOL=[1 4 8 12 15 21 24 26 34 36]

NOZE=[3 4 4 3 6 3 2 8 2 1]

ITAG=[2 5 8 1 3 5 8 2 4 5 8 3 5 8 1 2 3 4 6 8 7 8 6 8 1 2 3 4 5 6 7 9 8 10 9]

LNXT=[2 3 0 5 6 7 0 9 10 11 0 13 14 0 16 17 18 19 20 0 22 23 0 25 0 27 28 29 30 31 32 33 0 35 0 0]

DE=[4 4 4 1 3 2 1 3 2 1]

CE=[-1 -1 -1 -1 -1 0 0 -1 -1 0 0 -1 0 0 -1 0 0 0 -1 -1 -1 -1 0 -1 0 -1 0 0 0 -1 0 0 -1 -1 -1 -1]

Por lo tanto sería necesario almacenar 16 elementos adicionales debido a que este sistema no fue analizado con anterioridad, en lo que a un ordenamiento óptimo se refiere. Si dicho análisis se desarrolla previamente, es posible generar el siguiente sistema con orden óptimo:

$$\begin{bmatrix} 1 & & & & -1 & & & & & \\ & 1 & & & & -1 & & & & \\ & & 1 & -1 & & & & & & \\ & & -1 & 2 & & & & -1 & & \\ -1 & & & & 4 & & -1 & & & \\ & -1 & & & & 2 & & -1 & & \\ & & & & -1 & & 4 & & & -1 \\ & & & & & -1 & & 3 & -1 & -1 \\ & & & -1 & & & & -1 & 3 & -1 \\ & & & & & & -1 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_4 \\ x_7 \\ x_{10} \\ x_9 \\ x_3 \\ x_6 \\ x_2 \\ x_5 \\ x_8 \\ x_1 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 3 \\ 1 \\ 6 \\ 2 \\ 1 \\ 7 \\ 8 \\ 4 \end{bmatrix} \tag{6.5}$$

Obsérvese que tanto las variables como los elementos de vector independiente del sistema inicial han variado de posición. Con este orden de procesamiento de variables, no se genera ningún elemento de relleno, con lo que se debe reducir el número de operaciones necesarias para la llegar a la solución del sistema en general. El esquema de almacenamiento compacto para el sistema ordenado se enseña a continuación:

LCOL=[1 2 3 4 6 8 10 12 15 18]

NOZE=[1 1 1 2 2 2 2 3 3 3]

ITAG=[5 6 4 3 9 1 7 2 8 5 10 6 9 10 4 8 10 7 8 9]

LNXT=[0 0 0 5 0 7 0 9 0 11 0 13 14 0 16 17 0 19 20 0]

DE=[1 1 1 2 4 2 4 3 3 4]

CE=[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

Es posible observar que el ordenamiento óptimo de una matriz no sólo reduce el número de operacioens necesarias para llegar a la solución del sistema, sino también, reduce el espacio de memoría requerido para el almacenamiento de la matriz[2]. Para este sistema ordenado, se tiene que su descomposición triangular compacta sería:

L=[-1.00 -1.00 -1.00 -0.33 -1.00 -1.00 -0.50 -0.27 -0.50 -1.00]

D=[1.00 1.00 1.00 1.00 3.00 1.00 3.66 2.00 1.50 1.72]

U=[-1.00 -1.00 -1.00 -1.00 -0.33 -1.00 -0.27 -0.50 -0.50 -1.00]

Con lo cual es posible obtener la solución del sistema:

$$\mathbf{x} = [8{,}00\ 43{,}33\ 37{,}66\ 34{,}66\ 5{,}00\ 38{,}33\ 6{,}00\ 31{,}33\ 30{,}66\ 18{,}00]$$

6.4.3. Obtención de los Elementos de la Matriz Z_{BUS} a partir del Almacenamiento de la Matriz Y_{BUS}.

Este ejemplo tiene como objeto evidenciar que el esquema de almacenamiento compacto que se adopto en este trabajo es lo suficientemente flexible como para extraer determinada columna de la inversa de un sistema, información funtamental en el estudio de corto circuito de sistemas eléctricos de potencia. Para ello considere la siguiente matriz de admitancia nodal tomada de [19].

$$\mathbf{Y_{BUS}} = \begin{bmatrix} -j30{,}0 & j10{,}0 & 0 & j20{,}0 & 0 \\ j10{,}0 & -j26{,}2 & -j16{,}0 & 0 & 0 \\ 0 & j16{,}0 & -j36{,}0 & 0 & j20{,}0 \\ j20{,}0 & 0 & 0 & -j20{,}0 & 0 \\ 0 & 0 & j20{,}0 & 0 & -j20{,}0 \end{bmatrix}$$

[2]Esto se hace evidente si se comparan las dimensiones de los vectores del esquema de almacenamiento de los sitemas ordenado y sin ordenar.

El esquema de almacenamiento aplicado a la matriz de admitancia nodal genera los siguientes vectores:

LCOL=[1 3 6 9 13]

NOZE=[2 3 3 4 2]

ITAG=[2 4 1 3 4 2 4 5 1 2 3 5 3 4]

LNXT=[2 0 4 5 0 7 8 0 10 11 12 0 14 0]

DE=[-j30.0 -j26.2 -j36.0 -j20.0 0 -j20.0]

CE=[j10 j20 j10 j16 0 j16 0 j20 j20 0 0 0 j20 0]

El vector independiente para generar la columna cuatro de la matriz de impedancias $\mathbf{Y}_{\mathbf{BUS}}$ se debe definir como:

b=[0 0 0 1 0]

Así, las solución del sistema genera lo siguiente:

x=[j5.1 j5.0 j5.0 j5.15 j5.0]

lo cual es precisamente lo que se pretendía obtener. Es evidente que el uso del almacenamiento compacto de matrices es ineficiente cuando es aplicado a problemas de pequeña escala, en el cual el grado de dispersidad es bajo. Pero, cuando es preciso operar con problemas de gran tamaño, como lo son los sistemas eléctricos reales, el uso de esta metodología es una valiosa herramienta que están a la vanguardía con los sistemas contemporáneos donde es preciso obtener respuestas confiables en tiempo real.

Capítulo Septimo

Pruebas y Análisis de Resultados

El objetivo fundamental del desarrollo del presente proyecto consiste en encontrar el orden óptimo de un sistema de ecuaciones lineales que minimice el número de operaciones a realizar al momento de solucionarlo, y con esto, generar un error mínimo en la respuesta del sistema. Para interpretar mejor la reducción del error, se ha explicado con anterioridad que el almacenamiento compacto de la matriz de coeficientes, después de ésta ser ordenada, contiene el mínimo número de elementos necesarios para la solución del sistema.

El presente capítulo pretende validar la metodología diseñada a través de la aplicación en varios sistemas de interés. Para ello, se presentarán ciertos sistemas de ecuaciones lineales que bien pueden representar cualquier sistema físico con características dispersas, pero que para efectos de análisis de resultados representan sistemas eléctricos de prueba IEEE. El orden en que se efectúan las pruebas será el siguiente:

- Ordenamiento óptimo del sistema a través del Simulated Annealing.
- Almacenamiento compacto del sistema usando el esquema de Zollenkopf.
- Solución del Sistema Compacto a travez de la factorización **LDU**.

Para ilustrar la forma en que se presentarán los resultados se comenzará con el desarrollo de un sistema de pequeña escala utilizado como ejemplo en el capítulo 4. Esto es con el fin de proporcionar un mejor entendimiento de la información que se generará a través de la metodología implementada en este proyecto; sin embargo, para fines de la verdadera sustentación de los resultados sólo se generará la información más relevante, ya que los sistemas son de gran escala y sería tedioso para el lector interpretar información poco útil para tal fin.

7.1. Sistema de Prueba de Fase 90

Se retomará el sistema eléctrico de 17 barras presentado en el capítulo 4, el cual se ilustra en la figura 7.1. Para la finalidad de este ejemplo se asume que todas las impedancias de rama son iguales a 1Ω, este valor es irrelevante ya que lo que se pretende ilustrar es el análisis de los resultados de la metodología. La estructura matricial de este

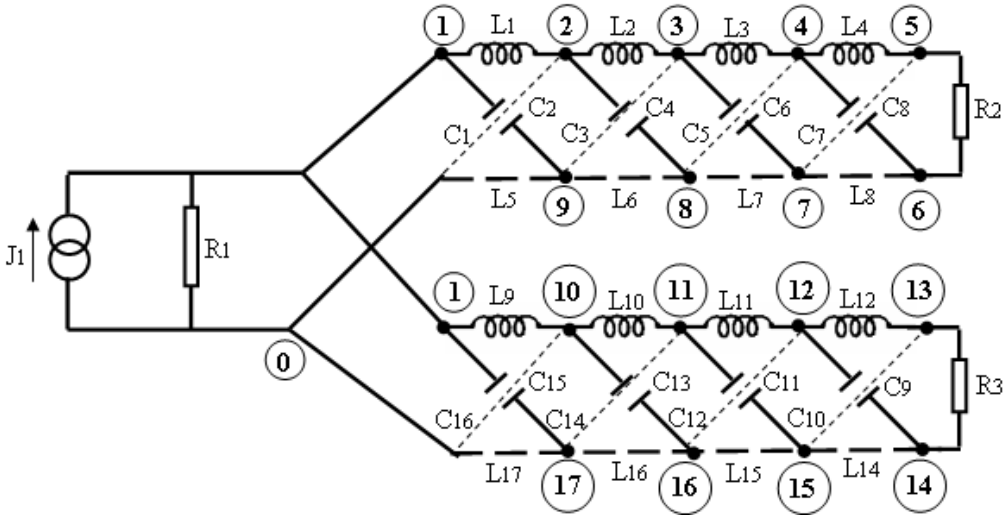


Figura 7.1: Sistema de prueba de 17 barras

sistema se representa en la figura 7.2

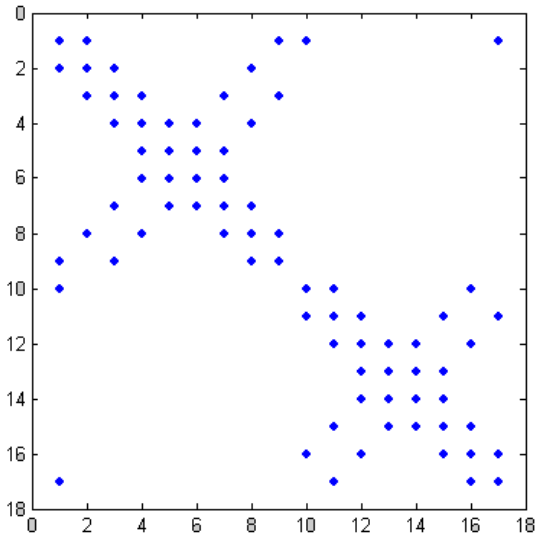


Figura 7.2: Estructura matricial del sistema sin ordenar. Numero de *fill-in=68*

Ingresando este sistema matricial de la manera adecuada al programa de optimización fundamentado en el ***Simulated Annealing***, se genera el siguiente orden de variables

con el cual se obtiene un número mínimo de elementos de relleno.

$$\mathbf{x_{opt}} = [5\ 6\ 13\ 14\ 4\ 7\ 12\ 15\ 3\ 8\ 11\ 16\ 2\ 9\ 10\ 17\ 1]$$

por lo tanto, se genera una función óbjetivo igual a:

$$f(x) = 12$$

cabe recordar que esta función objetivo representa el número de elementos de relleno generados con la secuencias de variables del vector **x**. El tiempo de cómputo invertido en la obtención de este orde fue de 4.3 segundos, habiendo ejecutado el programa desde la plataforma de MATLAB. Es necesario resaltar que este orden no es único y que la incumbente del programa revela que existen varias conbinaciones con las cuales se obtiene una función objetivo de igual valor, en este caso, es posible generar una función objetivo con la misma calidad en 0.67 segundos. La representación del sistema aplicando este orden de variables se ilustra en la figura **??**

Después de generada esta información, es necesario empaquetar el sistema. Para ello se hace uso del programa de almacenamiento, al cual es necesario ingresarle la matriz A y el orden de las variables a ser almacenadas.

LCOL=[1 4 7 10 13 18 23 28 33 38 43 48 53 57 61 65 69]

NOZE=[3 3 3 3 5 5 5 5 5 5 5 5 4 4 4 4 4]

DE=[3 3 3 3 4 4 4 4 4 4 4 4 3 3 3 3 5]

CE=[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1
0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 0 -1 -1 -1 -1 -1]

Nótese claramente los ceros que aparecen en el vector **CE**, los cuales indican o hacen referencia a las posiciones en las cuales aparecerá un elemento *fill-in* en el proceso de inversión del sistema. Con esta información es posible comenzar la factorización compacta del sistema, así, se generan las siguientes matrices almacenadas de forma compacta.

L=[-1/3 -1/3 -1/3 -1/2 -1/3 -1/2 -1/3 -1/3 -1/2 -1/3 -1/2 -1/3 -1/3 -1/2 -1/3 -1/2 -1/3
-1/3 -1/2 -1/3 -1/2 -1/3 -1/3 -1/2 -1/3 -1/2 -1/2 -1/3 -1/2 -1/3 -1/2 -1/2 -1/2 -1 -1/2
-1]

D=[3 8/3 3 8/3 3 8/3 3 8/3 3 8/3 3 8/3 2 3/2 2 3/2 1]

U=[-1/3 -1/3 -1/3 -1/2 -1/2 -1/3 -1/3 -1/3 -1/2 -1/2 -1/3 -1/3 -1/3 -1/2 -1/2 -1/3 -1/3
-1/3 -1/2 -1/2 -1/3 -1/3 -1/3 -1/2 -1/2 -1/3 -1/3 -1/3 -1/2 -1/2 -1/2 -1/2 -1 -1/2 -1/2
-1]

Finalmente, es posible obtener la solución del sistema. Es necesario aclarar antes de efectuar este procedimiento que el vector independiente debe ser ordenado según lo indica el vector de variables x para poder que la solución sea la correcta. Así, se tiene que:

$\mathbf{x}$ =[24 24 24 24 23.5 23.5 23.5 23.5 22.5 22.5 22.5 22.5 21 21 21 21 17]

Finalmente, el número de operaciones [1] realizadas sobre el sistema almacenado y ordenado corresponde a 106, en contraste con las 162 operaciones efectuadas si el sistema se almacena pero no se ordena.

7.2. Sistemas de prueba IEEE

La metodología propuesta se aplicó a diversos sistemas de prueba estandarizados en la IEEE. Básicamente, se aplicó el flujo de carga DC en cada uno de los sistemas empleando el empaquetamiento compacto de la matriz de coeficientes. Los resultados obtenidos se resumen en la tabla 7.1. En esta tabla se presentan los resultados del desarrollo del flujo de carga DC, tanto sin emplear un ordenamiento previo como aplicando la metodología basada en el Simulated Annealing. Esta información se presenta de este modo con el fin de evidenciar la utilidad que se obtiene al ordenar previamente la matriz de coeficientes antes de desarrollar el sistema como tal.

Sistema	**Sin Ordenar**		**Ordenado**	
	Fill-in	**No. Operaciones**	**Fill-in**	**No. Operaciones**
IEEE-14	14	112	8	76
IEEE-30	174	316	28	170
IEEE-57	790	1060	118	388
IEEE-118	1692	2286	168	762
IEEE-300	14256	15674	514	1932

Cuadro 7.1: Resultados obtenidos en sistemas de prueba IEEE

Un comportamiento interesante a resaltar de los resultados obtenidos en la tabla anterior, consiste en la relación porcentual que existe entre el número de operaciones realizadas antes y después de la matriz ser ordenada. Entre mayor sea el tamaño del sistema, el porcentaje de disminución del número de operaciones aumenta. Por ejemplo, para el sistema de prueba IEEE-14 este porcentaje es del 32.14 %, mientras que para el sistema IEEE-300 la cifra es del 87.67 %. En cuanto a la reducción del número de elementos de *"relleno"*, se puede observar que la relación también es directa, pues a medida que el número de barras del sistema es mayor, el porcentaje de disminución de *fill-in* aumenta. Así, para los sistemas de prueba IEEE-14 e IEEE-57 los porcentajes indican una disminución del 42.86 % y 85.06 % respectivamente, para el sistema de 300 barras esta cifra llega al 96.34 %. Estas dos características hacen que la metodología propuesta sea aún más interesante, pues como es sabido, en la práctica los sistemas reales presentan un gran número de nodos y es aquí donde el algoritmo toma su mayor dimensión y eficacia.

[1] Una operación equivale a realizar una multiplicación (división) o una suma (resta).

Sistema	**Porcentaje de disminución**	
	Fill-in	**No. Operaciones**
IEEE-14	42.86 %	32.14 %
IEEE-30	83.91 %	46.20 %
IEEE-57	85.06 %	63.40 %
IEEE-118	90.07 %	66.67 %
IEEE-300	96.39 %	87.67 %

Cuadro 7.2: Disminución porcentual de operaciones y elementos de relleno

En la tabla 7.3 se ilustran diferentes metodologías de ordenamiento fundamentadas en las matemáticas aplicadas al problema. La base de datos utilizada para este análisis corresponde al sistema IEEE-118. Esta tabla contiene el número de *fill-in* resultante, al igual que el número de operaciones necesarias para dar solución al sistema empleando cada una de las metodologías citadas.

Métodología	**Resultados**	
	Fill-in	**No. Operaciones**
Sch2	1055	2812
Sch3	1097	2887
GF1	846	2341
GF2	833	2342
GF3	868	2522
MDML	796	2261
MLMD	762	2448
MFI-1	759	2261
MFI-2	732	2197
Annealing	168	762

Cuadro 7.3: Comparación de resultados entre diferentes metodologías de ordenamiento

Puede observarse claramente la superioridad de la metodología propuesta en la presente investigación con respecto a las diferentes metodologías de comparación, lo que valida aún más el método. La tabla anterior fue tomada del artículo de Wang y Gooi [20]. A continuación, se explican las diferentes abreviaturas utilizadas en la tabla 7.3.

Sch2 $\longrightarrow$ *Esquema 2 de Tinney*
Sch3 $\longrightarrow$ *Esquema 3 de Tinney*
GF1 $\longrightarrow$ *Algoritmo 1 de Gómez y Franquelo*
GF2 $\longrightarrow$ *Algoritmo 2 de Gómez y Franquelo*
GF3 $\longrightarrow$ *Algoritmo 3 de Gómez y Franquelo*
MDML $\longrightarrow$ *Algoritmo 1 de Betancourt*
MLMD $\longrightarrow$ *Algoritmo 2 de Betancourt*
MFI-1 $\longrightarrow$ *Algoritmo 1 de Wang y Gooi*
MFI-2 $\longrightarrow$ *Algoritmo 2 de Wang y Gooi*
Annealing $\longrightarrow$ *Metodología propuesta en el proyecto*

Para efectos de una mayor ilustración, se han incluido como parte de las herramientas de análisis de resultados, el comportamiento de la incumbente a través de cada ciclo ejecutado del programa de optimización. Como puede observarse, en algunas ocasiones el algoritmo alcanza el óptimo en un tiempo mucho menor al invertido por la totalidad del proceso. No obstante, entre mayor sea el tamaño del sistema analizado, este comportamiento se modifica hasta tal punto, que el mejor resultado obtenido se logra sólo unos cuantos ciclos antes de finalizar las iteraciones. Esta particularidad le proporciona mayor validez a la metodología, ya que para sistemas grandes, que en general son los reales, la convergencia del método mejora sustancialmente. Así mismo, se presentan las diferentes estructuras de las matrices antes y depués de éstas ser ordenadas.

7.2.1. IEEE-14

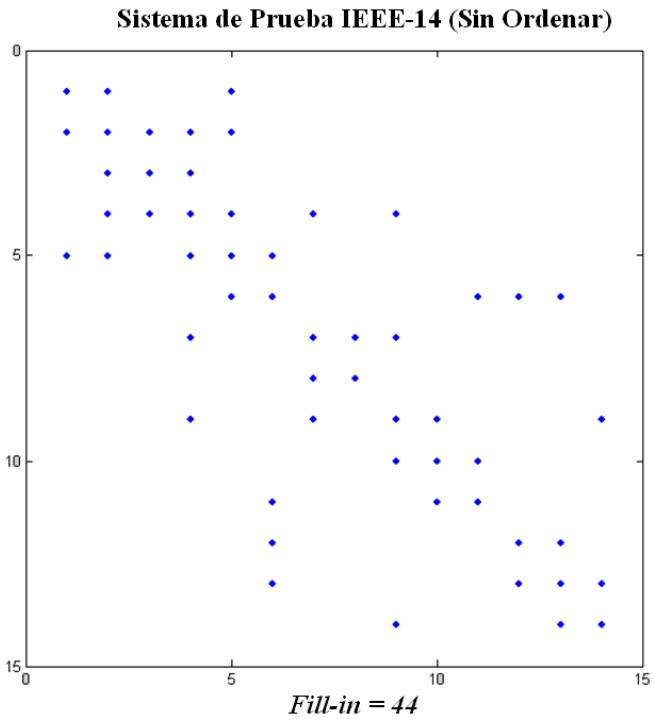


Figura 7.3: Estructura del sistema sin ordenar

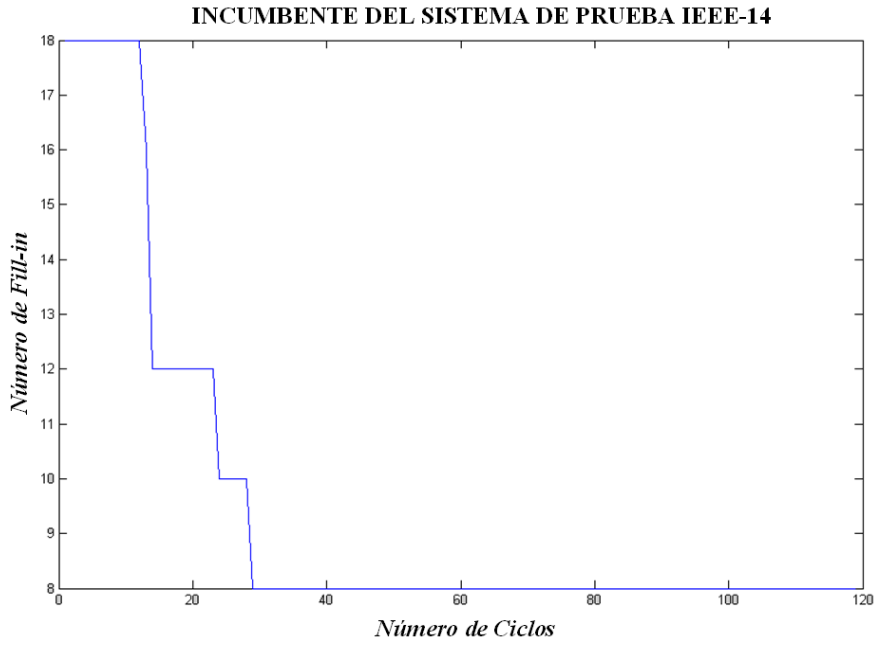


Figura 7.4: Comportamiento de la incumbente

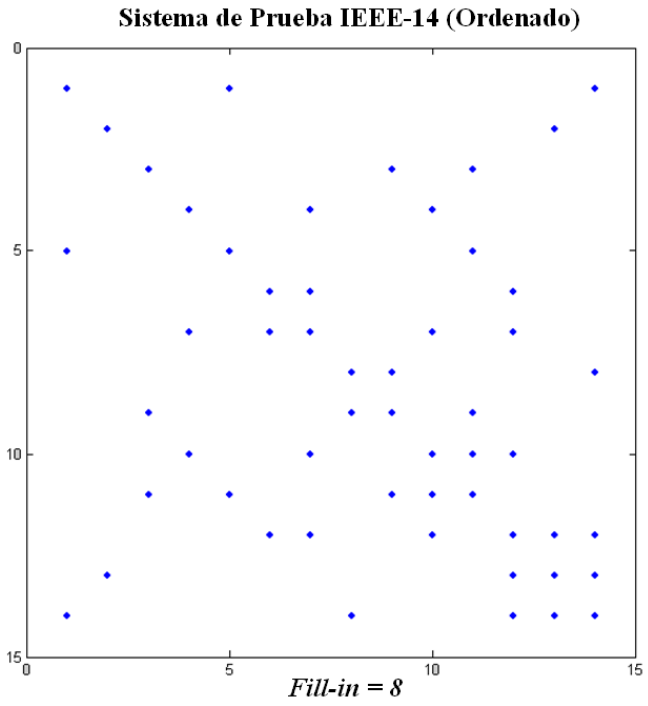


Figura 7.5: Estructura del sistema ordenado

7.2.2. IEEE-30

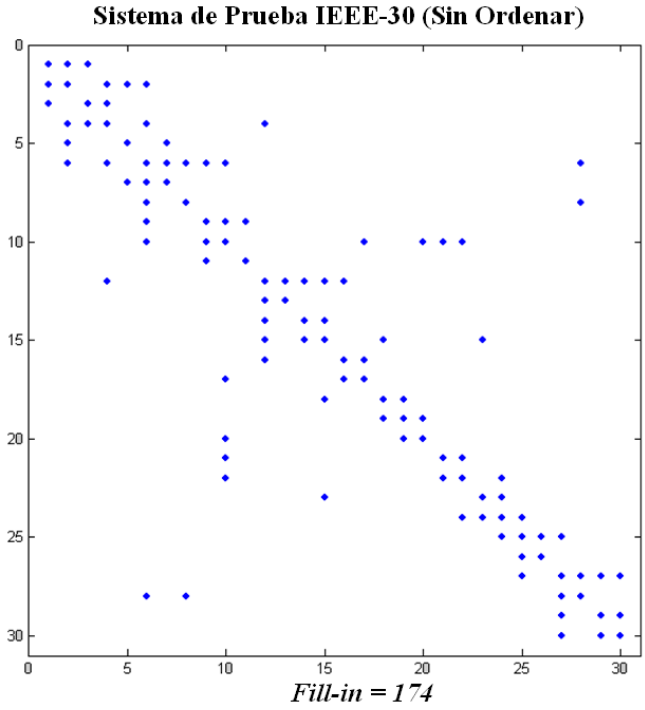


Figura 7.6: Estructura del sistema sin ordenar

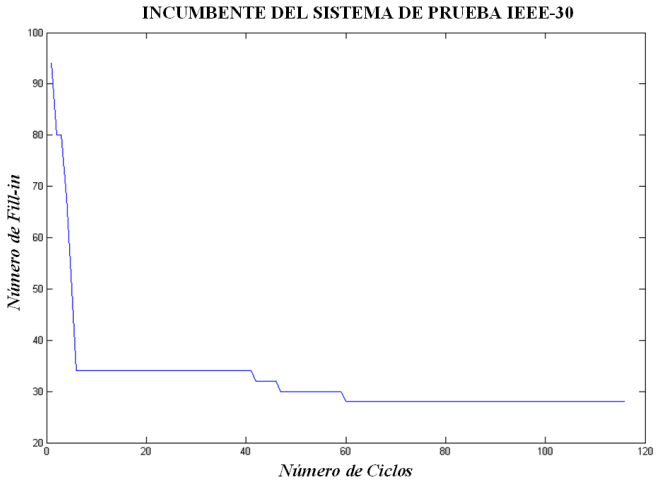


Figura 7.7: Comportamiento de la incumbente sistema

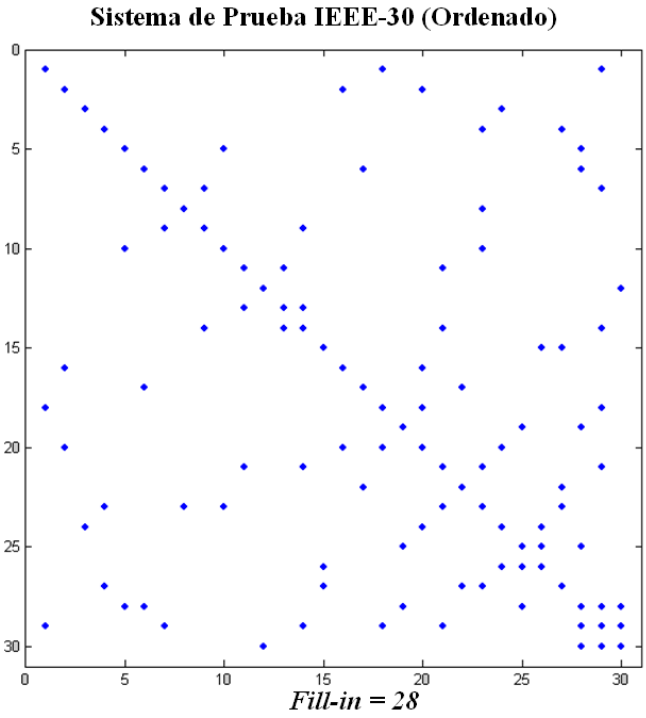


Figura 7.8: Estructura del sistema ordenado

7.2.3. IEEE-57

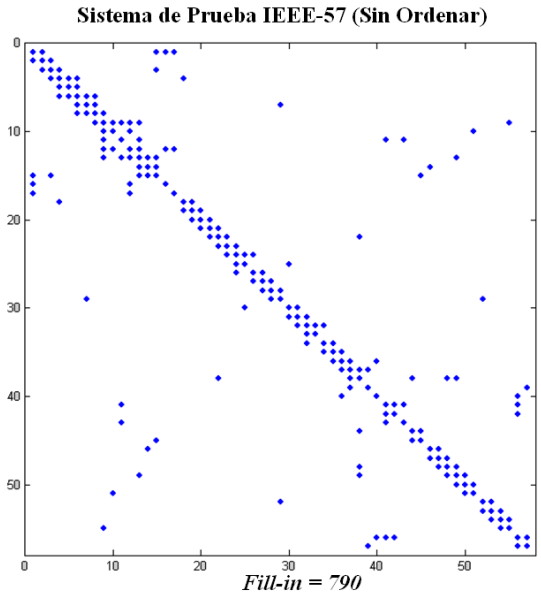


Figura 7.9: Estructura del sistema sin ordenar

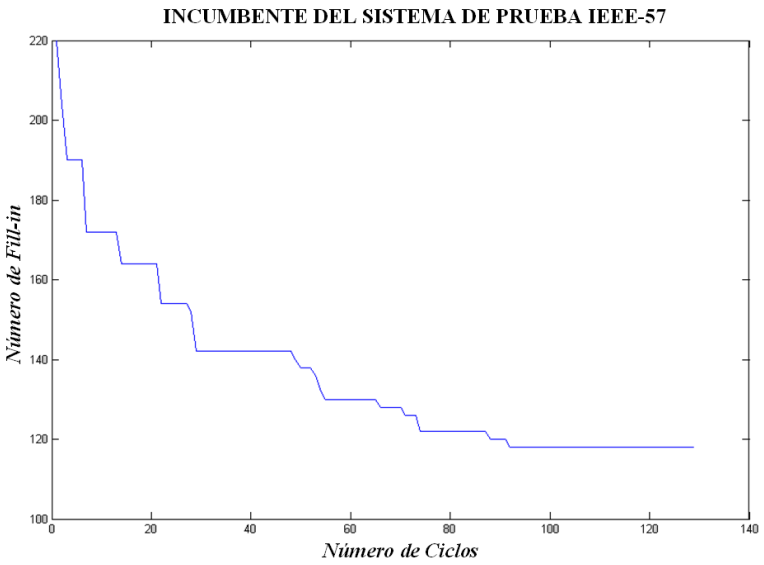


Figura 7.10: Comportamiento de la incumbente

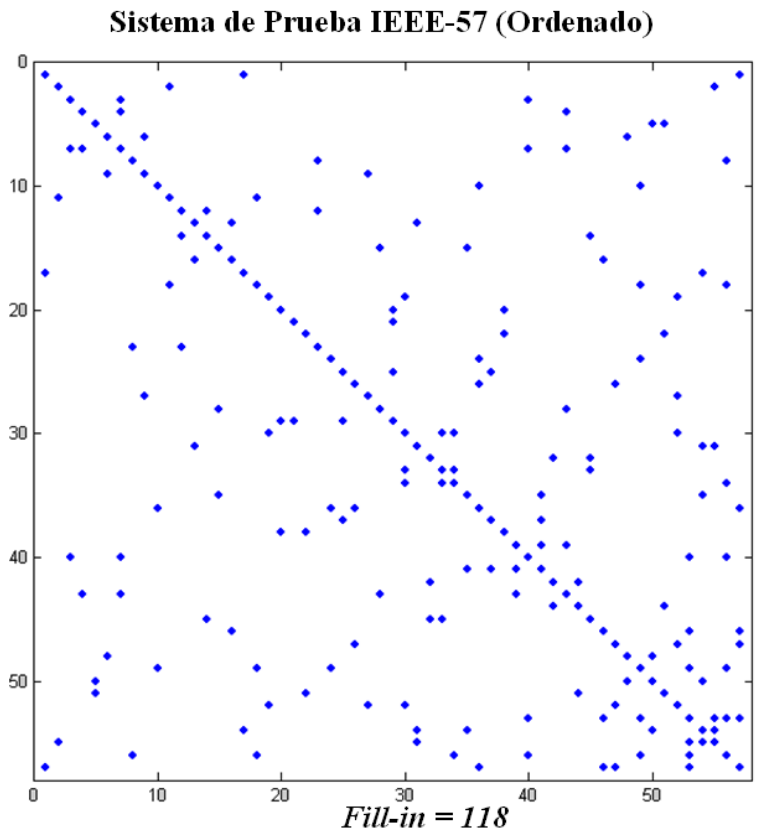


Figura 7.11: Estructura del sistema ordenado

7.2.4. IEEE-118

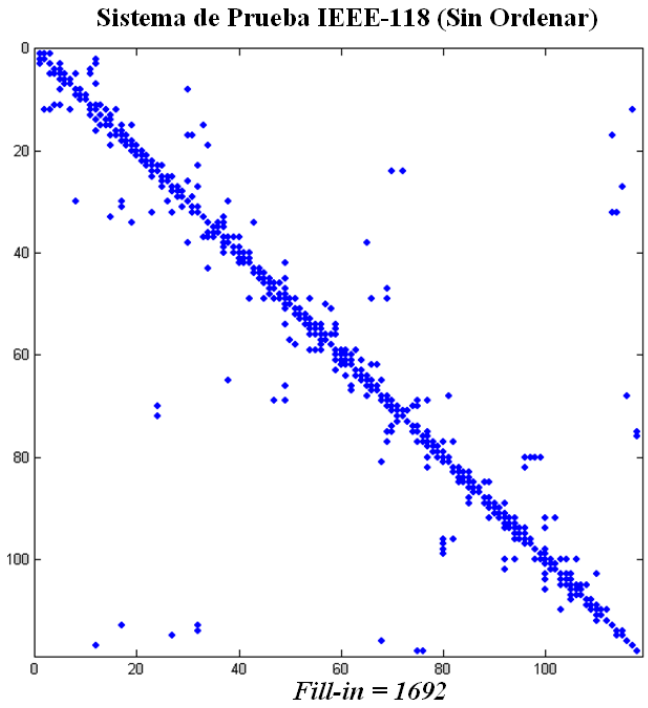


Figura 7.12: Estructura del sistema sin ordenar

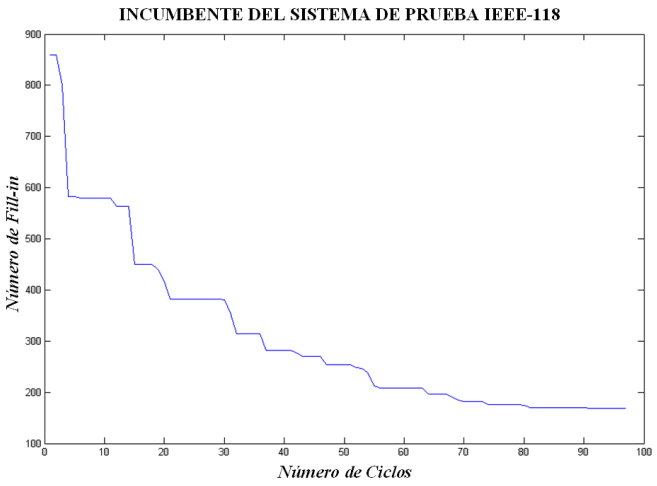


Figura 7.13: Comportamiento de la incumbente sistema

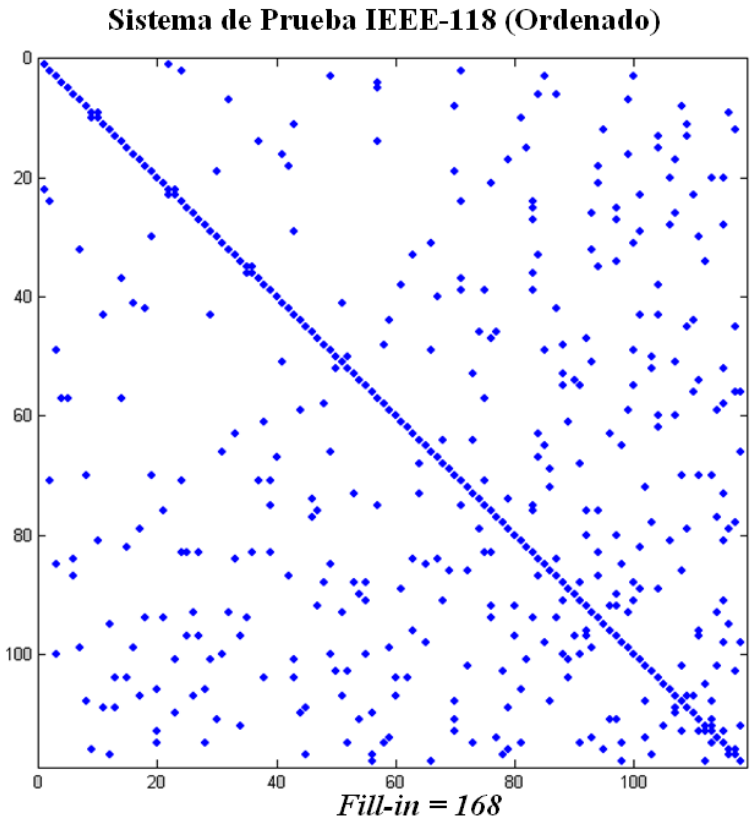


Figura 7.14: Estructura del sistema ordenado

7.2.5. IEEE-300

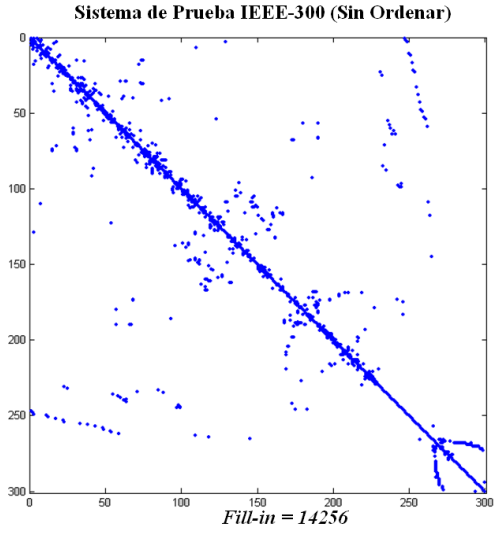


Figura 7.15: Estructura del sistema sin ordenar

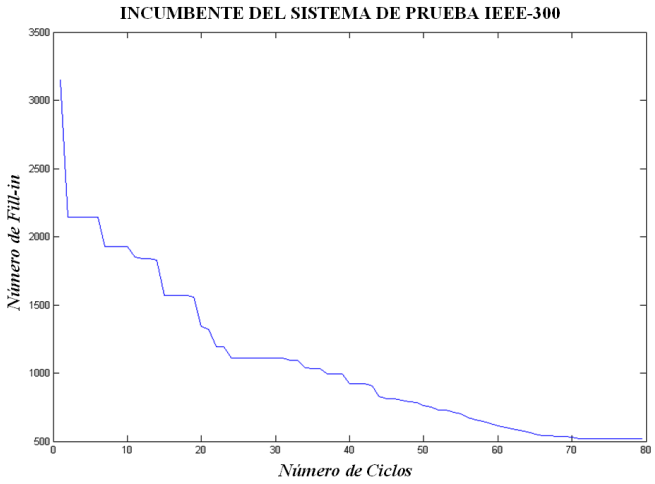


Figura 7.16: Comportamiento de la incumbente

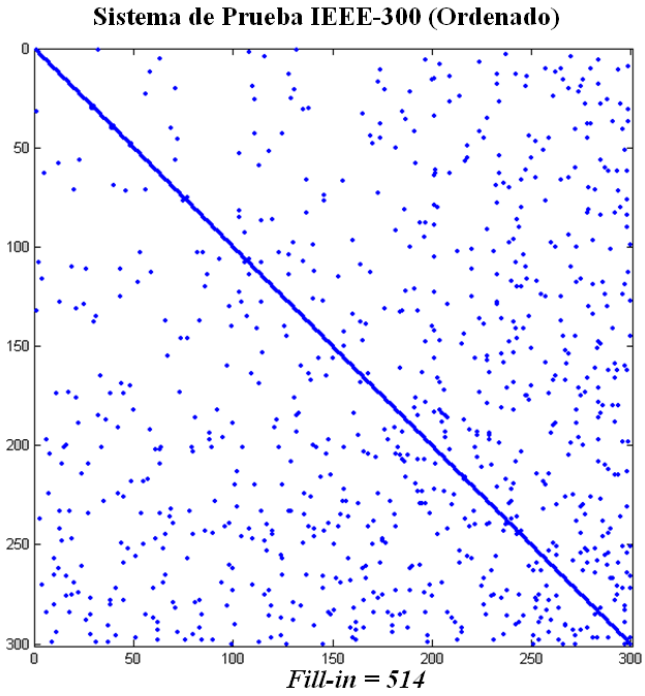


Figura 7.17: Estructura del sistema ordenado

Capítulo Octavo

Conclusiones

A partir del presente proyecto se puede concluir que:

- Se propone una metodología completa para la solución de sistemas de ecuaciones lineales, donde se encuentra el orden del sistema capaz de reducir el número de elementos fill-in; se almacenan los elementos no nulos de la matriz de coeficientes, además de información adicional como apuntadores y se reserva un espacio para los elementos de relleno. Con el fin de aprovechar al máximo la dispersidad de los sistemas a resolver, se plantea la solución a partir de la factorización de la matriz y no recurriendo a la inversión explícita de la misma.

- La aplicación de técnicas de optimización combinatorial como el Simulated Annealing al problema del ordenamiento matricial, arroja excelentes resultados en cuanto a la calidad de la respuesta. Pues como se pudo constatar con las pruebas realizadas durante la investigación, la metodología que se propuso logró mejorar la mejor respuesta obtenida por técnicas convencionales, reduciendo el número de elementos de relleno y, a su vez, el número de operaciones necesarias para la obtención de la solución del sistema de manera significativa.

- El tiempo invertido en la búsqueda de una solución óptima es grande, pues debido a la naturaleza combinatorial del problema del ordenamiento óptimo, la técnica de optimización debe realizar una búsqueda exhaustiva en el espacio de soluciones, el cual a su vez, crece exponencialmente con el tamaño del problema. A pesar de esto, para aplicaciones como contingencias, corto circuito o donde sea necesario efectuar un gran número de flujos de carga; la metodología propuesta reduce el tiempo computacional, el número de operaciones, el espacio requerido en memoria y el error que se introduce en la respuesta. Así, es bastante viable y además beneficioso, invertir un cierto tiempo en encontrar el orden óptimo del sistema, pues con ello se logrará mejorar en gran medida la eficiencia de los procesos de cálculo donde interviene.

- A medida que el programa producto de la presente investigación se somete a problemas cada vez más grandes, las cualidades que lo identifican y otras características representativas mejoran ostensiblemente. Por ejemplo, tal cual se puede observar en la tabla 7.2, entre mayor sea el tamaño del sistema, el número de operaciones

que se debe realizar para llevar a cabo el proceso de solución disminuye en gran medida. Dentro de las características propias del programa, es de resaltar el perfeccionamiento del proceso de convergencia a medida que éste se aplica a problemas de gran envergadura. Este comportamiento queda claro en las figuras 7.4 y 7.16, donde se puede observar que para el sistema IEEE-14 el método encontró la mejor solución mucho antes de éste terminar; caso contrario al del IEEE-300, para el cual, la mejor solución fue alcanzada sólo uno cuantos ciclos antes de parar el proceso completamente.

Bibliografía

[1] Castro Carlos A. *"Tecnicas de esparsidade"*. Universidad de Campinas-UNICAMP. 2003.

[2] D. Sperandio. *"Cálculo Numérico"*. Pearson. 2003.

[3] National Physical Laboratory. *"Modern Computing Methods"*. Notes on Applied Sciencie. No. 16, Second Edition. London. 1962.

[4] S. D. Conte. *"Elementary Numerical Analysis"*. MacGraw-Hill. 1965.

[5] Aarts E., Korst J. and Michiels W.

[6] Garey M. R. and Johnson D. S. *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*. Freeman. San Francisco. 1979.

[7] Ausiello G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A. and Protasi M. *"Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties"*. Springer. Berlin. 1999.

[8] Aarts E. H. L. and Lenstra J. K. *"Local Search in Combinatorial Optimization, Princeton University Press"*. Princeton, NJ. 2003.

[9] Kirkpatrick S., Gelatt Jr C. D. and Vecchi M. P. *"Optimization by simulated annealing"*. Science Vol 220. pp 671-680. 1983.

[10] Černý V. *"Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm"*. J. Optim. Theory Appl. Vol. 45. pp 41-51. 1985

[11] Metropolis M., Rosenbluth A., Rosenbluth M., Teller A. and Teller E. *"Equation of state calculations by fast computing machines"*. J. Chem. Phys. Vol. 21. pp 1087-1092. 1953.

[12] Binder K. *"Monte Carlo Methods in Statistical Physics"*. Springer. Berlin. 1978.

[13] Lin S. *Computer solutions of the traveling salesman problem.* BellSyst.Tech. J. Vol. 44. pp 2245-2269. 1965.

[14] Lin S. and Kernighan B. W. *An effective heuristic algorithm for the traveling salesman problem*. Oper. Res. Vol. 21. pp 498-516. 1973.

[15] van Laarhoven P. J. M. and Aarts E. H. L. *Simulated Annealing: Theory and Applications*. Reidel. Dordrecht. 1987.

[16] Collins N. E., Eglese R. W. and Golden B. L. *Simulated annealing: An annotated bibliography*. Am. J. Math. Manage. Sci. 8. pp 209-307. 1988.

[17] Romeo F. and Sangiovanni-Vincentelli A. *A theoretical framework for simulated annealing*. Algorithmica. Vol 6. pp 302-345. 1991.

[18] Gallego R., Escobar A. y Romero R.*"Técnicas de Optimización Combinatorial"*. Universidad Tecnológica de Pereira. Pereira. 2006.

[19] Grainger J y Stevenson Jr. W. *"Análisis de Sistemas de Potencia"*. McGraw-Hill. México. 1996.

[20] Wang Y.Q. and Gooi H.V. *"New ordering methods of sparcing matrix inversion via diagonalization"*. IEEE Transactions on Power Systems. Vol. 12, No. 3. Aug. 1997.