UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

AMBIENTACION INTERACTIVA BASADA EN OBJETOS PARA LA SIMULACION DE SISTEMAS DE POTENCIA

T E S I S

QUE PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERIA ELECTRICA CON ESPECIALIDAD EN SISTEMAS ELECTRICOS DE POTENCIA

P R E S E N T A:

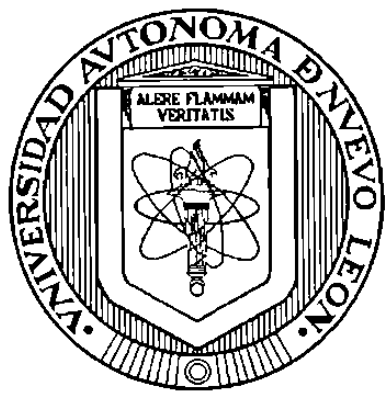
JOSE ALBERTO AVALOS GONZALEZ

CD. UNIVERSITARIA ENERO DE 1994

AMBIENTACION INTERACTIVA BASADA EN OBJETOS PARA LA SIMULACION DE SISTEMAS DE POTENCIA

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA



AMBIENTACION INTERACTIVA BASADA EN OBJETOS PARA LA SIMULACION DE SISTEMAS DE POTENCIA

TESIS

QUE PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS
EN INGENIERIA ELECTRICA CON ESPECIALIDAD EN SISTEMAS ELECTRICOS DE POTENCIA

PRESENTA

JOSE ALBERTO AVALOS GONZALEZ

MONTERREY, N L ENERO, 1994

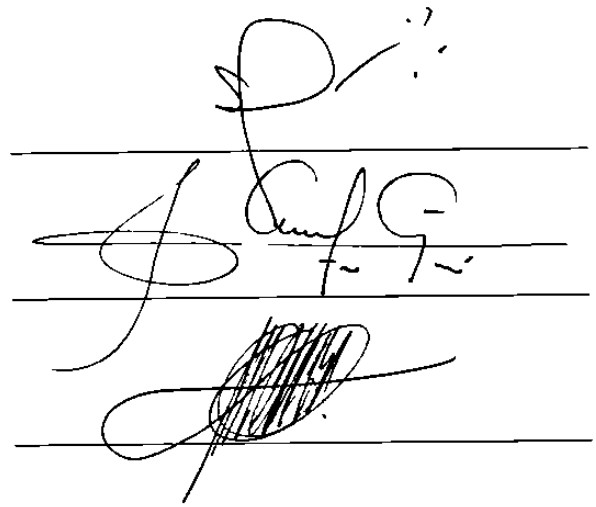
AMBIENTACION INTERACTIVA BASADA EN OBJETOS PARA LA SIMULACION DE SISTEMAS DE POTENCIA

Los miembros del Comité aprueban la Tesis
de Maestría de Jose Alberto Avalos Gonzalez

Dr Salvador Acha Daza
Asesor

Dr Florencio Aboytes Garcia

Dr Oscar Leonel Chacón Mondragon



Dedicado a mi madre:

A quien le debo todo y solo acierto a decirle "Gracias".

AGRADECIMIENTOS

Agradezco profundamente a mis hermanos que tomaron la brecha para que yo siguiera el camino, gracias.

A mi hermano Oscar cuyos logros personales me alientan a seguir adelante.

A mis sobrinos que me hacen sentir más viejo al verlos crecer pero que alegran mi existencia.

A mi amiga Mariana por su valiosa amistad.

Además, agradezco

- Al Doctor Salvador Acha Daza de forma muy especial por su excelente asesoramiento, el cual requirió de un gran esfuerzo de su parte. Para él mi admiración y respeto.

- Al Dr. Oscar Chacón Mondragón su amistad y ayuda en el transcurso de mi desarrollo profesional, así como el permitirme colaborar con él en algunos de sus proyectos.

- Al Dr. Florencio Aboytes García por sus importantes sugerencias.

- Al Doctor Isaías Elizarraras Alcaraz un ejemplo a seguir de trabajo y firmes convicciones.

- Agradezco a la Escuela de Ingeniería Eléctrica de la Universidad Michoacana de San Nicolás de Hidalgo, su fuerte apoyo para la realización de mis estudios.

- Al Consejo Nacional de Ciencia y Tecnología por su apoyo económico.

RESUMEN

AMBIENTACION INTERACTIVA BASADA EN OBJETOS PARA LA SIMULACION DE SISTEMAS DE POTENCIA

Publicación No._____

José Alberto Avalos González, M.C. en Ingeniería Eléctrica

Universidad Autónoma de Nuevo León, 1994

Profesor Asesor: Dr. Salvador Acha Daza

El desarrollo de simuladores digitales como herramientas de análisis en la planeación, la operación y el control de los Sistemas Eléctricos de Potencia (SEP), ha permitido un crecimiento y desarrollo sostenido de los sistemas eléctricos al aumentar la capacidad de análisis y el conocimiento de su comportamiento por parte del personal, en sus diferentes áreas.

Este trabajo presenta el desarrollo de un "*Simulador Digital Interactivo*" para el análisis de SEP, planteando como objetivo principal el crear un ambiente amigable que permita al usuario una interacción plena con los algoritmos, los modelos de los elementos representados y los parámetros del sistema. El simulador presenta diferentes herramientas de análisis, así como la posibilidad de manejar distintos dispositivos de hardware computacional como entrada/salida dentro del mismo ambiente.

El simulador tiene un desarrollo modular de manera de permitir la inserción de nuevos módulos en forma sencilla y directa, y éstos a su vez pueden interactuar con los módulos ya existentes si la aplicación así lo requiere.

Se emplearon herramientas computacionales modernas las cuales por su flexibilidad de manejo permiten realizar una programación eficiente y de fácil mantenimiento, así como la posible reutilización del código en desarrollos posteriores.

Se explican las bases para el manejo de éste tipo de programación, basada en objetos, generando la estructura modular del simulador para permitir la adición de nuevas aplicaciones. La aplicación del ambiente Windows y el sistema operativo DOS en PC compatible permite un desarrollo versátil y adecuado para el análisis, entrenamiento e investigación de los sistemas eléctricos de potencia.

Mediante el uso de herramientas de análisis de los SEP en estado estable, como estudio de flujos de secuencia positiva y el análisis de cargabilidad de enlaces, se ilustra la generación de los módulos base con los que cuenta el simulador interactivo.

Se pone a disposición del lector, el simulador digital interactivo ya sea como una herramienta de apoyo para el análisis de SEP o bien para valorar y verificar las perspectivas de desarrollo de este tipo de ambientación. El simulador puede obtenerse contactando al autor o bien en el Programa Doctoral en Ingeniería Eléctrica de la Universidad Autónoma de Nuevo León, institución en la cual se realizó este desarrollo.

INDICE

INDICE (continuación)

INDICE (continuación)

INDICE (continuación)

LISTA DE FIGURAS

CAPITULO 2

CAPITULO 3

CAPITULO 4

LISTA DE FIGURAS (continuación)

CAPITULO 5

LISTA DE FIGURAS (continuación)

CAPITULO 1
INTRODUCCION

En este capítulo se presentan diversos aspectos generales a considerar en el desarrollo de un *Simulador Digital Interactivo* para el análisis de SEP. Se describen las diferencias sustanciales entre *simulación en línea y fuera de línea*, el concepto de *ambiente interactivo* y su evolución en los últimos años, también se presentan diversos avances en hardware y software así como las tendencias en su aplicación. Se incluye una descripción de la filosofía de los llamados *Sistemas Abiertos*, se mencionan conceptos sobre *Procesamiento en Paralelo* y el posible uso de *Transputers*, por último se describe el desarrollo global del trabajo desglosado en sus diferentes capítulos.

1.1 SIMULACION DIGITAL DE SISTEMAS ELECTRICOS DE POTENCIA

En la planeación, operación y control de un sistema eléctrico se realizan una gran cantidad de estudios tales como: flujos, fallas, estabilidad, pronóstico de demanda, despacho económico, etc.; cada uno de los cuales requiere el desarrollo de algoritmos numéricos que permitan obtener soluciones rápidas y confiables. En la actualidad existe una gran cantidad de herramientas computacionales implementadas, algunas de las cuales se han logrado entrelazar formando importantes simuladores digitales para el análisis de los SEP. La poca interacción real que presentan este tipo de simuladores en la selección externa de cada uno de sus módulos y periféricos, la tediosa tarea de cambiar datos para imponer una nueva condición de operación, así como la costosa plataforma de desarrollo que restringía su aplicación a cierto número de personal, limitaron en gran medida los beneficios reales de estos simuladores.

Una importante plataforma de desarrollo para simuladores digitales son las estaciones de trabajo, las cuales cuentan con capacidad gráfica, memoria y velocidad de procesamiento importantes; desafortunadamente el alto costo de estos equipos, y de su entorno de software y dispositivos periféricos, limita su empleo pleno en diversas

aplicaciones. El avance tecnológico logrado con el surgimiento de las computadoras personales (PC) en los años 80's y su evolución hasta nuestros días, con características importantes como: gran capacidad de memoria y almacenamiento, elevada velocidad de procesamiento y alta resolución para el manejo gráfico, además de la accesibilidad de éste equipo por su bajo costo, ha dado origen a la generación de una infinidad de soporte en software y dispositivos periféricos para el desarrollo de aplicaciones en diversas áreas, convirtiendo así a estas computadoras en otra importante plataforma de desarrollo. Las características descritas de las computadoras personales llevaron a seleccionarlas como la plataforma más versátil y aplicable para el desarrollo del simulador digital que se presenta en este trabajo.

1.2 SIMULACION EN LINEA Y FUERA DE LINEA

La implementación de un simulador digital para el análisis de SEP puede realizarse con diversos objetivos, tales como: apoyo en la enseñanza e investigación, entrenamiento de operadores y personal involucrado en áreas específicas de análisis en la planeación y el control del SEP, o bién como una herramienta de apoyo directo en los centros de control. Cada uno de los distintos objetivos planteados involucra diferentes características de implementación del simulador digital, tanto en las herramientas de análisis y modelado, como en el software y hardware a utilizar, los cuales están en función del tamaño del sistema que se desea analizar y de la velocidad de respuesta deseada. La velocidad de respuesta requerida da lugar a dos grandes tipos de simuladores digitales: simulador "*en línea*" y simulador "*fuera de línea*", los cuales delimitan el grado de detalle en el modelado de elementos (completos o aproximados) y el tipo de algoritmos que se implementarán para el análisis, así como el hardware, software y dispositivos entrada/salida a utilizar. A continuación se describen ambos tipos de simuladores, así como sus requerimientos y restricciones más importantes.

1.2.1 SIMULACION EN LINEA

En los centros de control modernos se requiere la "*operación en línea*" de un

simulador digital lo cual implica la posibilidad de realizar distintos tipos de análisis del SEP, basados en condiciones reales de operación, obtenidas mediante a mediciones físicas en diferentes puntos del sistema. Para llevar acabo lo anterior se hace necesario la actualización constante de la topología de la red a través de dispositivos de medición (que incluye sensores, transductores y sistemas de comunicación) y el procesamiento de información, lo cual requiere de un equipo de cómputo (hardware) lo suficientemente rápido, así como de un sistema operativo capaz de manejar de manera eficiente la información y los dispositivos periféricos (impresores, graficadores, pantallas gráficas, etc.), los cuales a su vez deben de cubrir los requisitos de velocidad de respuesta y resolución necesarios para la presentación de información y resultados.

La probable falla en los sistemas de medición, comunicación o en el equipo de adquisición de datos (SCADA por sus siglas en inglés), hace necesario validar la información y tratar de reconstruir fielmente la topología real del sistema, sin la cual no sería posible realizar estudios posteriores. En la Figura 1.1 se muestra un esquema de validación de información procesada por medio de un estimador de estado, el cual recibe la configuración del sistema encontrada por el procesador de topología.

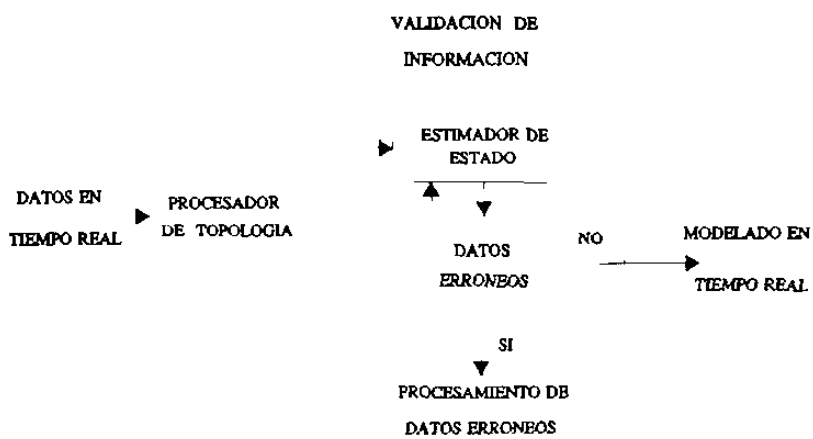


Fig. 1.1 *Esquema de validación de mediciones para modelado en tiempo real.*

El tiempo empleado en el proceso de simulación está en función del tamaño del sistema, del tipo de estudio, de los algoritmos implementados, así como del equipo computacional utilizado. A continuación se presentan datos de la velocidad de procesamiento estimado para diversos tipos de análisis en SEP[11]:

Estudio	Número de Operaciones estimadas
◦ Control Automático de Generación	0.1 MIPS*
◦ Control supervisorio y adquisición de datos	1.2 MIPS
◦ Asignación de unidades	2.0 MIPS
◦ Análisis de redes	5.0 MIPS

Los requerimientos computacionales observados llevan al empleo de Estaciones de Trabajo o Computadoras Centrales, así como al uso de sistemas operativos como Unix, VxWorks y VRTX32[10], diseñados especialmente para operación en tiempo real. Las restricciones sobre desplegados rápidos, manejo de memoria y velocidad de procesamiento en las computadoras personales aún restringe su empleo en aplicaciones con uso intensivo de operaciones numéricas; aunque con la evolución observada en memoria disponible, velocidad de procesamiento, integración de procesadores matemáticos, etc.; pueden ser consideradas como una opción para sistemas de tamaño reducido. La capacidad de procesadores múltiples por parte del computador central puede resultar bastante provechosa en la ejecución de estudios en paralelo. Los algoritmos numéricos a utilizar normalmente serán aproximados, con el fin de aumentar su rapidez de respuesta, así mismo el modelo de los elementos utilizado debe ser lo más simplificado posible sin deteriorar la solución.

---

* *Millones de instrucciones por segundo para una red de 500 nodos*

1.2.2 SIMULACION FUERA DE LINEA

La operación "*fuera de línea*" de un simulador digital permite mayor flexibilidad y menos restricciones computacionales en su implementación, ya que no hay requerimientos estrictos en tiempo de ejecución. Los requerimientos básicos son: capacidad de memoria acorde a las dimensiones del sistema a analizar y un tiempo de respuesta razonable que permita simular diversas condiciones de operación del sistema sin perder continuidad en la interpretación de resultados.

En este tipo de simuladores digitales se implementan bases de datos previamente diseñadas, las cuales se van modificando según las distintas condiciones de operación que se deseen analizar. Desaparece así toda posible incertidumbre en cuanto a la configuración ó topología del sistema, que se tiene con la operación en línea. Permite realizar estudios de planeación a corto, mediano y largo plazo, así como formular una infinidad de estudios que involucran flujos, fallas, estabilidad, armónicos, despacho económico, calendarización de combustibles, así como otros más generales para el manejo de recursos energéticos. Una característica importante es enfatizar la capacidad de análisis que proporcionan los simuladores digitales "*fuera de línea*" al permitir realizar estudios bajo condiciones que no es factible llevar a cabo "*en línea*", y que requieren un análisis profundo y detallado, así como el empleo de algoritmos completos y el modelado a un mayor grado de detalle de los elementos de los SEP, logrando de esta manera una mejor representación del comportamiento del sistema.

Los simuladores digitales para el entrenamiento de operadores reúnen características tanto de "*operación en línea*" como de "*operación fuera de línea*", ya que resulta necesario enfrentar al operador del sistema a condiciones similares a la realidad, por tanto, la simulación de contingencias, cambios en el sistema y la respuesta a las acciones tomadas por el propio operador deben realizarse con una rapidez aproximada a la real. El tipo de eventos simulados deben ser tan comunes o peculiares que permitan sensibilizar a los operadores en los puntos claves de operación del SEP. Ambos tipos

de simuladores aún contando con objetivos distintos, exigen el manejo de algoritmos eficientes, así como el optimizar el manejo de información.

1.3 NECESIDAD DE TECNICAS COMPUTACIONALES EFICIENTES

La gran cantidad de información requerida por los diversos tipos de análisis, el grado de detalle y exactitud deseados en los modelos matemáticos, así como las operaciones numéricas a realizar, requiere de técnicas computacionales eficientes que aprovechen las características del SEP, de sus elementos y de los propios algoritmos numéricos a emplear. La optimización de memoria puede lograrse utilizando técnicas de empaquetado de matrices dispersas, algoritmos de factorización matricial (triangularización, bifactorización, etc.), ordenamiento óptimo, etc., los cuales permiten preservar la dispersidad de las matrices. La simplificación del modelado de los elementos del SEP de acuerdo a su grado de participación en cada tipo de estudio (detallando aquellos con mayor influencia y simplificando al máximo los que por su respuesta no intervienen en forma significativa en el estudio), el empleo de algoritmos que permitan aprovechar los desacoplamientos posibles entre grupos de variables del SEP, etc., reducen considerablemente el número de operaciones numéricas a realizar incrementando la velocidad de ejecución de los algoritmos.

La implementación de métodos con estabilidad y robustez numérica para los algoritmos empleados garantizan una rápida y excelente solución bajo prácticamente cualquier condición de operación. Sin embargo, pueden presentarse condiciones ya sea de la red o del punto de operación analizado para los cuales existan problemas numéricos de solución, en estos casos es deseable implementar algoritmos adicionales alternos que proporcionen información sobre la posible localización del problema y criterios viables de solución, a este respecto las técnicas de programación lineal y no-lineal son opciones alternas importantes. El éxito de un simulador no estriba solamente en la eficiencia de los algoritmos numéricos implementados, sino también en la flexibilidad e interacción real que brinde al usuario.

1.4 CONCEPTO DE AMBIENTE INTERACTIVO Y SU EVOLUCION

El concepto de *ambiente interactivo* surge como una necesidad de "diálogo" entre la computadora y el usuario, diálogo que debe realizarse a través de una comunicación directa y natural. Proporcionarle al usuario un ambiente de trabajo amigable le permite realizar sus tareas de una manera más confortable, cuando las tareas llegan a involucrar tiempos muy prolongados resultan monótonas e incluso molestas. Una forma de lograr lo antes mencionado es presentar ante el usuario solamente la información necesaria y suficiente en cada una de las opciones del programa, las cuales por sí mismas deben guiarle a un desarrollo lógico de seguimiento del proceso, permitiéndole retornar en todo momento al menú principal o a pasos anteriores. El contar con ayuda local en cada etapa, así como el poder interactuar con el mayor número posible de parámetros o elementos manejados, permitirá lograr un verdadero ambiente interactivo.

En el ámbito de los SEP se han desarrollado importantes simuladores digitales interactivos, en la década de los 70's aún con las grandes restricciones en el equipo de cómputo de la época se realizaron varios simuladores importantes[4,5,6]. El impacto de las computadoras personales y estaciones de trabajo en los 80's brindó la oportunidad de crear herramientas con características interactivas más extensas y amigables al usuario, permitiendo por su bajo costo ponerlas al alcance de un mayor número de personal, incrementando simultáneamente su habilidad en la operación y el control de los SEP. Un gran avance se logró al implementar, dentro de varias universidades del mundo, laboratorios digitales para el desarrollo de algoritmos numéricos para análisis e investigación, mismos que servían de base para sus programas de estudio, formando parte posteriormente de simuladores industriales. Se crearon nuevas inquietudes y necesidades, mejorando por tanto la capacidad interactiva de los simuladores.

Las herramientas de software existentes hoy en día permiten generar simuladores interactivos con una gran versatilidad, moverse entre aplicaciones dentro de un mismo ambiente de trabajo, manejar una gran diversidad de dispositivos entrada/salida, etc.

Una de las características fundamentales requeridas en este tipo de ambientes es que el ambiente interactivo debe permitir seleccionar de una manera rápida y sencilla cada una de las herramientas habilitadas dentro del simulador, tales como: diferentes algoritmos de solución, desplegado gráfico o tabular de soluciones intermedias o de la solución final de los diversos algoritmos, ya sea por medio de monitores o bien a través de diversos dispositivos entrada/salida (impresoras, unidades de almacenamiento de información, etc.). Es deseable contar con el control pleno de cada una de las etapas de cálculo y su evolución, de tal manera que sea factible interactuar con el simulador en pasos intermedios del proceso o bien al término del mismo, permitiendo así modificar o ajustar parámetros y reflejar su efecto en forma inmediata.

Estos últimos puntos reflejan la flexibilidad que es deseable tener en un simulador interactivo, los cuales deben analizarse detenidamente en cuanto a factibilidad y conveniencia en su implementación, así como el soporte de los sistemas operativos y la arquitectura de los procesadores al desarrollo que realiza. La capacidad del lenguaje de programación FORTRAN para el cálculo numérico, aunado a su gran facilidad de manejo, lo han colocado como la herramienta más utilizada en el desarrollo de algoritmos para el análisis de los SEP. Por otro lado, su dificultad para el manejo de dispositivos entrada/salida, así como la poca flexibilidad para crear un ambiente interactivo requieren del empleo de herramientas externas para la ambientación de estos programas. Este aspecto resulta importante si se desea dar al simulador la facilidad de interactuar por medio de ambientación para realizar acciones tales como: detener la solución, mostrar resultados y continuar la solución ó cambiar parámetros y reiniciar el proceso de estudio. El detener la ejecución de un programa implica acciones como: parar la solución al término de la iteración actual, grabar los resultados que en ese instante se tienen, activar las opciones de resultados en la ambientación, analizar los mismos y posteriormente decidir si se continua el estudio o se cambia algún parámetro del sistema antes de continuar. Bajo estas circunstancias, para reanudar la ejecución del proceso puede aprovecharse como punto de arranque del algoritmo la condición existente en el momento que se detuvo la solución, ya que puede resultar impráctico

almacenar toda la información generada por el algoritmo antes de interrumpir el programa, información como: banderas, matrices factorizadas, etc.

Una forma de evitar estas restricciones sería el contar con módulos separados para presentar resultados intermedios, ejecutando éstos desde el programa de FORTRAN lo cual es razonable, pero existe un problema ya que el sistema operativo DOS restringe a solamente dos programas consecutivos, la posibilidad de que un programa internamente ejecute otro. Por tanto, al ejecutar el programa de ambientación y éste a su vez ejecutar el algoritmo, se trunca la posibilidad de que éste último pueda ejecutar un tercer programa de presentación de resultados. Incluso en ambientación Windows, que permite el manejo de multitareas, las cuales se refieren a correr módulos independientes (por lo menos en el sistema operativo DOS), resulta en una restricción importante. Por lo tanto, si se piensa en crear un simulador que seleccione diferentes tipos de estudios con manejo de ambientación separada, es necesario controlar en forma externa la comunicación entre algoritmos, aunque esto es posible lograrlo de una manera transparente al usuario es importante tenerlo en cuenta.

1.5 SOFTWARE Y HARDWARE PARA EL MANEJO DE AMBIENTES INTERACTIVOS

Existen diversas alternativas para el desarrollo de ambientes interactivos en simuladores digitales para análisis de los SEP. Lenguajes como *Smalltalk*, creado en 1973 con fines de generación de programas interactivos, ha llegado a ser una herramienta poderosa; actualmente éste lenguaje maneja completamente la *programación orientada a objetos*. Lenguajes como C++ ofrecen importantes recursos para el desarrollo de ambientación de programas, manejados a través de objetos, siendo la facilidad para emplear este lenguaje relativa ya que siempre se ha caracterizado por ser críptico y además deja bastante responsabilidad al programador en los errores que éste introduce involuntariamente. Visual Basic ha logrado revolucionar la flexibilidad en la generación de código directamente relacionado con cada ventana o con elementos de

control dentro del ambiente, a tal grado, que se ha desarrollado la versión Visual C++ de la misma compañía. Esta relación directa entre controles y código puede facilitar bastante la programación de estos ambientes, por tanto resultaría interesante valorarla. Existen otros lenguajes también propicios para ambientación de programas pero en esta investigación solamente se han analizado los que se consideran más populares y más viables para la aplicación a nivel industrial.

A nivel de hardware, el mayor crecimiento que se ha tenido a partir de la primera computadora personal, es el desarrollo paralelo de procesadores entre las compañías INTEL 8088 y Motorola 6800, que permitieron a las grandes compañías IBM y APPLE CO. popularizar estas computadoras, las cuales aunque con poca capacidad de memoria y lentitud en el procesamiento, resultó invaluable el beneficio global proporcionado. Posteriormente, la evolución de procesadores más rápidos como el 80286 por INTEL, el 68000 por Motorola y la creación de procesadores matemáticos, incrementaron la velocidad de ejecución de operaciones numéricas, capacidad de manejo de diversos dispositivos periféricos, etc.

La introducción de procesadores con manejo de palabras de 32 bits 80386 y 68020, incrementaron la velocidad de las computadoras personales y el manejo de memoria RAM desde 2 a 16 Mbytes y configuraciones superiores, así como el almacenamiento en unidades de disco entre 80,120,230 Mbytes, relojes de procesamiento en el rango de 16 - 66 Mhz, capacidad de video CGA, VGA, SuperVGA, etc., dieron una perspectiva clara de las grandes posibilidades de desarrollo de las computadoras personales.

1.6 Concepto de sistemas abiertos

La gran diversidad de compañías dedicadas a la fabricación de computadoras, compiladores, sistemas operativos, etc., cada una de ellas con su arquitectura y normas propias, ha originado serios problemas en el desarrollo generalizado de aplicaciones.

Cada compañía asegura un alto porcentaje de compatibilidad de su equipo, sistemas operativos o compiladores, pero la realidad es que la transportabilidad de programas en muchas ocasiones no ha resultado satisfactoria ni transparente. Este tipo de problemas afecta grandemente a la industria eléctrica, un ejemplo se tiene al adquirir una estación de trabajo de cierta marca, en la cual se han desarrollado aplicaciones de análisis de SEP por una compañía desarrolladora de software, si posteriormente otra compañía desarrolla un conjunto de algoritmos complementarios que se desean adquirir, nadie asegura que éstos trabajarán en la computadora disponible, incluso si ambas herramientas fueron desarrolladas bajo el mismo sistema operativo. Existe pues el problema de que hay varias compañías desarrollando versiones distintas de sistemas operativos como Unix, con normas y protocolos distintos, siendo el mismo problema que se tiene en las computadoras personales bajo el sistema operativo DOS y otros más.

Estos problemas han dado lugar a intentar la creación de "*Sistemas Abiertos*" en los cuales se busca estandarizar el desarrollo de Hardware y Software que permita una migración confiable de aplicaciones entre diversas computadoras y sistemas operativos. Se busca también el poder transportar código fuente entre diferentes compiladores de un mismo lenguaje. En la industria eléctrica se intenta normalizar entre las diferentes compañías desarrolladoras de software para análisis de los SEP, aspectos tales como entrada y salida de información de tal manera que puedan entrelazarse en forma directa aplicaciones realizadas por compañías distintas[25].

1.7 PROCESAMIENTO EN PARALELO Y TRANSPUTERS

La velocidad de ejecución lograda con los nuevos microprocesadores de 32 bits ha sido satisfactoria pero no suficiente, principalmente en el desarrollo de algoritmos con uso intensivo de operaciones numéricas. La posibilidad de instalar 2 ó más procesadores en una misma computadora, repartiendo tareas computacionales entre ellos, brinda una gran posibilidad para el desarrollo tanto de algoritmos numéricos rápidos, como para el manejo de tareas múltiples para incrementar la interacción en ambientación de los

simuladores digitales entre los métodos de análisis y los diferentes dispositivos entrada/salida.

La inclusión de procesadores múltiples da lugar a diversas modificaciones tanto en la programación de los algoritmos para la repartición de tareas entre los procesadores, como en el sistema operativo y compiladores a utilizar. Existen diversas opciones para incrementar la velocidad de solución de algoritmos numéricos, tales como: el empleo de procesadores múltiples en una misma máquina, la interconexión de varias computadoras, etc. Un método alternativo importante consiste en el empleo de "**TRANSPUTERS**"[26], equipo que consta de un microprocesador de 32 bits, unidad de memoria RAM estática y diversas líneas bi-direccionales de comunicación para ser interconectado con otros elementos o dispositivos similares. Las reglas de procesamiento en paralelo para la conexión de varios procesadores, así como los algoritmos que ahí se emplean son trasladables a la computadora personal central que manejará y controlará los diferentes dispositivos instalados. Una ventaja adicional de los transputers es la facilidad con la que se podría ir incrementando el número de elementos interconectados conforme crezcan las necesidades de análisis, tamaño del sistema o velocidad de procesamiento deseada.

1.8 CONTENIDO DEL PRESENTE TRABAJO

En el presente trabajo se ha desarrollado un "*Simulador Digital Interactivo*" para el análisis de los SEP, se han incluido las herramientas de análisis de Cargabilidad de Líneas de Transmisión y Flujos de Carga, el cual incluye tres métodos clásicos: Newton-Raphson Formal, Desacoplado Rápido y Gauss-Seidel. La ambientación se desarrolló bajo ambiente Windows® empleando el lenguaje de programación C++, utilizando una computadora personal PC-486 con reloj de 33 MHz, disco duro de 110 MBytes y 4 MBytes de memoria RAM.

En el Capítulo 2 se presenta una descripción de la programación por objetos en el lenguaje C++, así como el diseño de ambientación de programas en ambiente Windows® mediante el empleo de la biblioteca de funciones (OWL por sus siglas en inglés) de la compañía BORLAND.

En el Capítulo 3 se describe la forma en que interactua la ambientación desarrollada en modo gráfico con los algoritmos desarrollados en modo texto. Se presenta también la formulación de los algoritmos de flujos de carga implementados.

El Capítulo 4 muestra una descripción del ambiente e interacción del simulador así como su manejo, utilizando el módulo de flujos de carga. Por medio de corridas se demuestra la flexibilidad y las bondades de la herramienta desarrollada. Se plantean alternativas y desventajas al interactuar lenguajes de alto nivel como FORTRAN, Quick BASIC™ y C++, en los cuales se han implementado diversos algoritmos numéricos, así como al combinar los ambientes Windows y DOS, en el desarrollo del simulador.

En el Capítulo 5 se presenta el módulo de análisis de cargabilidad incluido en el Simulador Digital, así como la interacción lograda en el mismo. Se describe la integración de los módulos de cargabilidad y flujos de carga implementados en como un grupo de aplicaciones hacia Windows y las ventajas que esto produce.

Por último en el Capítulo 6 se presenta las conclusiones y aportaciones de esta investigación, así como un conjunto de recomendaciones respecto a extensiones de este trabajo y las alternativas de desarrollo del mismo.

CAPITULO 2
PROGRAMACION EN AMBIENTE WINDOWS

En el presente capítulo, se presentan los conceptos fundamentales de la *programación orientada a objetos* (POO), sus características y ventajas de desarrollo. Se presenta la forma de definir *clases* y derivar a partir de ellas *objetos*, que son la base de la POO. Se describe el desarrollo de ambientación Windows® en el lenguaje C++ empleando las clases definidas por la compañía Borland, mostrando las diferentes clases que se requiere formar para crear ventanas, diálogos y controles, así como la forma de definir funciones para controlar la respuesta a los mensajes enviados al seleccionar menús, botones de diálogos y controles en general.

Con la finalidad de presentar las herramientas fundamentales para la creación de programas en ambiente Windows, a partir de ejemplos ilustrativos, se presupone un conocimiento medio de programación en lenguaje C y nociones sobre el correspondiente C++.

2.1 PROGRAMACION ORIENTADA A OBJETOS

El crecimiento en el empleo de computadoras en diferentes medios de la industria, el comercio, las instituciones educativas, las dependencias gubernamentales, etc., dio origen al desarrollo de una infinidad de aplicaciones que con el tiempo fueron creciendo, planteando los usuarios nuevas inquietudes y necesidades en su implementación, a tal grado que las herramientas de software resultaron insuficientes en la programación a gran escala. Fue necesario cambiar totalmente la forma y diseño de programas a través de la programación estructurada[14], la cual obligó al desarrollo de código más eficiente, con posibilidad de generar aplicaciones de mayor tamaño.

Posteriormente, los especialistas en el desarrollo de software se percataron de la gran cantidad de código que se rehacía al generar una aplicación nueva, o al modificar

alguna ya existente con características implementadas anteriormente en otros programas. También se observó la problemática de dar mantenimiento y hacer crecer las aplicaciones, encontrando necesario, para poder realizar modificaciones, conocer tanto las variables utilizadas como la forma en que habían sido implementadas cada una de las funciones. Esta problemática dió origen a la llamada *programación orientada a objetos*, cuyo propósito principal es generar código eficiente, reutilizable, fácil de mantener e incrementar sus características de funcionamiento, creando entes independientes denominados "*objetos*".

2.2 OBJETOS

Un objeto puede verse como una variable derivada de un tipo especial de dato, el cual internamente engloba diversos tipos de datos, así como las funciones que los procesan para la realización de diferentes tareas. Como ejemplo podría definirse un tipo de dato denominado "*matriz*", el cual internamente manejará datos como: posición de elementos en renglón y columna, su valor numérico, índices, etc., además, también puede contener diversas funciones que definen operaciones matriciales como: multiplicación, inversión, factorización, cálculo del determinante, etc. A partir de este tipo de dato llamado "*matriz*" puede derivarse otras variables denominadas "*objetos*", que contienen la información de una matriz y pueden realizar las operaciones matriciales que se deseen; llamando simplemente a la función por medio del objeto respectivo. Pueden definirse también objetos específicos para manipular la información de una red eléctrica, por ejemplo: que obtenga la matriz Y_{nodal}, o el Jacobiano, etc., a partir de datos de interconexión de la red y sus elementos, así como voltajes y potencias nodales. La intención primordial de la POO es crear objetos independientes en código y de los datos de cualquier otra parte del programa, así como que éstos representen elementos físicos de fácil comprensión y utilización.

2.2.1 CLASES

Los tipos de datos a partir de los cuales se derivan los objetos se denominan

"*clases*". Las clases básicamente pueden verse como estructuras de datos en las que se definen algunas variables de diferentes tipos. La diferencia primordial entre *estructuras* y *clases* consiste en que en una clase pueden definirse, además de variables, funciones o prototipos de funciones como elementos internos de la clase, por ejemplo:

```
struct Linea
{
  int x1, x2, y1, y2,
};
Linea A, B;
void DibujaLinea(Linea   B)
{
   .
   .
}

main()
{
  A.x1 = 20;
  A.x2 = 40;
  A.y1 = 15;
  A.y2 = 15;
  DibujaLinea(A);

  return(0);
}
```

```
class Linea
{
  int x1, x2, y1, y2,
  int DibujaLinea();
};
Linea A, B;

main()
{
  A.x1 = 20;
  A.x2 = 40;
  A.y1 = 15;
  A.y2 = 15;
  A.DibujaLinea();
  B.x1 = 40;
  B.x2 = 20;
  B.y1 = 40;
  B.y2 = 40;
  B.DibujaLinea();
  return(0);
}
```

En el ejemplo se observa la diferencia clara en la declaración y el manejo de las *clases*, mientras que al manejar estructuras éstas son pasadas como parámetros a una función. Al emplear *clases* las variables y funciones trabajan en un mismo entorno, teniendo acceso directo las funciones a las variables de la clase. Esto es equivalente a tener variables globales dentro de la clase, y utilizar variables y funciones locales para el programa en su conjunto. Cuando se requiere definir más de un objeto, derivado de una misma clase, es fácil confundir lo que sucede cuando se llama una función de la clase, ¿sobre qué datos va a operar ésta?. Cuando se crea un objeto derivado de una clase determinada, sucede lo mismo que al crear una variable tipo estructura; se separa memoria para los datos de la clase asignados al objeto definido. Al generar un objeto nuevo se le asigna otro bloque de memoria, distinto al anterior, direccionando cada bloque con el nombre dado al objeto determinado; así al llamar una función de la clase

por medio de un *objeto* B, ésta operará con los datos asignados al *objeto* B y al llamar la función por medio del *objeto* A, ésta tomará los datos correspondientes al *objeto* A.

La capacidad de las clases para manejar en un mismo entorno datos y funciones, así como la posibilidad de reutilizar código al crear objetos independientes, permitiendo incrementar o variar sus características de funcionamiento, tiene lugar debido a propiedades importantes inherentes a las clases. Estas propiedades son:

- Encapsulado
- Herencia
- Polimorfismo

las cuales se describen a continuación.

Encapsulado: Esta propiedad es la que le permite a las clases conjuntar datos y funciones en un mismo entorno. Permite acceso directo a los datos de la clase a las diferentes funciones definidas en la misma.

Herencia: Proporciona una característica muy importante que permite definir clases derivadas de otras (denominadas clases base), heredando las propiedades de la clase base e incrementando sus datos y características de funcionamiento, permitiendo reutilizar el código ya existente.

Polimorfismo: Esta propiedad está ligada con la propiedad de herencia, permite que una función definida en la clase base sea creada nuevamente con el mismo nombre en la clase derivada, ya sea cambiando completamente la definición de la función, o bien llamando a la función original para su ejecución y posteriormente adicionar nuevas características de funcionamiento. En esta propiedad puede verse más claro, la gran posibilidad de reutilización del código ya existente.

Dentro de una clase existirán algunos datos y funciones que son para manejo interno, de tal manera que el permitir su acceso a otro ámbito del programa puede causar problemas e incluso generar resultados erróneos en el proceso. También es lógico pensar que se necesita el acceso a algunos datos y funciones sin el cual estas clases no tendrían ninguna utilidad práctica. Por estos motivos, a los elementos dentro de una clase, datos y funciones, se les ha dado una clasificación diferente:

private: Este tipo de elementos sólo es accesado por las funciones propias de la clase y por un tipo especial de funciones definidas en las clases derivadas llamadas *"funciones amigas"*.

public: Este tipo de elementos puede ser accesado desde cualquier ámbito del programa.

protected: Este tipo de elementos puede ser accesado, tanto por las funciones propias de la clase como por las funciones de las clases derivadas.

En resumen, se ha descrito la definición de clase como un tipo de dato especial que incluye, particularmente, funciones para manipular los datos definidos en la clase. Estos datos y funciones son creados en un entorno local que permite controlar el acceso a los mismos en las diferentes partes del programa. La complejidad, así como la generación óptima o no del código de un objeto quedan integradas en forma independiente para su empleo en desarrollos posteriores, permitiendo un mantenimiento y crecimiento de programas relativamente sencillo.

2.3 AMBIENTACION

Hasta hace poco el desarrollo de ambientación de cualquier programa requería implementar una gran cantidad de código. Se creaban funciones para generar ventanas principales, así como ventanas de diálogo para introducir datos o seleccionar opciones.

Se diseñaban elementos de control para cada ventana, menú, botón, línea para entrar datos, etc., se controlaba el movimiento del ratón, sus selecciones, la captura y respuesta a eventos; en fin era necesario realizar una gran cantidad de funciones para mantener el control completo de la ambientación. Con el surgimiento y desarrollo de la POO, se ha generado una gran cantidad de *clases* que permiten realizar de manera sencilla la mayor parte de las tareas requeridas en el manejo de ambientación; Windows principalmente. La posibilidad de aprovechar e incrementar las características de *clases base*, al definir nuevas *clases derivadas*, permite enfocar la programación exclusivamente a desarrollar una plena interacción de las opciones del ambiente con el usuario colocando menús, activándolos ó inhibiendo su acción. Es posible generar diálogos con sus diferentes controles y funciones de respuesta a los mismos, todo a través de la manipulación de las *clases base* ya existentes, principalmente para la interacción con parámetros y opciones del programa.

2.3.1 CLASES PRINCIPALES

El desarrollo de un programa hacia Windows se inicia generando una nueva clase, a partir de la clase principal "**TApplication**", la cual realiza las funciones primordiales de la aplicación como: inicializar instancias o procesos, construir e inicializar la ventana principal, así como generar un ciclo infinito de espera de respuesta por parte del usuario, que a su vez el ciclo procesa y envía a las diferentes instancias de la aplicación. Windows envía cuatro diferentes parámetros al ejecutar una aplicación; los cuales se incluyen para ser manejados dentro del programa como datos miembro de la clase derivada de **TApplication**. Los parámetros y su objetivo son descritos a continuación:

hInstance: Manejador del proceso o instancia de la aplicación que en ese momento se está ejecutando.

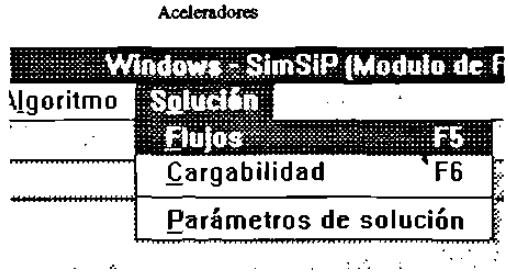
hPrevInstance: Manejador del proceso previo al actual en la misma aplicación. Este será cero si se está en el proceso inicial.

nCmdLine: Puntero a una cadena, la cual contiene los comandos en línea de la aplicación. Por ejemplo, "Editor.exe flujos.dat /m", indica que al programa "*editor*" se le pasan los parámetros "flujos.dat" y "/m".

hCmdShow: Contiene el modo de desplegado inicial de la aplicación, el cual será utilizado al llamar la función **Show** para mostrar la ventana principal.

Por ejemplo la definición de la nueva clase **TFlujos**, derivada de **TApplication**, se realiza de la forma siguiente:

```
class TFlujos : public TApplication
{
  TFlujos(    LPSTR Aname, HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
              nCmdShow): TApplication(Aname, hInstance, hPrevInstance, lpCmdLine, nCmdShow){};
  void InitInstance()
  {
       TApplication::InitInstance();
       HAccTable =
       LoadAccelerators(hInstance,    "Flujos");
  };
  virtual void InitMainWindow();
};
```



La nueva clase **TFlujos** derivada de **TApplication** hereda todas las propiedades de la clase base. Llamando al constructor de la clase base, incluso redefine la función **InitInstance** empleando la propiedad de polimorfismo, la cual como primera acción ejecuta la función original **TApplication::InitInstance()**, adicionando posteriormente una tabla de aceleradores que define las diferentes teclas rápidas (F5, F6, Alt-S, etc.) manejadas en el programa. El objetivo de la función **InitInstance** es arrancar los procesos de aplicación en Windows, inicializa la ventana principal al llamar la función **InitMainWindow**, la cual creará y mostrará la ventana con el tamaño y apariencia asignados. Esta ventana se crea generando una clase derivada de **TWindow** la cual se analizará a continuación.

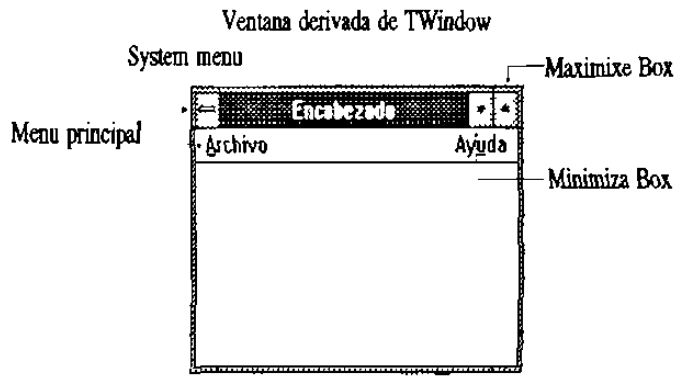
La ventana principal, así como cualquier otra ventana que se desee emplear en la aplicación, son generadas por medio de la clase **TWindow**, a partir de la cual se declara una clase derivada donde se definen las características de la ventana, tales como: estilo, posición, tamaño, colores de fondo, menús, etc. Se definen controles si se desea incluirlos, así como funciones de respuesta, tanto de los comandos de menús como de los controles incluidos en la ventana. A continuación se presenta la declaración de una clase derivada de **TWindow**, a través de la cual se explicará cada una de las funciones requeridas por la clase derivada y su objetivo.

```
class inicio : public TWindow
{ public:

  HBRUSH    hBKBrush;
  HBITMAP   hBKBrushBmp;
  HMENU     menu;
  PTStatic  Mensajes;
  TCambiosCom    GeneraStruct;
  TEjecutaStruct     EjecutaStruct;

  inicio(PTWindowsObject  AParent, LPSTR
ATitle);
  ~inicio();
  virtual BOOL CanClose();

  virtual void GetWindowsClass(
  WNDCLASS&  WndClass);
  virtual void Paint(HDC  PaintDC, PAINTSTRUCT&);
  virtual void CMGeneradores(RTMessage  Msg)
             = [CM_FIRST  + CM_Generadores];
  virtual void CMCambiosCorr(RTMessage  Msg)
             = [CM_FIRST  + CM_CambiosCorr  ];
};
```



La clase "*inicio*" cuenta originalmente con un constructor definido por la función **inicio(PTWindowsObject AParent, LPSTR ATitle)**, los parámetros que se pasan a esta función son **AParent=NULL** si se está creando la ventana principal, **AParent=this** si es cualquier otra ventana, el parámetro **ATitle** normalmente se utiliza para definir el título o encabezado de la ventana. En el constructor se definen los atributos de la ventana como: estilo, tamaño, etc., se le asigna el menú de opciones previamente generado por medio de la utilería *Resource Workshop*. En caso de que la ventana contenga controles,

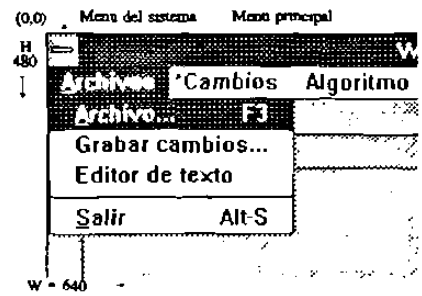
serán creados en esta función constructora, además se separará memoria para las estructuras empleadas en el manejo de los controles. Finalmente, una posible forma de declarar el constructor sería la siguiente:

```
inicio::inicio(PTWindowsObject  AParent, LPSTR ATitle): TWindow(AParent, ATitle),
{
  Attr.Stile  = WS_OVERLAPPED  | WS_SYSMENU  | WS_MINIMIZEBOX;
  Attr.X  = 0;
  Attr.Y  = 0;
  Attr.H  = 480;
  Attr.W  = 640;

  / Asigna el menu "Menu Principal", definido en el resource
  AssigMenu("Menu Principal");

  memset(&GeneraStruct,  0x0, sizeof GeneraStruct);
  memset(&EjecutaStruct,  0x0, sizeof EjecutaStruct);

  mensajes  = new TStatic(this,  -1, "", 8, 415, 306, 16, NULL );
}
```



La estructura "*Attr*" definida en la clase **TWindow**, es donde se asignan el tamaño y estilo deseados de la ventana. Por medio de la función **AssigMenu** se coloca el menú, previamente diseñado. La función **memset** inicializa en ceros la estructura **EjecutaStruct** empleada en los controles de diálogos y de la ventana. Por último la función **TStatic** genera un elemento de control estático o de texto, sobre el cual normalmente no se ejerce control, asignándolo a un puntero del tipo **PTStatic** para obtener control e interactuar en cada etapa del proceso enviando diferentes mensajes de acuerdo al avance del programa.

Se ha presentado la simplicidad de adicionar características al constructor de la ventana principal, pero debe tenerse en mente que varias de las funciones que realmente definen el comportamiento de la ventana son realizadas por funciones definidas en la clase base **TWindow** y ejecutadas por el constructor principal de la misma; por ejemplo, el cambiar el tamaño de la ventana, maximizarla, minimizarla, moverla, etc. Estas y muchas otras acciones permiten evaluar la trascendencia de las propiedades de *herencia* y *polimorfismo* de las clases.

La clase "*inicio*" cuenta también con un destructor definido por la función **~inicio()** el cual permite, cuando se selecciona la opción de cerrar la ventana, borrar objetos creados en la clase, liberar memoria asignada a estructuras o elementos en la clase. Para la clase *inicio* anterior la función destructor se define como:

```
inicio::~inicio()
{
  DeleteObject(menu);
}
```

en esta función destructor se borra el objeto *menu*, y además internamente se llama al destructor de la clase base para borrar la ventana principal.

Normalmente cuando se desea terminar una aplicación o un proceso dentro de la misma, puede cometerse el error de no guardar los cambios o bien olvidar el restablecer algún parámetro modificado. Por tanto, es importante que al intentar cerrar la aplicación se envíe un mensaje para confirmar la salida, siendo éste precisamente el objetivo de la función **CanClose**, la cual puede definirse de la manera siguiente:

```
BOOL inicio::CanClose()
{
  BOOL salir;

  if (GetApplication() ->ExecDialog(new TSalirDialog(this, "Salir")) == IDYES)
    {
        DestroyWindow(HWindow);
        Salir = TRUE;
    }
  else
        Salir = FALSE;
          return(Salir);
}
```

Esta función podría haberse generado con un simple **MessageBox**, pero con la intención de mejorar la visualización del programa se diseño esta forma. Con lo anterior se observa como protegerse de una selección errónea de la opción *salir* sin perder información, o bien salir definitivamente en caso de recibir una confirmación.

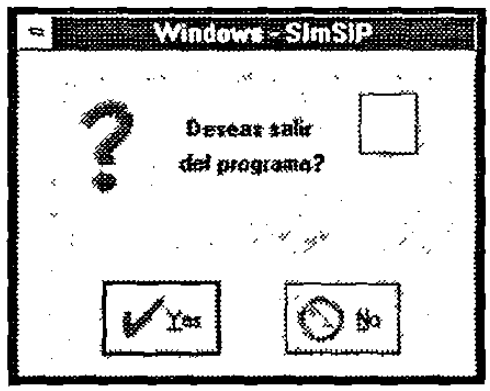


Fig. 2.1 *Caja de diálogo generada por la funcion CanClose().*

La función **GetWindowClass** permite cambiar atributos especiales de una ventana, tales como: color de fondo, icono representativo, tipo de cursor para el mouse, así como el asignar un menú como inicial o "default" de la ventana. Es necesario al definir una nueva función **GetWindowClass** dentro de una clase derivada, llamar dentro de la misma la respectiva función de la clase base y posteriormente incluir los nuevos atributos deseados. La definición de **GetWindowClass** se realiza de la siguiente forma:

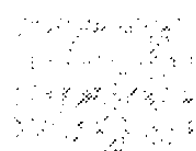
```
void inicio::GetWindowClass(WNDCLASS&      WndClass)
{
 TWindow::GetWindowClass(WndClass);

 hBKBrushBmp  = LoadBitmap(GetApplication()->hInstance,     "Bit_Map");
 hBKBrush  = CreatePatternBrush(hBKBrushBmp);
 WndClass.hbrBackground    = hBKBrush;

 GeneraStruct.Boton_1   = TRUE;
 EjecutaStruct.Boton_1   = TRUE;
 EjecutaStruct.Boton_2   = TRUE;

 strcpy(EjecutaStruct.Itera,   "T&olerancia");
}
```

Bit-Map



Debe observarse como es llamada inicialmente la función original de la clase base **TWindow**. La siguiente función asigna un *bitmap* denominado "Bit_Map", definido en el archivo de "resources", a la variable **hBKBrushBmp** y ésta se asigna, a través de la función **CreatPatternBrush**, como la característica del fondo de la ventana, dando una apariencia bastante atractiva a la presentación. Por último, se aprovecha esta función

para asignar valores iniciales a datos de estructuras, principalmente aquellos relacionados con conjuntos de controles agrupados: Radio Buttons, Check Boxes, etc., lo cual se detallará más adelante.

La función **Paint**, declarada en la clase **inicio**, es llamada cada vez que se produce un cambio en la visualización por cualquier aplicación o ventana en Windows. Windows envía el mensaje **WM_Paint** que es interceptado por la aplicación, la cual ejecuta la función **WMPaint**, que a su vez llama a la función **Paint** de **TWindow**. La función **Paint** puede ser redefinida para modificar sus características y rescribir en la ventana los elementos que se tenían anteriormente. Una forma de definir la función **Paint** sería:

```
void inicio::Paint(HDC  PaintDC, PAINTSTRUCT&)
  {
   static LOGFONT lf;
   HBRUSH hbr,
   HFONT Letra,
   short i=100,
         ypos=345;

     Se dibujan los recuadros de la pantalla principal
            x1 y1  x2  y2
   Dibuja(PaintDC,  0, 19, 637, 439, FALSE),
   Dibuja(PaintDC,  3, 416, 315, 436, TRUE),
   Dibuja(PaintDC,  316, 416, 634, 436, TRUE),

   lf.lfWeight=FW_NORMAL,
   lf.lfCharSet=OEM_CHARSET,
   lf.lfPitchAndFamily  FF_MODERN;
   lf.lfOutPrecision=OUT_STRING_PRECIS;
   strcpy(lf.lfFaceName,"Courier  New"),
   static char Letrero[]="Copyright  (c), DOCTORADO  INGENIERIA  ELECTRICA  - POTENCIA",

   Dibuja(PaintDC,  0, 0, 636, 19, TRUE),

   Letra=CreateFontIndirect(&lf ,

   DeleteObject(hbr),
   DeleteObject(Letra),
}
```

esta función dibuja principalmente recuadros y letreros en la ventana principal. Es importante señalar que los objetos que son creados en esta funcion, tales como: **Letra**

y **hbr** derivados de **HFONT** y **HBRUSH**, deben ser borrados en la misma función antes de que ésta termine, ya que cada vez que la función es llamada se separará un nuevo bloque de memoria para cada objeto, ocasionando problemas de memoria no sólo a la aplicación sino incluso al propio Windows.

Hasta este momento han sido descritas dos de las clases más importantes involucradas en el desarrollo de ambientación de programas bajo Windows, se utiliza el esquema presentado por la compañía Borland en sus Bibliotecas de funciones hacia Windows para el lenguaje C++ versión 3.1. A partir de estas dos clases es posible crear una aplicación sencilla que despliegue una ventana principal con menús, recuadros y diversos letreros en pantalla; aún cuando las funciones de respuesta a comandos de menús no hayan sido habilitadas. La forma de realizar la aplicación es la siguiente: dentro de la función principal **main**, **WinMain** para Windows, se crea un objeto **Flujos** a partir de la clase **TFlujos** derivada de **TApplication**. A partir del objeto **Flujos** se ejecuta la función **Run()**, miembro de su clase **TApplication**, de la forma **Flujos.Run()** que a su vez llama a las funciones internas de la clase como **InitInstance()** e **InitMainWindow()**, las cuales inicializan procesos, generan y muestran la ventana principal. La función **WinMain** tendrá la siguiente estructura:

```
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpCmdLine, int nCmdShow)
{
 TAplicacion inicia("SimSIP  Modulo de Flujos", hInstance, hPrevInstance,
            lpCmdLine, nCmdShow);
 inicia.Run();

 return inicia.Status;
}
```

la función **Run()** no regresa hasta que se destruye la ventana principal y se cierra la aplicación.

Así mismo, la función **InitMainWindow()** crea un objeto de ventana de la clase "*inicio*" derivada de **TWindow** en la forma siguiente:

```
void TFlujos::InitMainWindow()
{
        MainWindow  = new inicio(NULL, "Windows-SimSiP  (Módulo de Flujos)";
}
```

se observa como se le asignan los parámetros **AParent=NULL** por ser la ventana principal y **ATitle="Windows-SimSiP (Módulo de Flujos)"** como encabezado de la misma.

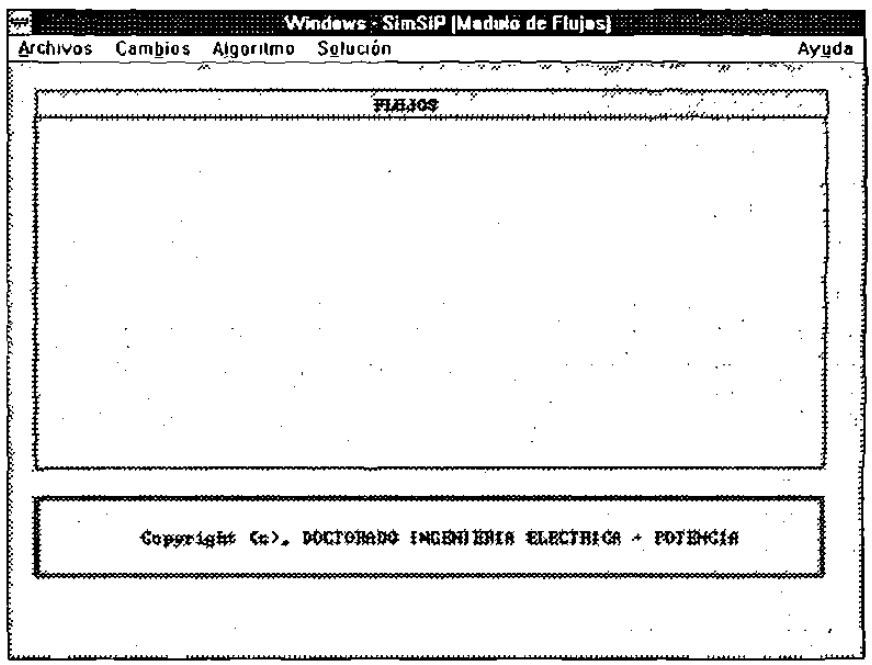


Fig. 2.2 *Pantalla principal, Modulo de Flujos*

Un punto importante, más no tratado aún, es la respuesta a comandos de menús en la ambientación. Cada vez que el usuario selecciona un menú, ya sea utilizando el ratón o por medio de teclas rápidas, se genera y envía un mensaje a la aplicación, la cual a su vez lo captura y procesa como a continuación se describe. Cada una de las opciones incluidas dentro de la barra de menús tiene asignado un identificador exclusivo, de tal forma que cuando el usuario selecciona un menú se genera un mensaje específico que la opción procesará llamando a la función correspondiente asignada al menú. La declaración de la función se coloca dentro de la clase que se desea responda a tal evento, como ejemplo se ha colocado en la definición de la clase *inicio* la declaración de

una función que responde a la opción del menú **Parámetros de solución**, la cual se define de la forma siguiente:

```
virtual void CMCambiosCorr(RTMessage Msg)
        = [CM_FIRST + CM_CambiosCorr ];
```

se observa el identificador **CM_CambiosCorr** para la opción de menú **Parámetros de solución** en la barra de menús. La definición de la función respectiva se realiza como:

```
void inicio::CMCambiosCorr(RTMessage)
{ }
```

Las funciones de respuesta de menús pueden realizarse de dos formas: una efectuando operaciones o acciones concretas en forma directa, además de activar o desactivar opciones para darle una continuidad y seguimiento a la ambientación, la otra consiste en presentar una o varias ventanas de diálogo consecutivas que permitan al usuario interactuar con los parámetros y opciones de la aplicación, activando o desactivando opciones, variando parámetros, tolerancias, límites de capacidad de elementos, sacar de operación elementos o bien reconectarlos en el sistema, etc. Las ventanas de diálogo son diferentes al tipo de ventanas derivadas de la clase **TWindow**, tanto en apariencia como en funcionamiento y utilización. En el siguiente apartado se describe el empleo de ventanas de diálogo y sus controles.

2.3.2 DIALOGOS Y CONTROLES

La interacción con parámetros, así como con las diferentes opciones de la ambientación se lleva acabo a través de diversos dispositivos de control como: **Edits** para introducir datos o cambiar parámetros, **RadioButtons** para seleccionar una opción entre varias posibles, **CheckBoxes** para activar múltiples opciones a la vez, **StaticText** para desplegar mensajes de texto ya sea fijos o cambiantes de acuerdo al desarrollo del proceso, **ListBoxes** para desplegar listas de nombres o parámetros para una selección directa, **ComboBoxes** este tipo de controles es una combinación del control **Edit** con el control **ListBox**, permitiendo interactuar con la búsqueda en la lista.

Normalmente la interacción con parámetros y opciones en una aplicación se lleva acabo por medio de controles, a través de acciones momentáneas dentro del ambiente, por esta razón la mayor parte de controles son colocados en un tipo especial de ventana denominada Diálogos. Estas ventanas de diálogo tienen características de forma y funcionamiento distintas a las derivadas de la clase **TWindow**.

Existen dos tipos de ventanas de diálogo que pueden utilizarse: *modal* y *modeless*. El tipo de diálogo *modal* se caracteriza por la corta duración de su visualización, ya que únicamente existirá mientras se introducen cambios o se seleccionan opciones en el mismo, después de lo cual se destruye. En cambio los diálogos del tipo *modeless* pueden estar presentes todo el tiempo que se requiera en la aplicación; permitiendo incluso interactuar con otros procesos de la misma. Al ejecutar diálogos tipo *modal* se detienen momentáneamente todos los procesos de la aplicación hasta que sea destruida.

Las ventanas de diálogo son derivadas de la clase **TDialog** y al igual que la clase **TWindow** analizada cuenta con elementos como: un constructor, un destructor y la función **CanClose** como miembros principales. A una ventana de este tipo se le puede incluir un menú, el cual se emplearía en la forma descrita en párrafos anteriores, para ventanas normales, así como las respuestas a comandos del menú. Un ejemplo de una clase denominada **Ejecucion** derivada de **TDialog** se presenta a continuación.

```
class Ejecucion:public  TDialog
{
 public:
   char Dato[80];
   PTStatic      TolR, TolI, Iteracion;
   PTEdit        Edita;
   PTRadioButton  RIButton1,  RIButton2;
   PTCheckBox    CheckBox;
   PTGroupBox    GroupBox;

   Ejecucion Ejecucion(PTWindowsObject     AParent, int ResourceId);
   ~Ejecucion();
   virtual void HandleGroupBoxMsg( TMessage&     Msg) = [ID_FIRST + 509];
 private:
   BOOL ValidDato();
};
```

Analizando la formación de la clase **Ejecucion** puede verse que han sido declarados diversos apuntadores a diferentes tipos de controles tales como: PTStatic, PTEdit, PTRadioButton, PTCheckBox y PTGroupBox, los cuales serán utilizados en la ventana de diálogo construida con esta clase. La declaración de la función constructor está dada como: **Ejecucion::Ejecucion(PTWindowsObject AParent, int ResourceId)**, en la definición de esta función se crean todos los controles que incluirá la ventana de diálogo. La definición de la ventana de diálogo tanto en tamaño, apariencia y funcionamiento, así como la creación de controles dentro de la misma, definiendo su posición, tamaño y diversas características de funcionamiento, pueden realizarse directamente a través de código o bien empleando algún editor comercial especializado en la generación de diálogos, controles, menús, etc., para ambiente Windows. Editores como: el SDK, el Resource Workshop del propio Borland, etc., reducen considerablemente el tiempo requerido para la generación de los diferentes elementos de visualización en el programa de ambientación, cada uno de ellos manejado por medio de un identificador (variable tipo entero), el cual a su vez permitirá el control de los elementos.

La definición de la función del constructor es la siguiente:

```
Ejecucion .Ejecucion(PTWindowsObject     AParent,  int  ResourceId) TDialog(AParent,    ResourceId)
{
  new TEdit(this,  502, sizeof(((inicio  *)Parent) >EjecutaStruct NDato 1)),
  TolR  = new TBStatic(this,  517, sizeof(((inicio  *)Parent) >EjecutaStruct TolR),    NULL);
  new TEdit(this,  503, sizeof(((inicio  *)Parent) >EjecutaStruct NDato_2)),
  TolI  = new TBStatic(this,  518, sizeof(((inicio  *)Parent)->EjecutaStruct.TolI),    NULL ),
  new TEdit(this,  504, sizeof(((inicio  *)Parent)->EjecutaStruct.NDato 3)),
  new TEdit(this,  505, sizeof(((inicio  *)Parent)->EjecutaStruct NDato_4)),
  new TEdit(this,  506, sizeof(((inicio  *)Parent) >EjecutaStruct NDato_5)),
  CheckBox   = new TBCheckBox(this,  ID_CHECK 1,  NULL ),
  GroupBox   = new TGroupBox(this,  509, NULL ),
  RTButton1  = new TBRadioButton(this,  ID BOTON 17,  GroupBox),
  RTButton2  = new TBRadioButton(this,  ID BOTON_18,  GroupBox),
  Iteracion  = new TBStatic(this,  515, sizeof(((inicio  *)Parent)->EjecutaStruct.Itera),    NULL);
  Edita      = new TEdit(this,  512, sizeof(((inicio  *)Parent)->EjecutaStruct NDato 6));
    Funci≤n necesaria para permitir cambiar el texto en el control TStatic
  Iteracion->EnableTransfer();
  TolR >EnableTransfer(),
  TolI >EnableTransfer(),
  TransferBuffer  = (void far*)&(((inicio  * Parent)->EjecutaStruct(),
}
```

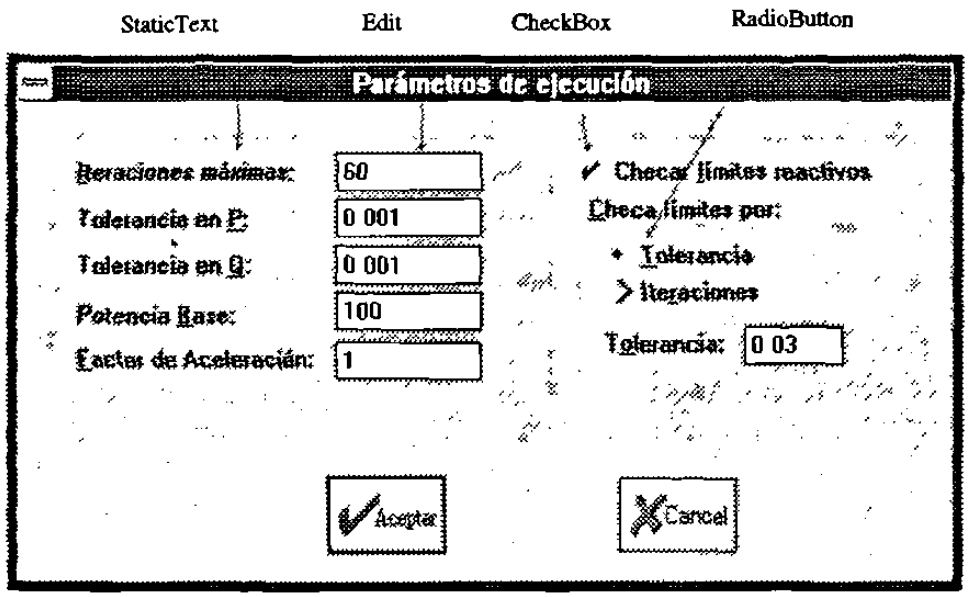


Fig. 2.3 *Ventana de diálogo derivada de la clase Ejecución*

La primera línea de código del constructor creará un control de tipo **Edit**, o sea una línea de entrada de datos. Al analizar el conjunto de parámetros requeridos por **TEdit**, se observa que el segundo parámetro es un entero el cual define el identificador del control en el diálogo que previamente ha sido diseñado a través del *Resource Workshop*. El tercer parámetro corresponde al número de caracteres que permitirá introducir el control al usuario, número que debe corresponder al tamaño de la variable **NDato_1** tipo **char** perteneciente a la estructura **EjecutaStruct**, la cual será definida más adelante.

El segundo elemento creado en el constructor es un control del tipo **Static**, los cuales normalmente son utilizados para colocar etiquetas a controles o bien textos dentro del diálogo. Puede observarse en este caso particular que se ha asignado el control al apuntador **TolR**, como dato miembro de la clase **Ejecucion**, de tipo **PTStatic**. Esta forma de crear el control permitirá cambiar en forma interactiva el texto desplegado; cambio que se encuentra deshabilitado para este tipo de controles. Habilitar el cambio del texto a controles de tipo Static se efectúa llamando a la función de transferencia de

la forma siguiente: **TolR->EnableTransfer()**, la cual se encuentra definida en la antepenúltima línea del constructor. Los parámetros requeridos son similares a los de controles de tipo **TEdit**, como segundo parámetro el identificador y como tercer parámetro el tamaño máximo de la cadena de texto.

Otro tipo de control generado en el constructor de la clase **Ejecucion** es el de los **Check Boxes**. Igual que al control de texto estático se asigna el control al apuntador **CheckBox** del tipo **PTCheckBox**, definido como dato miembro de la clase **Ejecucion**, se le pasa como segundo parámetro el identificador del control y como tercer parámetro, si se desea agrupar un conjunto de controles, un apuntador de agrupamiento del tipo **PTGroupBox**, en caso contrario se pasa el parámetro NULL como en el caso mostrado.

Los controles denominados **Radio Buttons** presentan un conjunto de opciones al usuario de las cuales sólo podrá seleccionar una a la vez. El segundo parámetro que se pasa a esta función es el identificador del control, y como tercer parámetro se coloca un apuntador de agrupamiento, el cual permitirá agrupar diversos conjuntos de controles del tipo **Radio Buttons** o bien **Check Boxes**.

En la descripción de los diversos controles presentados se ha omitido la definición del primer parámetro que recibe cada uno de ellos, ya que para todos es el mismo y se describe a continuación. Cada ventana y diálogo que es creado en una aplicación proviene de una ventana base denominada origen, a excepción de la ventana principal, a partir de la cual se generó la nueva. A esta última ventana se le pasa su ventana origen, parámetro que normalmente estará dado como **this** ya que la ventana activa, antes de generar la nueva, será la que mandó crearla. Los controles requieren, de la misma forma, como primer parámetro la ventana origen y puede observarse que a todos los controles creados en el diálogo se les ha pasado el parámetro **this**.

Cada uno de los controles en un diálogo requiere que se le asigne un valor, texto o estado inicial, también es necesario capturar los cambios y selecciones realizadas por

el usuario. Este intercambio de información se lleva a cabo por medio de una estructura de datos, la cual contará con un tipo de dato específico requerido por cada control: los controles **Static** y **Edit** requieren variables tipo *caracter*, los controles **Radio Buttons** y **Check Boxes** requieren variables tipo *unsigned* que definirán su estado ya sea que se encuentre seleccionado o no seleccionado, TRUE o FALSE.

La estructura de datos se crea definiendo un dato para cada control, exactamente en el mismo orden en que éstos han sido creados en el constructor, por ejemplo; para los controles incluidos en el constructor de la clase **Ejecucion**, la estructura de datos requerida tendrá la forma siguiente:

```
struct TEjecutaStruct
{
  char NDato_1[13],
       EdiR[20],
       NDato_2[13],
       Edi1[20],
       NDato_3[13],
       NDato_4[13],
       NDato_5[13],
  BOOL Boton_1,
       Boton_2,
       Boton_3,
  char Itera[14],
       NDato_6[13];
};
```

se observa el orden de creación de controles en el constructor como corresponde, en tipo de dato, a las variables definidas en la estructura. La última instrucción dentro del constructor de la clase **Ejecucion** transfiere la estructura **EjecutaSTruct**, derivada de la estructura anterior, como dato miembro en la clase **inicio**; la cual define precisamente la ventana origen (Parent Window), para el intercambio de información con los controles.

Una vez descrito el constructor de la clase **Ejecucion** se definirá el destructor de la misma como **~Ejecucion()**, siendo el trabajo principal de esta función el liberar la memoria ocupada por el diálogo al momento de ser destruido, la definición del

destructor es la siguiente:

```
Ejecucion::~Ejecucion()
{  delete TolR;
   delete Iteracion;
   delete Edita;
   delete RTButton1;
   delete RTButton2;
   delete GroupBox;
}
```

Es importante la función que ocupa el destructor en la aplicación, ya que sin éste se generarían problemas de memoria al dejar almacenados los elementos creados.

Por último, en la declaración de la clase **Ejecucion** se encuentra la función **virtual void HandleGroupMsg(RTMessage Msg) = [ID_FIRST + 509];** la cual es idéntica a las funciones creadas para respuesta a comandos de menús. Efectivamente, esta función es utilizada para manejar comandos generados al seleccionar controles de la ventana de diálogo; para el caso particular presentado esta función responde al seleccionar cualquiera de los dos **Radio Buttons** agrupados en el diálogo. Este tipo de funciones de respuesta a comandos puede definirse para cualquier control dentro del diálogo, el caso presentado se utiliza para cambiar texto de controles estáticos, así como el parámetro del último de los controles de edición.

Se han presentado los elementos de control comúnmente utilizados en una aplicación. Considerando tanto los detalles básicos, así como los más importantes, al crear un diálogo. Se ha mostrado el manejo de la respuesta a comandos de selección de controles, con lo cual es posible incrementar considerablemente la interacción de un programa y brindar efectos visuales que harán más atractiva la utilización del mismo.

Existe otro tipo de control llamado **ComboBox**, el cual se ha analizado por separado debido tanto a su forma de manejarse, como al impacto que brinda su utilización dentro de la ambientación. En un programa de ambientación se requiere

seleccionar elementos o parámetros del sistema de estudio, los cuales normalmente son identificados a través de nombres o claves que el usuario debe memorizar perfectamente para una selección correcta del mismo. Los controles **ComboBox** permiten agrupar la lista de nombres, moviéndose a través de ella para seleccionar el elemento deseado. Para mostrar la utilidad de este tipo de control se ha generado una nueva clase **Cambios** derivada de **TDialog**, por medio de la cual se generará una ventana de diálogo conteniendo un elemento de control del tipo **ComboBox**. El código que define la clase **Cambios** es el siguiente:

```
class Cambios:public  TDialog
{
 public:
   PTComboBox   TopoloCom,
   PTRadioButton  Rama_1, Rama 2, Shunt 1, Shunt 2, Shunt_3, Taps,

   Cambios::Cambios(PTWindowsObject     AParent,  int ResourceId),
   ~Cambios(),
   virtual  BOOL  CanClose();

   virtual  void  ActTopCom(RTMessage    Msg)
       = [ID FIRST  + 403 ],

   void  BorraCom(PTComboBox    P);
};
```

El constructor respectivo de la clase **Cambios** se define de la manera siguiente:

```
Cambios::Cambios(PTWindowsObject     AParent,  int ResourceId) TDialog(AParent,    ResourceId)
{
      Topologia
      TopoloCom   = new TComboBox(this,   403, sizeof NomNodo[0]),

        Rama_1  = new TBRadioButton(this,   ID BOTON 10,  NULL ),
        Rama_2  = new TBRadioButton(this,   ID_BOTON 11,  NULL );
       Shunt 1  = new TBRadioButton(this,   ID BOTON 12,  NULL ),
       Shunt_2  = new TBRadioButton(this,   ID BOTON_13,  NULL);
       Shunt_3  = new TBRadioButton(this,   ID BOTON 14,  NULL),
         Taps   = new TBRadioButton(this,   ID BOTON_15,  NULL ),

      TopoloCom->EnableTransfer(),

      TransferBuffer   = (void far*)&(((inicio   *)Parent) > TopoloStruct),
}
```

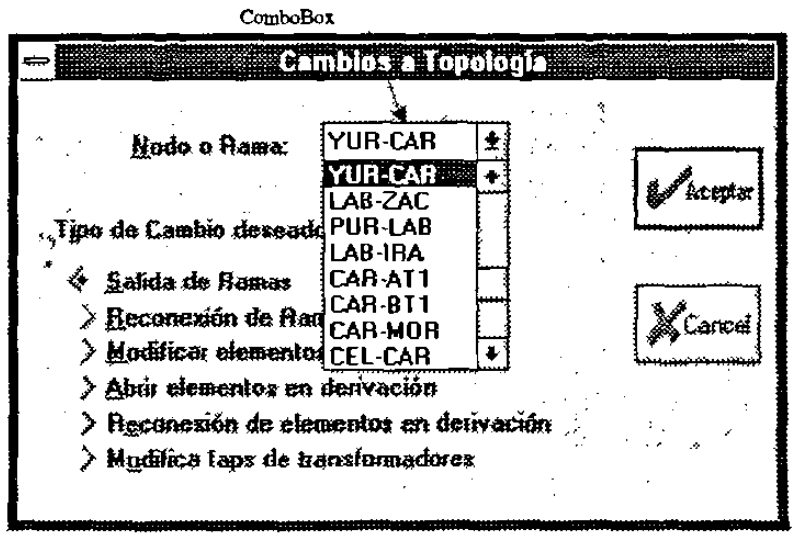


Fig. 2.4 *Ventana de dialogo empleando ComboBoxes.*

El control **ComboBox** es asignado a un puntero **TopoloCom** del tipo **PTComboBox** definido como dato miembro de la clase *Cambios*, los parámetros asignados al control son nuevamente: el primer parámetro es **this**, el segundo parámetro es un *identificador* definido en el archivo de "*Resources*" y como tercer parámetro la longitud máxima de la cadena con el nombre en la lista.

La estructura de datos que manipulará los datos del diálogo se define como sigue:

```
struct TCambiosCom
{
  PTComboBoxData   NDato,

  BOOl  Boton_1,
        Boton_2,
        Boton_3,
        Boton_4,
        Boton_5,
        Boton_6,
};
```

Esta estructura muestra un dato especial para el **ComboBox** del tipo **PTComboBoxData**, el cual a su vez requiere ser inicializado dentro del constructor de

la clase *inicio*, la cual define la ventana principal, adicionando dentro de su constructor la línea de código siguiente:

```
inicio::inicio(PTWindowsObject    AParent,  LPSTR  ATitle):TWindow(AParent,    ATitle)
{
  TopoloStruct.NDato   = new TComboBoxData(),
  TopoloStruct.NDato->AddString("");
}
```

además es necesario al terminar la aplicación borrar este objeto creado, modificando el destructor de la clase inicio de la forma siguiente:

```
inicio::~inicio()
{
  delete  TopoloStruct.NDato,
  DeleteObject(menu),
}
```

Las funciones miembro adicionales de la clase **Cambios** permiten responder a mensajes producidos al seleccionar los controles.

2.4 CONCLUSIONES

Se han presentado las herramientas fundamentales para crear un programa interactivo en ambientación Windows. Se han definido las clases principales a partir de las cuales pueden derivarse nuevas características de funcionamiento. Se presentó código fuente que puede ser tomado como base para ser adecuado a las necesidades propias de una nueva aplicación. Se han descrito los diferentes controles a incluir en ventanas de diálogo, principalmente se mostró la flexibilidad que brinda su inclusión en la interacción con el ambiente.

CAPITULO 3

INTERACCION ENTRE AMBIENTACION EN WINDOWS Y ALGORITMOS EN MODO TEXTO

En este capítulo se describe la forma en que interactuan los algoritmos, implementados en lenguajes de programación distintos al C++, con la ambientación dada al simulador digital. Se describen brevemente los métodos de Gauss-Seidel, Newton-Raphson y Desacoplado Rápido para el problema de *Flujos de Carga*; habiéndose programado en QuickBasic, Turbo Pascal y QuickBasic respectivamente. Se supone sistemas trifásicos balanceados por lo cual se emplea la representación de secuencia positiva de la red.

3.1 INTERACCION ENTRE AMBIENTACION Y ALGORITMOS

En el proceso de intercambio de información entrada/salida, la interacción entre el programa de ambientación y los algoritmos se lleva acabo a través de archivos auxiliares. Los archivos de datos para el caso base son cargados a memoria dentro de la ambientación, la cual genera un nuevo archivo de datos con los cambios y especificaciones establecidos por el usuario. Por su parte los algoritmos se ejecutan usando el archivo auxiliar y generando, después de su ejecución, un grupo de varios archivos, los cuales contienen los reportes de resultados obtenidos. Los reportes serán mostrados por medio de un editor de texto dentro del programa de ambientación pero en forma tabular. La posibilidad de presentar en forma gráfica los resultados, es decir sobre un diagrama unifilar del sistema, representa una alternativa importante, y que es posible desarrollar con las herramientas de software actuales de una manera relativamente sencilla. Aspecto que se planea implementar en un futuro cercano.

La posibilidad de interactuar en forma directa entre los algoritmos y la ambientación, no es factible por la restricción que implica el haber generado en

diferentes lenguajes de programación cada uno de los programas. La ambientación fue generada en C++[34] mientras que los algoritmos se implementaron en QuickBasic[35] y Turbo Pascal[36]. Lo importante en esta investigación fue la posibilidad de incrementar la flexibilidad y utilidad en el manejo de algoritmos programados con anterioridad, con la característica de haber sido probados ampliamente con diferentes sistemas de prueba[18,21] y las condiciones más variadas de carga y topología.

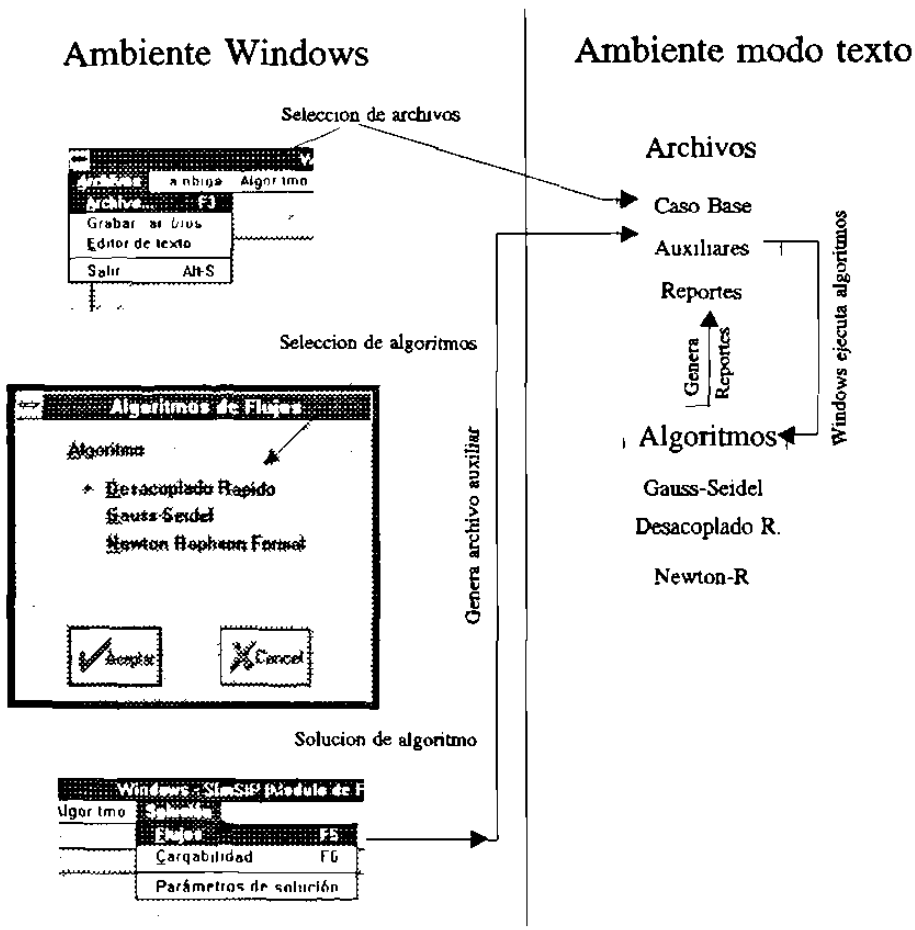


Fig. 3.1 *Interaccion entre ambientacion en Windows y algoritmos en modo texto*

Con la finalidad de mostrar los criterios tomados en la implementación de los diferentes algoritmos, se describe brevemente cada uno de éstos en los apartados siguientes.

3.2 FLUJOS DE CARGA

El estudio de flujos de carga es uno de los estudios más comúnmente utilizados para el análisis de los sistemas eléctricos de potencia. En el estudio de flujos se analizan, bajo condiciones específicas de generación y carga del sistema, la distribución de los flujos de potencia en la red, así como los voltajes nodales prevalecientes para la condición de operación especificada. Esta información permite tomar acciones para evitar la sobrecarga de elementos como líneas de transmisión y transformadores, así como el mantener un perfil de voltaje dentro de un rango de servicio permisible, a través de la conexión o desconexión de bancos de capacitores y reactores o bien por medio de la generación o absorción de reactivos por parte de los generadores conectados a la red y dentro de límites especificados de acuerdo a sus curvas de capabilidad.

El modelo de flujos de carga que ha sido implementado en este simulador interactivo presupone para su análisis sistemas de potencia trifásicos balanceados empleando, por tanto, únicamente la red de secuencia positiva. Se manejan los elementos convencionales que conforman una red eléctrica como son: líneas de transmisión, transformadores con tap fuera del valor nominal, elementos de compensación en derivación tanto inductivos como capacitivos, así como los generadores y cargas del sistema[23].

3.2.1 FORMULACION DEL ESTUDIO DE FLUJOS DE CARGA

Un estudio de flujos de carga proporciona como resultados voltajes y ángulos nodales del sistema, a partir de la demanda y generación especificadas. Los voltajes nodales complejos se utilizan para calcular los flujos de potencia por los diferentes elementos de la red[23]. Mediante el empleo de la formulación del análisis nodal se plantea el cálculo de los voltajes nodales en una red eléctrica (3.1).

$$\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ . & . & . & . \\ . & . & . & . \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ . \\ . \\ V_n \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ . \\ . \\ I_n \end{bmatrix} \qquad \textbf{(3.1)}$$

Los términos independientes en (3.1) representan las inyecciones netas de corrientes nodales del sistema las cuales normalmente no son conocidas, en cambio se conocen las inyecciones netas de potencias nodales a partir de las cuales se pueden obtener las corrientes respectivas. La expresión (3.2) muestra la relación de corrientes nodales en función de las inyecciones netas de potencia nodales.

$$I_i = \frac{S_i^*}{V_i^*} = \frac{P_i - j\,Q_i}{V_i^*} \qquad \textbf{(3.2)}$$

Sustituyendo (3.2) en (3.1) se obtiene finalmente el sistema de ecuaciones (3.3) a resolver.

$$\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ . & . & . & . \\ . & . & . & . \\ Y_{n1} & Y_{n2} & \cdots & Y_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ . \\ . \\ V_n \end{bmatrix} = \begin{bmatrix} \dfrac{P_1 - j\,Q_1}{V_1^*} \\ \dfrac{P_2 - j\,Q_2}{V_2^*} \\ . \\ \dfrac{P_n - j\,Q_n}{V_n^*} \end{bmatrix} \qquad \textbf{(3.3)}$$

La expresión (3.3) representa un sistema de ecuaciones no-lineales, requiriendo emplear un algoritmo iterativo[13] para su solución. Los métodos iterativos que serán analizados en este trabajo para el análisis de flujos de carga son: Gauss-Seidel, Newton-Raphson y Desacoplado Rápido. Los tres algoritmos han sido implementados en el

simulador digital interactivo y sus características principales se presentan a continuación.

3.3 ALGORITMO DE FLUJOS GAUSS-SEIDEL

En este algoritmo se desarrollan las expresiones de voltaje de (3.3) en la forma siguiente.

$$
\begin{aligned}
V_1 &= \frac{1}{Y_{11}}\left[\frac{P_1 - j\,Q_1}{V_1^*} - Y_{12}V_2 - Y_{13}V_3 - \ldots - Y_{1n}V_n\right] \\
V_2 &= \frac{1}{Y_{22}}\left[\frac{P_2 - j\,Q_2}{V_2^*} - Y_{21}V_1 - Y_{23}V_3 - \ldots - Y_{2n}V_n\right] \\
V_n &= \frac{1}{Y_{nn}}\left[\frac{P_n - j\,Q_n}{V_n^*} - Y_{n1}V_1 - Y_{n2}V_2 - \ldots Y_{(n\ 1)(n-1)}V_{(n-1)}\right]
\end{aligned}
\qquad \textbf{(3.4)}
$$

Las expresiones en (3.4) se resuelven una a una en forma secuencial, habiendo establecido voltajes iniciales a partir de los cuales se obtienen los voltajes mejorados. Cada voltaje obtenido es utilizado directamente en el cálculo del nuevo voltaje del nodo siguiente, este proceso se generaliza en la expresión (3.5), en la cual $V_i^{(k+1)}$ representa el voltaje del nodo *i* en la iteración $k+1$ y $V_i^{(k)}$ representa el voltaje en la iteración k.

$$
\begin{aligned}
V_1^{(k+1)} &= \frac{1}{Y_{11}}\left[\frac{P_1 - j\,Q_1}{V_1^{*(k)}} - Y_{12}V_2^{(k)} - Y_{13}V_3^{(k)} - \ldots - Y_{1n}V_n^{k}\right] \\
V_2^{(k+1)} &= \frac{1}{Y_{22}}\left[\frac{P_2 - j\,Q_2}{V_2^{*(k)}} - Y_{21}V_1^{(k+1)} - Y_{23}V_3^{(k)} - \ldots - Y_{2n}V_n^{(k)}\right] \\
V_n^{(k+1)} &= \frac{1}{Y_{nn}}\left[\frac{P_n - j\,Q_n}{V_n^{*(k)}} - Y_{n1}V_1^{(k+1)} - Y_{n2}V_2^{(k+1)} - \ldots Y_{(n\ 1)(n-1)}V_{(n-1)}^{(k+1)}\right]
\end{aligned}
\qquad \textbf{(3.5)}
$$

Se establecen dos criterios de convergencia en este método, la máxima variación absoluta en los voltajes nodales entre iteraciones y el máximo desajuste de potencia nodal. Ambos criterios se presentan en (3.6).

$$\begin{aligned} V_i^{(k+1)} - V_j^{(k)} &\le \varepsilon \\ \left| P_i^G - P_i^D - P_i^{calc} \right| &\le \varepsilon_P \\ Q_i^G - Q_i^D - Q_i^{calc} \Big| &\le \varepsilon_Q \end{aligned} \qquad (3.6)$$

En la expresión (3.6) el término ε se establece como una tolerancia dentro del rango (1e-4 $\le \varepsilon \le$ 1e-6) para voltajes, mientras que para desajustes de potencia ε_P y ε_Q podrían ser menos restrictivos, como 0.01 pu, dependiendo de la precisión deseada. El número de iteraciones del algoritmo para encontrar solución es elevado, además de incrementarse en forma considerable al aumentar el tamaño del sistema[18].

La expresión (3.5) genera resultados de magnitud de voltaje y ángulos nodales, lo cual es satisfactorio para los nodos de carga del sistema, no así para los nodos de voltaje controlado los cuales requieren mantener la magnitud de voltaje en un valor previamente especificado, $|V_i|^{esp}$. Debido a esta restricción, en los nodos tipo PV se refleja únicamente el cambio en el ángulo nodal, obtenido en cada iteración, manteniendo la magnitud del voltaje en el valor especificado. Los nodos de voltaje controlado normalmente incluyen generadores, compensadores síncronos o algún elemento de compensación de potencia reactiva, a través del cual se logra mantener la magnitud de voltaje en el valor especificado; lo anterior mediante la inyección o absorción de reactivos a la red dentro de los límites de operación. En caso de no poder satisfacer el suministro/consumo de reactivos requerido, la magnitud de voltaje no se puede mantener y el tipo de nodo debe cambiarse a tipo PQ; fijando en el límite violado la cantidad de reactivos Q.

En la Figura 3.2 se muestra el diagrama de flujo completo del algoritmo Gauss-Seidel implementado.

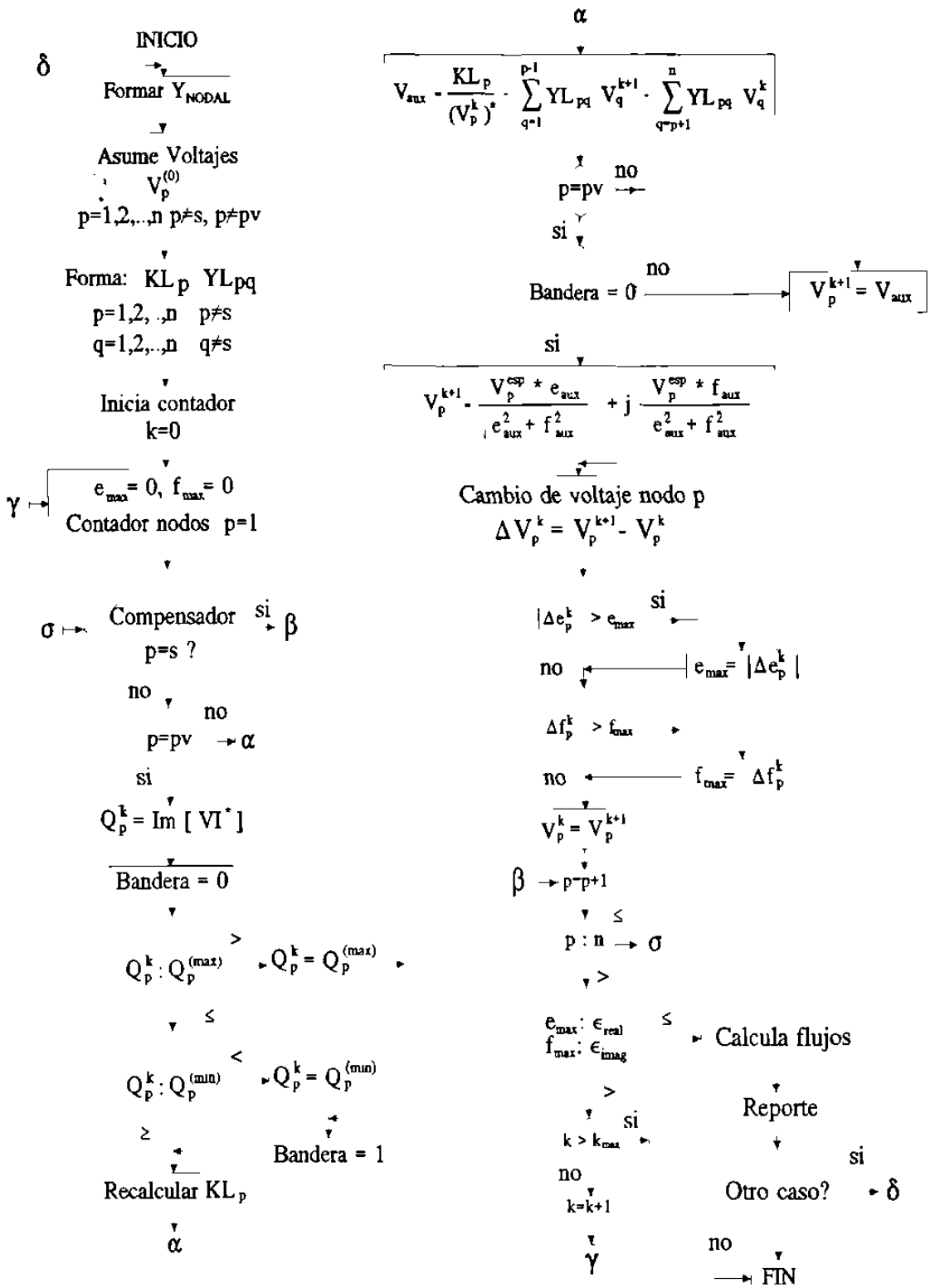


Fig. 3.2 *Diagrama de flujo correspondiente al algoritmo Gauss Seidel*

3.4 ALGORITMO DE FLUJOS NEWTON-RAPHSON

El algoritmo de Newton-Raphson para el análisis de flujos de carga se basa[19] en el desarrollo en serie de Taylor de las ecuaciones de desajustes de potencia nodal real y reactiva para cada nodo. El balance entre la potencia nodal generada, la potencia de demanda y la potencia nodal calculada puede expresarse como desajustes nodales dados por (3.7).

$$\begin{aligned} \Delta P_p &= P_p^G - P_p^D - P_p^{calc} \\ \Delta Q_p &= Q_p^G - Q_p^D - Q_p^{calc} \end{aligned} \qquad \textbf{(3.7)}$$

donde:

P_p^G, Q_p^G: Potencia generada real y reactiva del nodo p.

P_p^D, Q_p^D: Potencia de carga real y reactiva del nodo p.

P_p^{calc}, Q_p^{calc}: Potencia calculada real y reactiva del nodo p.

Las ecuaciones (3.8) y (3.9) muestran las expresiones de potencias calculadas real y reactiva de (3.7), en su forma polar[20].

$$P_p^{calc} = \sum_{q=1}^{n} V_p \; V_q | (G_{pq} \mathrm{Cos}\delta_{pq} + B_{pq} \mathrm{Sen}\delta_{pq}) \qquad \textbf{(3.8)}$$

$$Q_p^{calc} = \sum_{q=1}^{n} |V_p \; V_q \; (G_{pq} \mathrm{Sen}\delta_{pq} - B_{pq} \mathrm{Cos}\delta_{pq}) \qquad \textbf{(3.9)}$$

donde: $\delta_{pq} = \delta_p - \delta_q$

El objetivo es llevar a cero los desbalances nodales de manera iterativa, o sea lograr que $\Delta P_i = 0$ y $\Delta Q_i = 0$, por lo cual al utilizar la serie de Taylor en ambas expresiones, incluyendo sólo los términos de primer orden, se obtiene la forma linealizada (3.10) que se resuelve por un procedimiento directo.

$$\begin{bmatrix} \Delta P_1 \\ \vdots \\ \Delta P_{n-1} \\ \Delta Q_1 \\ \vdots \\ \Delta Q_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{\partial P_1}{\partial \delta_1} & \cdots & \frac{\partial P_1}{\partial \delta_{n-1}} & \frac{\partial P_1}{\partial V_1} & \cdots & \frac{\partial P_1}{\partial V_{n-1}} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial P_{n-1}}{\partial \delta_1} & \cdots & \frac{\partial P_{n-1}}{\partial \delta_{n-1}} & \frac{\partial P_{n-1}}{\partial V_1} & \cdots & \frac{\partial P_{n-1}}{\partial V_{n-1}} \\ \frac{\partial Q_1}{\partial \delta_1} & \cdots & \frac{\partial Q_1}{\partial \delta_{n-1}} & \frac{\partial Q_1}{\partial V_1} & \cdots & \frac{\partial Q_1}{\partial V_{n-1}} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial Q_{n-1}}{\partial \delta_1} & \cdots & \frac{\partial Q_{n-1}}{\partial \delta_{n-1}} & \frac{\partial Q_{n-1}}{\partial V_1} & \cdots & \frac{\partial Q_{n-1}}{\partial V_{n-1}} \end{bmatrix} \begin{bmatrix} \Delta \delta_1 \\ \vdots \\ \Delta \delta_{n-1} \\ \frac{\Delta V_1}{|V_1|} \\ \vdots \\ \frac{\Delta V_{n-1}}{V_{n-1}} \end{bmatrix} \quad \textbf{(3.10)}$$

El cálculo de los términos de primeras derivadas en (3.10) está dado por (3.11) a (3.18)[20].

$$\frac{\partial P_p}{\partial \delta_p} = -Q_p - |V_p|^2 B_{pp} \quad \textbf{(3.11)}$$

$$\frac{\partial P_p}{\partial V_p} = P_p + V_p^2 G_{pp} \quad \textbf{(3.12)}$$

$$\frac{\partial Q_p}{\partial \delta_p} = P_p - |V_p|^2 G_{pp} \quad \textbf{(3.13)}$$

$$\frac{\partial Q_p}{\partial V_p} = Q_p - V_p^2 B_{pp} \quad \textbf{(3.14)}$$

$$\frac{\partial P_p}{\partial \delta_q} = V_p V_q (G_{pq} \mathrm{Sen}\delta_{pq} - B_{pq} \mathrm{Cos}\delta_{pq}) \quad \textbf{(3.15)}$$

$$\frac{\partial P_p}{\partial V_q} = V_p V_q (G_{pq} \mathrm{Cos}\delta_{pq} + B_{pq} \mathrm{Sen}\delta_{pq}) \quad \textbf{(3.16)}$$

$$\frac{\partial Q_p}{\partial \delta_q} = -V_p V_q (G_{pq} \mathrm{Cos}\delta_{pq} + B_{pq} \mathrm{Sen}\delta_{pq}) \quad \textbf{(3.17)}$$

$$\frac{\partial Q_p}{\partial V_q} = V_p V_q (G_{pq} \mathrm{Sen}\delta_{pq} - B_{pq} \mathrm{Cos}\delta_{pq}) \quad \textbf{(3.18)}$$

donde:

P_p, Q_p son las potencias activa y reactiva calculadas.

$|V_p|$, $|V_q|$ son magnitudes de los voltajes nodales p y q respectivamente.

G_{pq}, B_{pq} valores de conductancia y susceptancia, elementos de Y_{nodal} entre nodos p y q.

El modelo matemático presentado corresponde a la versión polar del método de Newton-Raphson; existiendo la posibilidad de desarrollar la versión rectangular[21] correspondiente. Uno de los requerimientos del método es especificar los voltajes de arranque con valores cercanos a la solución, lo cual, si no se cumple, puede originar problemas de convergencia. En el análisis de SEP una condición inicial de voltajes, alejada de su valor nominal, puede presentar este tipo de problema numérico. La Figura 3.3 muestra el diagrama de flujo completo para el desarrollo del método de Newton-Raphson; los nodos de voltaje controlado que rebasan límites de generación de reactivos se cambian a nodos tipo carga, estos nodos se restablecen a tipo de voltaje controlado en la siguiente iteración; siempre que la generación de reactivos esté dentro de sus límites especificados.

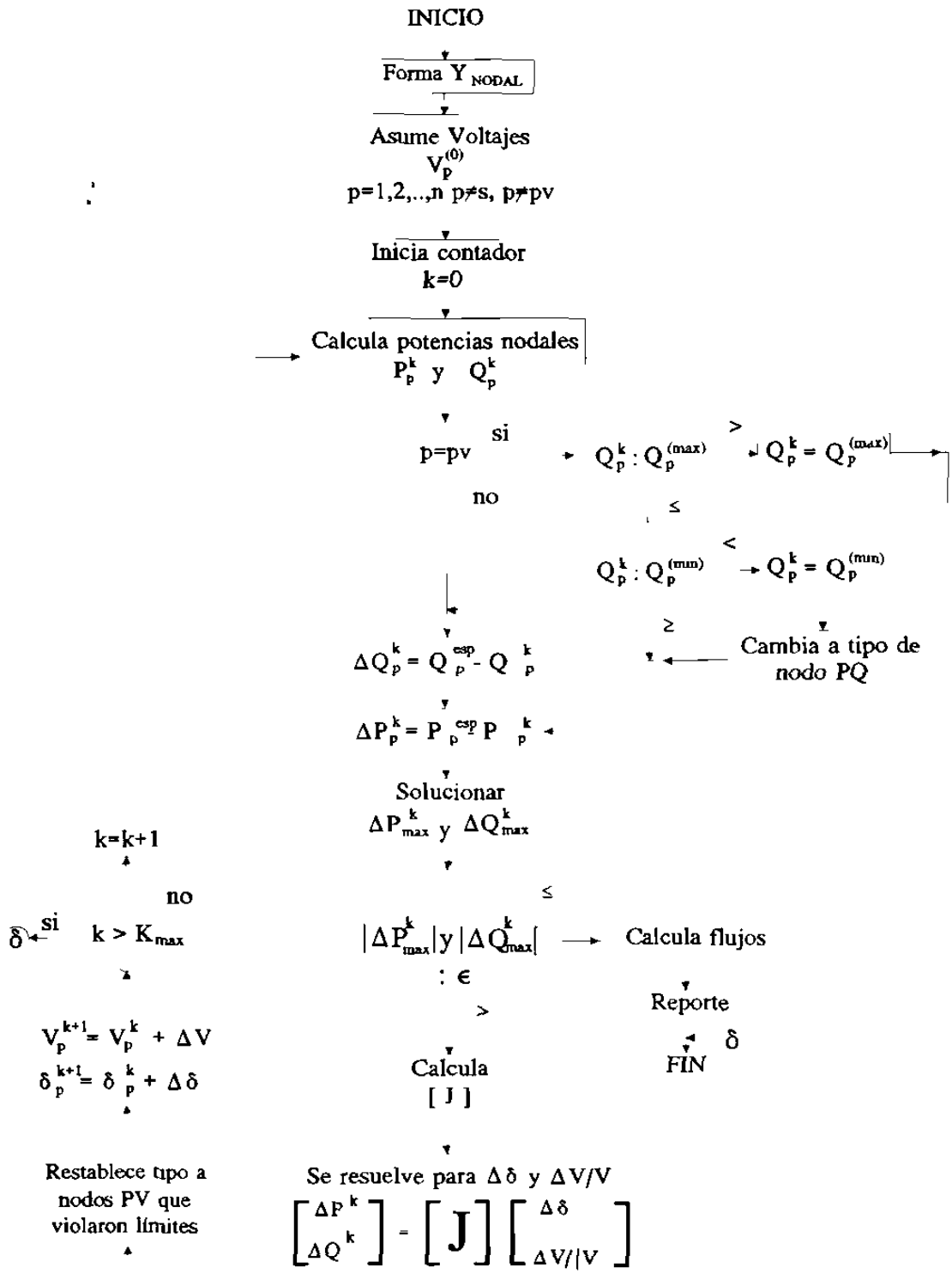


Fig. 3.3 *Diagrama de flujo correspondiente al algoritmo Newton Raphson*

3.5 ALGORITMO DE FLUJOS DESACOPLADO RAPIDO

El algoritmo desacoplado rápido para el análisis de flujos de carga se desarrolló en base al método de Newton-Raphson[17], se desprecian términos cuando el acoplamiento entre variables es pequeño, tales como: potencia real con magnitud de voltaje y potencia reactiva con ángulos nodales. Se consideran tanto condiciones normales de operación como características físicas de los elementos del sistema, siendo las propiedades más relevantes para el desarrollo del algoritmo las siguientes:

La expresión (3.10) del algoritmo de Newton-Raphson normalmente se maneja en forma condensada como se presenta en (3.19),

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ J & L \end{bmatrix} \begin{bmatrix} \Delta \delta \\ \frac{\Delta V}{|V|} \end{bmatrix} \tag{3.19}$$

donde el *Jacobiano* es la matriz (3.20)

$$\begin{bmatrix} H & N \\ J & L \end{bmatrix} \tag{3.20}$$

La primera aproximación consistió en despreciar de la matriz *Jacobiana* las submatrices [N] y [J], las cuales representan el acoplamiento entre la potencia real con la magnitud de voltaje y la potencia reactiva con el ángulo nodal respectivamente. Con la simplificación anterior se obtienen dos conjuntos de ecuaciones independientes, a resolver en forma alternada. Las expresiones en (3.21) forman el método Desacoplado[16].

$$\begin{aligned} [\Delta P] &= [H]\,[\Delta \delta] \\ [\Delta Q] &= [L] \left[\frac{\Delta V}{V} \right] \end{aligned} \tag{3.21}$$

La matriz [H] está definida por (3.11) y (3.15), a su vez la matriz [L] se define por (3.14) y (3.18). Por consideraciones prácticas en la operación de los sistemas eléctricos de potencia, así como de las características físicas de elementos, es posible usar las

aproximaciones siguientes:

$$Cos\delta_{pq} \approx 1; \quad G_{pq} Sen\delta_{pq} << B_{pq}; \quad Q_p << B_{pp} V_p^2$$

como resultado (3.21) se convierte en (3.22)

$$\begin{aligned} [\Delta P] &= [V \cdot B' \cdot V]\ [\Delta\delta] \\ [\Delta Q] &= [V \cdot B'' \cdot V] \left[\frac{\Delta V}{|V|}\right] \end{aligned} \qquad \textbf{(3.22)}$$

Las matrices [B'] y [B"] se forman con términos negativos de los elementos de susceptancia de la matriz Y_{NODAL}, obteniendo un desacoplamiento adicional en la formación de ambas *matrices*. De la *matriz* [B'] se eliminan los elementos de la red que afectan predominantemente el flujo de potencia reactiva, como son: el efecto capacitivo en las líneas de transmisión, capacitores y reactores en derivación y las ramas de compensación en paralelo de transformadores con tap fuera del nominal. En la matriz [B"] se elimina el efecto de defasamiento producido por transformadores defasadores y se elimina la resistencia de elementos del sistema al formar esta matriz.

Despejando los términos de voltaje en la expresión (3.22) se obtiene (3.23)

$$\begin{aligned} \left[\frac{\Delta P}{V}\right] &= [B' \cdot V]\ [\Delta\delta] \\ \left[\frac{\Delta Q}{V}\right] &= [B'']\ [\Delta V] \end{aligned} \qquad \textbf{(3.23)}$$

De la *expresión* (3.23) se *elimina* la influencia del flujo de potencia reactiva sobre el cálculo de $\Delta\delta$ haciendo los términos V de lado derecho de (3.23) la unidad, obteniéndose finalmente (3.24) para resolver en forma directa en cada iteración.

$$\left[\frac{\Delta P}{V}\right] = [B'][\Delta\delta]$$
$$\left[\frac{\Delta Q}{V}\right] = [B''][\Delta V] \qquad \textbf{(3.24)}$$

Los desacoplamientos y aproximaciones realizados en este algoritmo imponen ciertas restricciones prácticas en su uso, especialmente relacionadas con la convergencia de la solución, tales como: sólo podrá emplearse en SEP con niveles de tensión donde la relación *x/r* de sus elementos sea alto (5-10). Los sistemas reducidos a través de técnicas de equivalentes pueden generar ramas equivalentes con valores que invaliden las aproximaciones del algoritmo.

En el simulador desarrollado el manejo de límites de generación de potencia reactiva en los nodos de voltaje controlado en el algoritmo *Desacoplado Rápido* se implementó en base a *factores de sensitividad*[22,23], así se modifica el voltaje del nodo al valor propuesto por el factor de sensitividad para mantener fija la generación de potencia reactiva en el límite rebasado. De ésta forma en el algoritmo implementado el nodo siempre mantiene su condición de nodo de voltaje controlado, pero su valor se modifica de acuerdo al factor de sensitividad. Los factores de sensitividad se obtienen como la razón de cambio del voltaje necesario para producir una inyección de reactivos específica, o sea ΔV_i ΔQ_i, estos factores se obtienen como los elementos diagonales de la inversa de la matriz [B"] del modelo reactivo, el análisis correspondiente se muestra en el apartado siguiente.

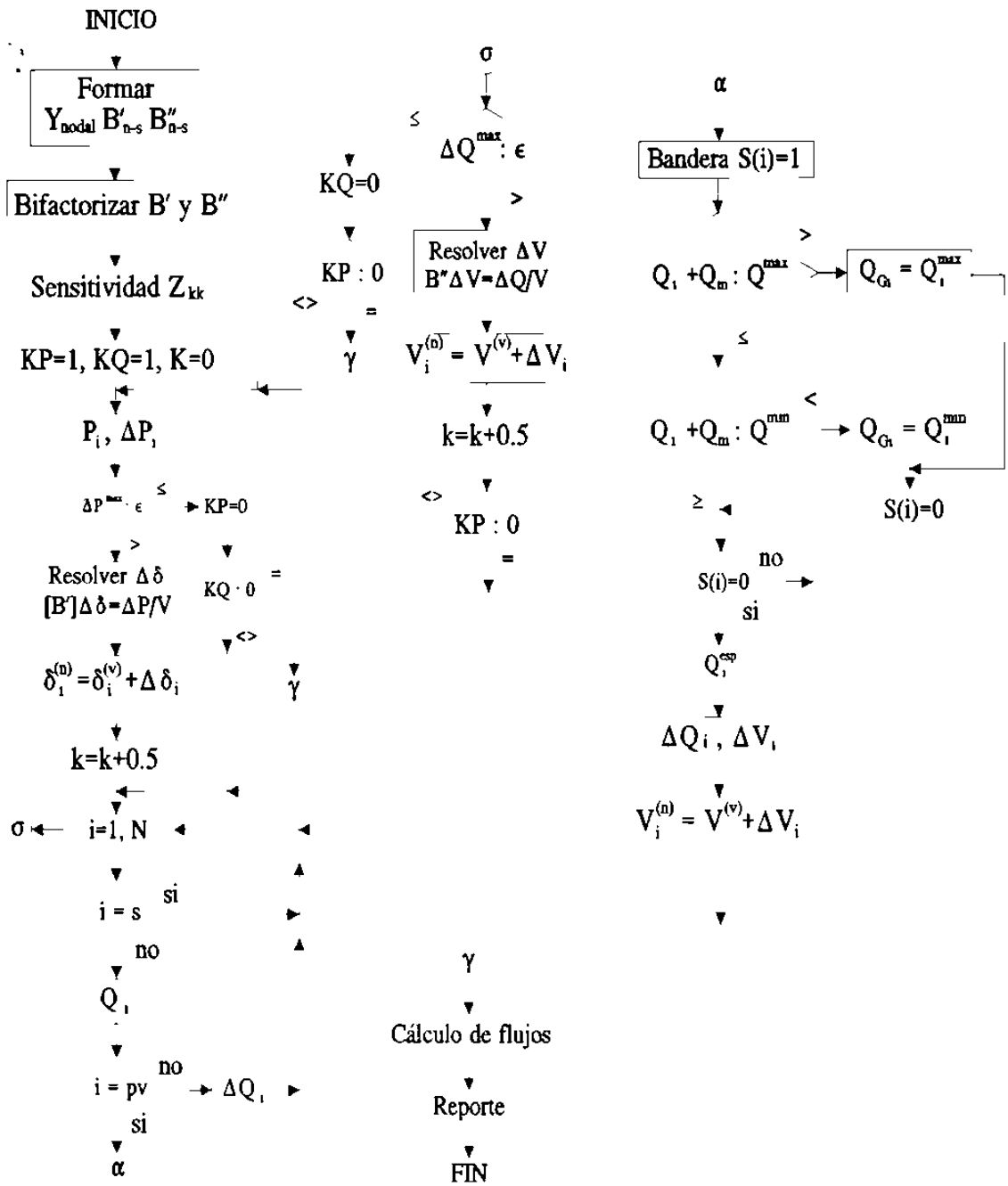


Fig. 3.4 *Diagrama de flujo correspondiente al algoritmo Desacoplado Rapido*

3.6 FACTORES DE SENSITIVIDAD, MODELO Q-V.

Para ilustrar el concepto y mostrar el cálculo de los factores de sensitividad se empleará el sistema mostrado en la Figura 3.5. Se definen los nodos de la manera siguiente:

Nodo 1: Nodo tipo PV o de voltaje controlado.

Nodos 2 y 3: Nodos tipo PQ o de carga.

Nodo 4: Nodo tipo S o compensador.

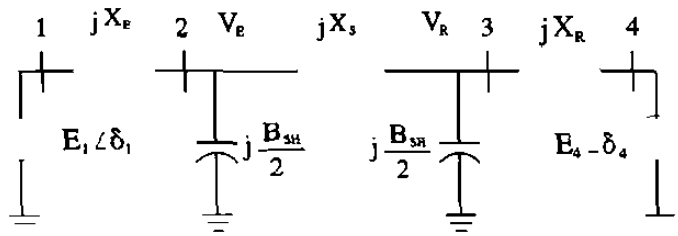


Fig. 3.5 *Sistema de prueba para análisis de sensitividad*

El modelo reactivo correspondiente al sistema de la Figura 3.5, incluyendo la ecuación del nodo 1 de voltaje controlado tiene la forma presentada en (3.25).

$$\begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22} & B_{23} \\ 0 & B_{32} & B_{33} \end{bmatrix} \begin{bmatrix} \Delta V_1 \\ \Delta V_2 \\ \Delta V_3 \end{bmatrix} = \begin{bmatrix} \dfrac{\Delta Q_1}{V_1} \\ \dfrac{\Delta Q_2}{V_2} \\ \dfrac{\Delta Q_3}{V_3} \end{bmatrix} \qquad \textbf{(3.25)}$$

Dentro del proceso iterativo del estudio de flujos, en el momento que sea rebasado el límite máximo o mínimo de generación de reactivos en un nodo de voltaje controlado, por ejemplo el nodo 1 del sistema de la Figura 3.5, se plantea un cambio ΔQ_1 en la potencia reactiva generada tal que ésta permanezca en el límite violado. El cambio en

la inyección ΔQ_1 será producido moviendo el voltaje del nodo 1, en un valor ΔV_1. El modelo reactivo (3.25) esquematizado en la Figura 3.6 se empleará para la obtención del factor de sensitividad $\Delta V_1/\Delta Q_1$ deseado.

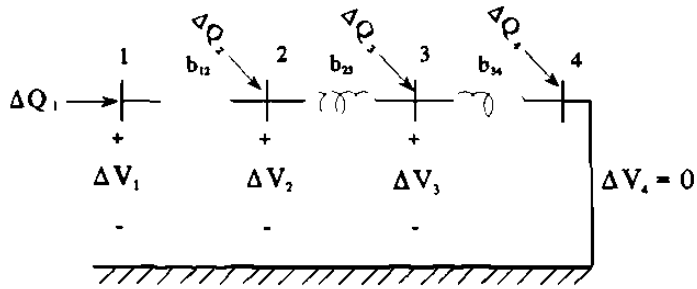


Fig. 3.6 *Modelo reactivo para el análisis de sensitividad*

Del modelo reactivo en la Figura 3.6 correspondiente a (3.25), se establece que en la solución se tendrá inyección cero en los nodos 2, 3 y 4, o sea $\Delta Q_2=0$, $\Delta Q_3=0$ y $\Delta Q_4=0$, la expresión (3.25) toma la forma presentada en (3.26).

$$\begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22} & B_{23} \\ 0 & B_{32} & B_{33} \end{bmatrix} \begin{bmatrix} \Delta V_1 \\ \Delta V_2 \\ \Delta V_3 \end{bmatrix} = \begin{bmatrix} \Delta Q_1 \ / \ V_1 \\ 0 \\ 0 \end{bmatrix} \qquad \textbf{(3.26)}$$

El objetivo de este desarrollo es encontrar la relación $\Delta V_1/\Delta Q_1$ independiente de los cambios ΔV_2 y ΔV_3, por tanto aplicando la *Inversión Parcial de Shipley*[17] tomando como pivote el elemento B_{33} de (3.26), se obtiene (3.27)

$$\begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & \left[B_{22} - \dfrac{B_{32}B_{23}}{B_{33}}\right] & -\dfrac{B_{23}}{B_{33}} \\ 0 & -\dfrac{B_{32}}{B_{33}} & -\dfrac{1}{B_{33}} \end{bmatrix} \begin{bmatrix} \Delta V_1 \\ \Delta V_2 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta Q_1 \ / \ V_1 \\ 0 \\ \Delta V_3 \end{bmatrix} \qquad \textbf{(3.27)}$$

renombrando términos resulta en

$$\begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22}^{*} & B_{23}^{*} \\ 0 & B_{32}^{*} & B_{33}^{*} \end{bmatrix} \begin{bmatrix} \Delta V_1 \\ \Delta V_2 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta Q_1 / |V_1| \\ 0 \\ \Delta V_3 \end{bmatrix} \tag{3.28}$$

En la expresión anterior se toma como pivote el elemento B_{22}^{*} y se obtiene (3.29)

$$\begin{bmatrix} \left[B_{11} - \frac{B_{21}B_{12}}{B_{22}^{*}}\right] & -\frac{B_{12}}{B_{22}^{*}} & -\frac{B_{12}B_{23}^{*}}{B_{22}^{*}} \\ -\frac{B_{21}}{B_{22}^{*}} & -\frac{1}{B_{22}^{*}} & -\frac{B_{23}^{*}}{B_{22}^{*}} \\ -\frac{B_{21}B_{32}^{*}}{B_{22}^{*}} & -\frac{B_{32}^{*}}{B_{22}^{*}} & \left[B_{33}^{*} - \frac{B_{32}^{*}B_{23}^{*}}{B_{22}^{*}}\right] \end{bmatrix} \begin{bmatrix} \Delta V_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta Q_1 / |V_1| \\ \Delta V_2 \\ \Delta V_3 \end{bmatrix} \tag{3.29}$$

De la expresión (3.29) desarrollando la primera ecuación se obtiene la relación (3.30).

$$\left[B_{11} - \frac{B_{21}B_{12}}{B_{22}^{*}}\right] \Delta V_1 = \Delta Q_1 \tag{3.30}$$

Por último, de (3.30) despejando el cambio ΔV_1 se obtiene (3.31), la cual representa la relación buscada.

$$\Delta V_1 = \left[B_{11} - \frac{B_{21}B_{12}}{B_{22}^{*}}\right]^{-1} \Delta Q_1 \tag{3.31}$$

El término $[B_{11} - B_{21} B_{12} \quad B_{22}^{*}]^{-1}$ representa el factor de sensitividad, a partir del cual se obtiene el cambio ΔV_1 requerido para ajustar V_1 y así mantener la generación de reactivos Q_1 dentro de sus límites. Analizando éste término se concluye que los factores de sensitividad para el cambio de voltaje nodal, respecto a un cambio de inyección de

potencia reactiva en el mismo nodo, se definen como el elemento diagonal de la inversa de la matriz [B"] del nodo correspondiente. De esta forma se concluye el desarrollo analítico para la obtención de los factores de sensitividad tipo $\Delta V_i / \Delta Q_i$ para nodos de voltaje controlado.

3.7 CONCLUSIONES

En este capítulo se han presentado las características de interacción entre la ambientación del simulador y los algoritmos. Se presentaron brevemente los métodos de Gauss-Seidel, Newton-Raphson y Desacoplado Rápido para el análisis de flujos de carga mismos que han sido implementados en el simulador digital interactivo. La importancia que representa la implementación de los tres algoritmos anteriores en el simulador interactivo es relevante, ya que permite al usuario no solo conocer los algoritmos sino contar con diversas herramientas de estado estable que le permitan solucionar casos de estudio con posibles problemas numéricos para encontrar solución, sensibilizándose de su comportamiento. Estas herramientas sirven de base para otro tipo de estudios como: cargabilidad de enlaces, análisis de fallas y uso de sensitividad para controlar el perfil de voltaje y reactivos, así como para evaluar la seguridad del sistema eléctrico de potencia.

CAPITULO 4

AMBIENTACION Y USO DEL SIMULADOR DIGITAL INTERACTIVO

En este capítulo se describe el ambiente de programación implementado en el simulador digital interactivo, así como la interacción lograda para con el usuario en el manejo de las diferentes opciones del mismo. Por medio de diversas corridas se demuestra la flexibilidad y bondades logradas en el simulador interactivo. Se describen a su vez los diversos problemas encontrados al intentar enlazar las aplicaciones implementadas cuando están desarrolladas en distintos lenguajes de programación, así como al mezclar el ambiente de programación Windows® con el ambiente DOS.

4.1 CARACTERISTICAS DEL SIMULADOR DIGITAL Y SU AMBIENTACION

El simulador interactivo fue diseñado para trabajar bajo ambiente Windows, los requerimientos de cómputo mínimos para su operación óptima incluyen Computadora Personal PC-386 Compatible con las siguientes características: 2 Mbytes de memoria RAM, coprocesador matemático no indispensable pero recomendable, monitor VGA y mouse. En software se recomienda emplear el sistema operativo MS-DOS versión 5.0 o posterior, así como el sistema Windows versión 3.1 ejecutado en modo extendido.

La motivación principal de la implementación de éste simulador interactivo fue el proporcionarle al usuario las herramientas fundamentales para el análisis de los SEP que le permitan la mayor interacción posible con los elementos y parámetros del sistema de estudio, buscando adicionalmente darle el mínimo de responsabilidades y cuidados innecesarios para su manejo. Uno de los puntos claves para lograr los objetivos planteados fue el presentar al usuario una secuencia ordenada en el desarrollo del estudio, mediante la activación y desactivación de opciones conforme avanza el proceso. Lo anterior permite, en cualquier etapa del estudio, retroceder a pasos anteriores o continuar con el estudio e incluso iniciarlo nuevamente. En todo momento se presenta

al usuario, por medio de ventanas de diálogo, la información mínima necesaria para proseguir con el estudio, empleando terminología y frases comúnmente utilizadas en el análisis de los SEP con lo cual se evita el abrumar al usuario con información confusa e irrelevante dentro del proceso. Desde la activación inicial en el uso del simulador el usuario encuentra habilitadas las mínimas opciones requeridas para comenzar el estudio. La Figura 4.1 muestra gráficamente el menú principal del módulo de flujos indicando tanto las opciones que han sido habilitadas, como aquellas que se encuentran deshabilitadas.

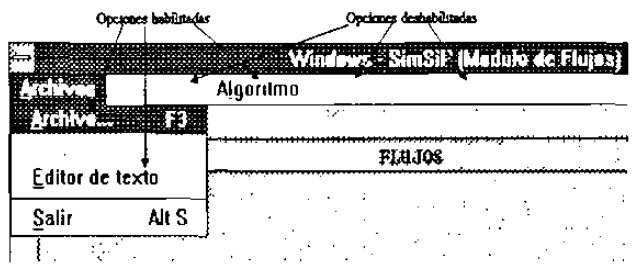


Fig. 4.1 *Menú principal inicial*

Como se observa en la Figura 4.1 el usuario solamente tiene la posibilidad de realizar cuatro acciones, las cuales son: seleccionar el caso de estudio por medio de un archivo de datos, elegir el algoritmo de flujos a emplear, la utilización del editor de texto para analizar un caso de estudio o para realizar cambios sustanciales en los datos base del mismo y por último, salir del simulador. La interacción con los diferentes elementos del sistema, así como con los parámetros que los representan, son un factor importante de flexibilidad en el uso y explotación del simulador.

4.2 INTERACCION CON ELEMENTOS Y PARAMETROS DEL SISTEMA

La interacción con elementos y parámetros del sistema es una de las ventajas primordiales que ofrece el simulador al usuario. Se tiene la posibilidad de conectar y desconectar líneas, transformadores y elementos en derivación, modificar taps de transformadores, valores de compensación de elementos en derivación, etc. Un problema latente en éste tipo de interacción es la gran cantidad de nombres y nemónicos

que es necesario memorizar para seleccionar los diferentes elementos del sistema. En el simulador digital se le ha dado solución a éste problema mostrándole al usuario, en todo momento, el conjunto de nombres de nodos, líneas, transformadores, etc., arreglados en listas que permiten una búsqueda sencilla y rápida para su selección; como se muestra en la Figura 4.2.

La opción "**Cambios**" es activada al seleccionar el archivo del sistema de estudio en el menú principal y presenta las diferentes alternativas para interactuar con los elementos y parámetros del sistema. La Figura 4.3 muestra las opciones del menú de cambios permitidos los cuales serán descritos en los apartados siguientes.

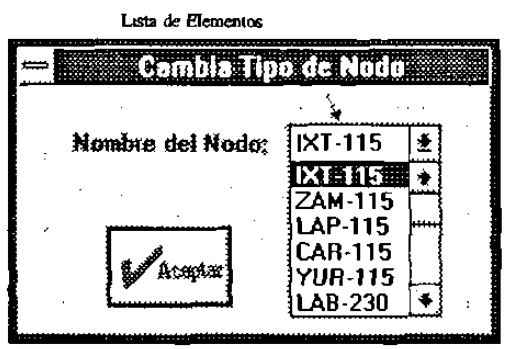


Fig. 4.2 *Lista de nombres de elementos.*

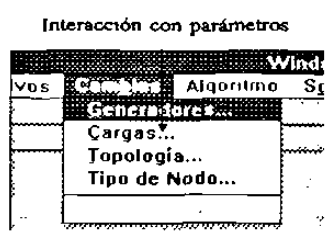


Fig. 4.3 *Menú Cambios*

4.2.1 CAMBIOS EN CONDICIONES DE OPERACION DE LOS GENERADORES

Esta opción permite interactuar sobre los nodos de generación del sistema, dando acceso a cambios en la potencia real generada, en los límites de generación de potencia

reactiva y en el valor del voltaje nodal especificado. La Figura 4.4 muestra la ventana de diálogo que permite seleccionar, tanto el nodo de generación, como el tipo de cambio deseado.

El aceptar la selección en la ventana de la Figura 4.4 lleva a un nuevo diálogo el cual muestra el valor actual del parámetro a cambiar y permite introducir el nuevo valor deseado. La forma correspondiente se muestra en la Figura 4.5

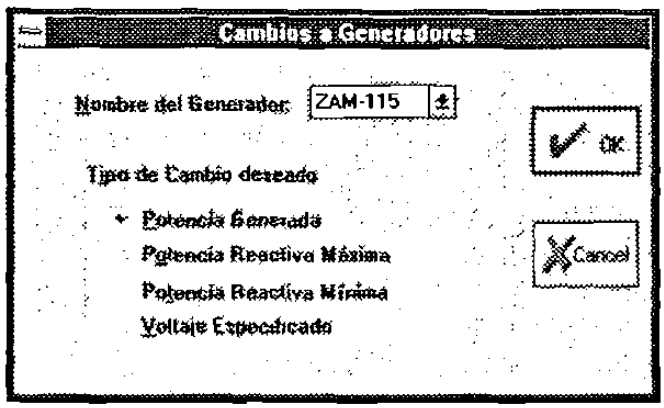


Fig. 4.4 *Cambios a Generadores*

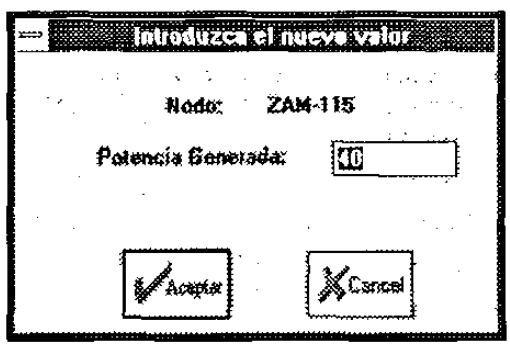


Fig. 4.5 *Acepta cambio de Generadores*

4.2.2 CAMBIOS A CARGAS ELECTRICAS NODALES

Esta opción permite interactuar con todos los nodos del sistema, dando oportunidad de cambiar la carga local instalada en cada uno de ellos; con la finalidad

de brindar mayor versatilidad al análisis del comportamiento de los SEP, se han incluido diferentes alternativas para efectuar cambios de carga: de un nodo a la vez, así como globalmente a todos los nodos del sistema. Los cambios pueden efectuarse, ya sea en valores de potencia real y reactiva, directamente, o bien en un porcentaje e incluso en el factor de potencia de la carga. La Figura 4.6 muestra la ventana de diálogo que permite seleccionar ésta diversidad de opciones.

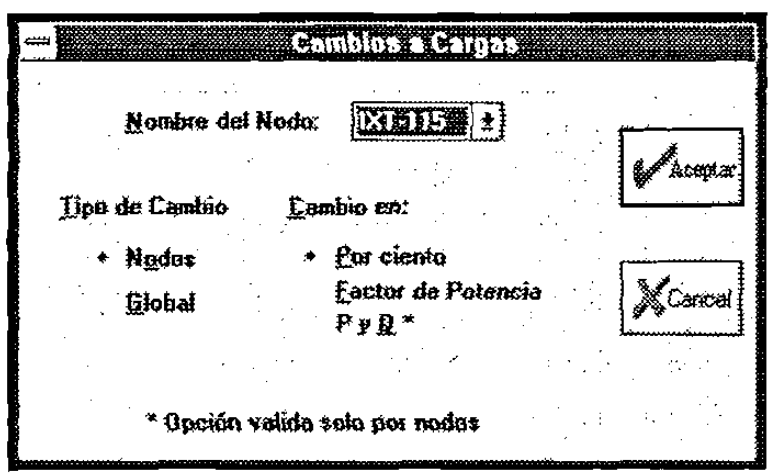


Fig. 4.6 *Cambios a cargas electricas nodales*

Una vez aceptada la selección y el tipo de cambio a realizar en la Figura 4.6, ésta lleva a un nuevo diálogo que mostrará los valores actuales de potencia a ser cambiados o bien por medio de una línea permite introducir el cambio deseado, como se muestra en las Figuras 4.7 y 4.8 respectivamente.

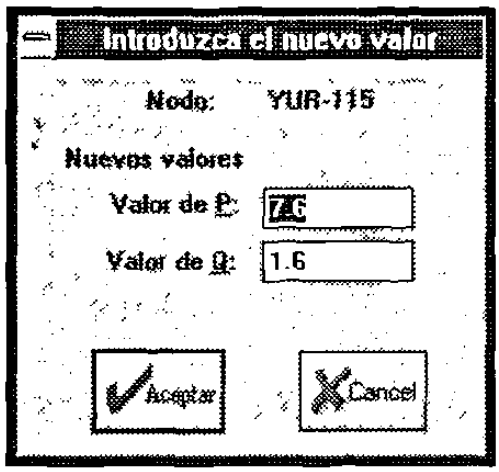


Fig. 4.7 *Cambio por nodo.*

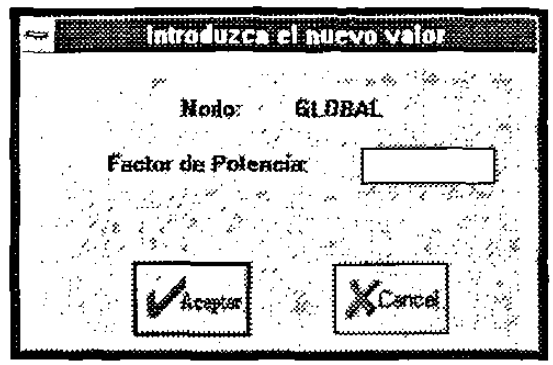


Fig. 4.8 *Cambio nodal global.*

4.2.3 CAMBIOS A TOPOLOGIA DE LA RED

Esta opción permite interactuar con todos los elementos modelados en la red eléctrica como son: líneas de transmisión, transformadores y elementos en derivación, dándole la posibilidad al usuario de conectar o desconectar cualquier elemento, cambiar sus parámetros, así como el modificar el valor de los taps de transformadores y de elementos en derivación. La Figura 4.9 muestra la ventana de diálogo para los cambios en Topología.

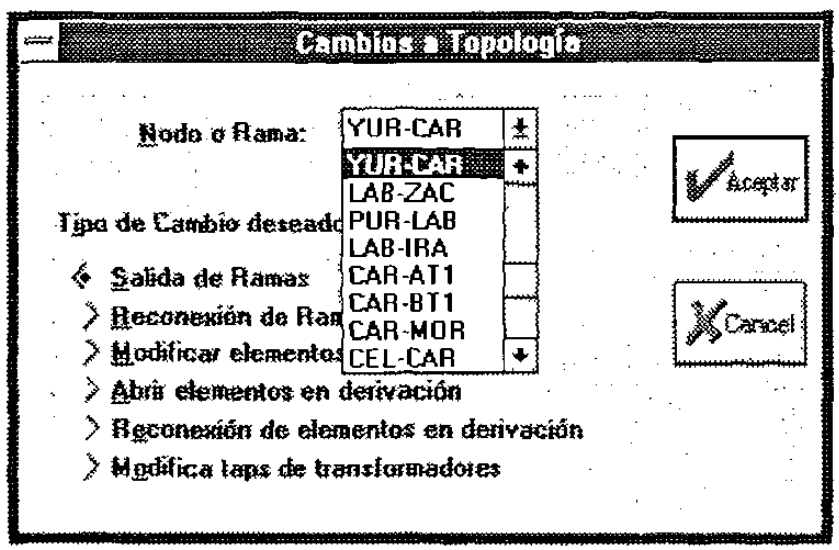


Fig. 4.9 *Cambios a Topología de la red*

La Figura 4.9 presenta características de interacción importantes, ya que debido al manejo de diferentes elementos como: líneas, transformadores y elementos en derivación, es necesario actualizar la lista de nombres de los elementos de acuerdo a cada selección del usuario. La rapidez con la cual es posible realizar estos cambios permite manejar las modificaciones de forma transparente al usuario y darle así una gran versatilidad en su uso. Seleccionar la opción de cambios de tap lleva a un nuevo diálogo en el cual se muestra el valor actual del tap y se introduce el cambio deseado. De la misma forma el seleccionar cambios a elementos en derivación lleva a una nueva interacción en la cual se muestra tanto el valor de la susceptancia en derivación como el tipo de la misma, inductivo o capacitivo, permitiendo cambiar su valor o bien el tipo de compensación. Las Figuras 4.10 y 4.11 muestran los diálogos correspondientes.

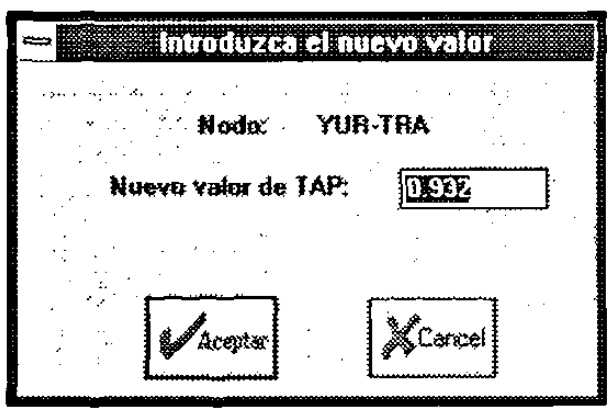


Fig. 4.10 *Cambio de taps en transformadores.*

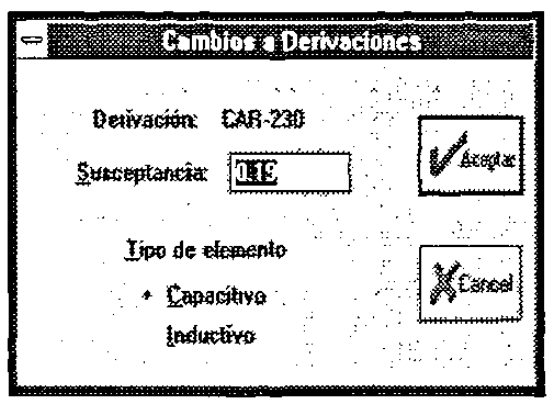


Fig. 4.11 *Cambio a elementos en derivacion*

4.2.4 CAMBIOS AL TIPO DE NODO

Esta opción permite interactuar con todos los nodos del sistema y cambiar el tipo de nodo previamente especificado, brindando con esto una mayor posibilidad de análisis del comportamiento de los SEP, así como un manejo alterno a problemas numéricos generados por límites físicos de generación, o violación de límites de operación segura y de calidad de servicio. La ventana de diálogo presentada en la Figura 4.2 permite seleccionar el nombre del nodo, esto a su vez conduce al diálogo mostrado en la Figura

4.12 donde se tiene el tipo de nodo actual y se introduce el cambio deseado.

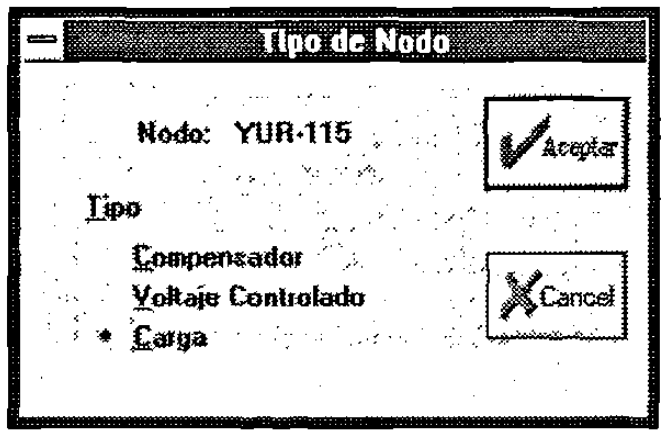


Fig. 4.12 *Cambio al tipo de nodo.*

En los apartados anteriores se ha mostrado las diferentes opciones con las que cuenta el usuario para la interacción con elementos y parámetros del sistema. La gran versatilidad en la selección de los elementos, así como en la introducción de cambios o modificaciones hacen que el simulador sea una herramienta útil y atractiva para el análisis de SEP. La selección de los algoritmos integrados en el simulador puede realizarse en forma rápida y sencilla, lo cual se describe a continuación.

4.3 SELECCION DE ALGORITMOS DE SOLUCION AL PROBLEMA DE FLUJOS

La opción algoritmo del menú principal, mostrado en la Figura 4.1, permite seleccionar entre los diferentes métodos numéricos, incluidos en el simulador, para el análisis de flujos, tales como: Newton-Raphson, Gauss-Seidel y Desacoplado Rápido. La Figura 4.13 muestra la ventana de diálogo que permite la selección del algoritmo deseado.

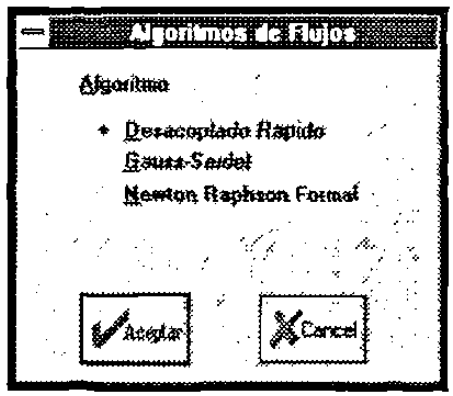


Fig. 4.13 *Algoritmos para solución al problema de flujos.*

4.4 PARAMETROS PARA CONTROLAR LA EJECUCION DEL PROCESO ITERATIVO

Esta opción le permite al usuario realizar dos tareas básicas, la solución directa del estudio de flujos, después de la cual son activadas las opciones de reportes, así como la especificación de constantes, tolerancias, iteraciones máximas, etc. La Figura 4.14 muestra la ventana de diálogo que permite al usuario proponer los parámetros y características requeridas en la solución.

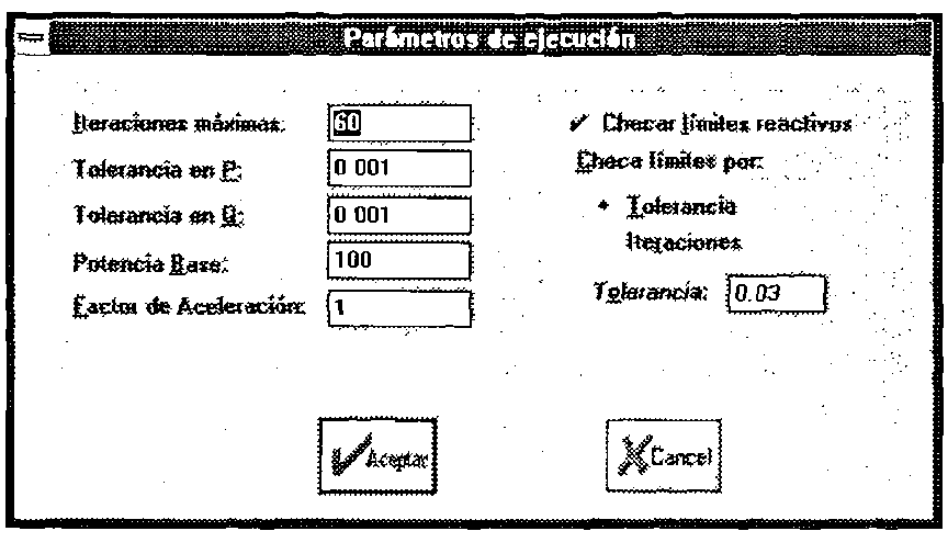


Fig. 4.14 *Parametros para controlar la ejecucion del proceso iterativo.*

Puede observarse en la Figura 4.14 que el usuario tiene la posibilidad de conjugar diferentes características para el control de la solución: cambiando iteraciones máximas,

tolerancias tanto al desajuste de potencia activa como al de potencia reactiva, la potencia base y el factor de aceleración. De la misma forma puede decidir si es conveniente activar los límites de generación de potencia reactiva en generadores y a partir de que iteración o tolerancia realizarla. La versatilidad que proporcionan estas opciones permiten al usuario tener un control más completo y efectivo de los algoritmos a fin de resolver posibles problemas de convergencia.

4.5 SELECCION DE REPORTE DE RESULTADOS

Los reportes disponibles, una vez lograda la convergencia del proceso iterativo, brindan al usuario la mayor información posible respecto de las condiciones de operación nodales y de flujos en el sistema; con este fin se incluyeron tres distintos tipos de reportes como son: reporte de flujos en elementos del sistema, reporte de condiciones nodales y reporte de generadores. Los tres reportes presentan diferente información del sistema con lo cual se complementa el análisis del mismo. El reporte de flujos, por ejemplo, presenta cada nodo del sistema con los elementos que lo unen a otros nodos, indicando el flujo de potencia real, la potencia reactiva, la potencia aparente, el tap en caso de ser transformador, además de un dato importante, las pérdidas en la línea.

La Figura 4.15 muestra el reporte de flujos para dos de los nodos del sistema de prueba de 14 nodos de la IEEE[18].

OWL Editor de Texto - C:\ASEP\Lineas.sal

Archivos Editar Buscar Bloque

Reporte de Flujos completo IEEE-14.FLU

De	1 IXT-115	MW	MVAR	MVA	TAP	1.060	0.0
	Slack	232.289	-16 892	232.903		Perdidas	
A	2 ZAM 115	156.730	20 404	158.052		4.294	7 253
A	5 YUR-115	75.560	3.512	75.641		2.763	6.085
De	2 ZAM-115	MW	MVAR	MVA	TAP	1.045	5.0
	Generador	40.000	42 405	58.294		Perdidas	
	Carga	21.700	12.700	25.143			
A	3 LAP-115	73.146	3.566	73.233		2.318	5.140
A	4 CAR-115	56.129	2.295	56.176		1.676	1.104
A	5 YUR-115	41.506	0.776	41 513		0.901	-0.872
A	1 IXT-115	-152.436	27.658	154.925		4.294	7.253

Fig. 4.15 *Reporte de flujos en elementos*

El reporte nodal proporciona las condiciones nodales de voltaje y ángulo en grados, potencia real en MW y la potencia reactiva de generación en MVARs y carga, así como los desajustes nodales logrados en la convergencia. La Figura 4.16 muestra un ejemplo de reporte nodal.

OWL Editor de Texto - c:\asep\voltajes.sol

Archivos Editar Buscar Bloque

Voltajes nodales sistema de prueba IEEE-14.FLU

Nodo	Tipo	Voltaje		Generacion		Carga		Desajuste	
		V	Ang	MW	MVAR	MW	MVAR	MW	MVAR
EXT-115	SLACK	1.060	0.0	232.3	-16.9	0.0	0.0		
ZAM 115	PV	1.045	-5.0	40.0	42.4	21.7	12.7	-0.045	
LAP-115	PV	1.010	-12.7	0.0	23.4	94.2	19.0	-0.049	
CAR-115	CARGA	1.019	-10.3	0.0	0.0	47.8	-3.9	-0.015	0.000
YUR 115	CARGA	1.020	-8.8	0.0	0.0	7.6	1.6	-0.025	-0.003
LAB-230	PV	1.070	-14.2	0.0	12.2	11.2	7.5	0.098	
CAR-0T1	CARGA	1.062	-13.4	0.0	0.0	0.0	0.0	-0.004	0.000

Fig. 4.16 *Reporte de condiciones nodales.*

El reporte de generadores muestra las condiciones de voltaje, potencia generada real y reactiva de los nodos de generación, así como los límites de generación de reactivos, permitiendo así una detección directa de posibles violaciones a los mismos. La Figura 4.17 muestra el reporte de generadores del sistema de prueba de 14 nodos del IEEE.

OWL Editor de Texto - c:\asep\volt_nod.sol

Archivos Editar Buscar Bloque

Reporte de Generadores sistema de prueba IEEE-14.FLU

Nodo	Voltaje		Generacion		Qmax	Qmin
	V	Ang	MW	MVAR	MVAR	MVAR
EXT 115	1.060	0.0	232.3	16.9	0.0	0.0
ZAM 115	1.045	-5.0	61.7	42.4	50.0	40.0
LAP-115	1.010	-12.7	94.2	23.4	40.0	0.0
LAB-230	1.070	-14.2	11.2	12.2	24.0	-6.0
CAR-0T2	1.090	13.4	0.0	17.3	24.0	-6.0

Fig. 4.17 *Reporte de generadores*

Se han presentado las diferentes opciones con las que cuenta actualmente el simulador interactivo, mostrando claramente la flexibilidad e interacción logradas. La interacción sobre los elementos y parámetros del sistema de estudio, contando con las listas de nombres de elementos, así como el seleccionar distintos controles de solución,

para los diferentes algoritmos implementados, son una muestra clara de la versatilidad y utilidad del simulador digital. Existen, como es natural, en este tipo de desarrollos ciertas restricciones en la utilización del simulador digital, algunas de ellas están relacionadas con el dimensionamiento de los programas para el manejo de SEP de gran escala, otras restricciones están relacionados con la problemática de interacción entre diferentes programas realizados en distintos lenguajes de programación. La descripción de éstas restricciones se realiza en el apartado siguiente.

4.6 RESTRICCIONES Y LIMITACIONES DEL SIMULADOR DIGITAL

La capacidad de memoria del simulador digital está limitada por los lenguajes de programación que se emplearon en el desarrollo de cada uno de los módulos que conforman el simulador. La mayoría de los lenguajes se encuentran restringidos al manejo en forma directa de solamente 640 Kbytes de memoria convencional. El tamaño máximo de los SEP a analizar dependerá de las características topológicas de la red y si ésta es o no mallada. Esta barrera de memoria puede ser franqueable mediante el empleo adicional de herramientas de software como el "*DOS-Extender de Phar Lap*" el cual permite accesar la memoria completa instalada en la computadora, ya sea de manera directa o indirecta. Este aspecto de manejo de memoria juega un papel importante en la interacción a lograr del simulador digital, debido principalmente a la necesidad de interactuar con los elementos y parámetros del sistema ya que ésto involucra el cargar a memoria los datos completos del sistema o bien bloques de ellos para su manejo.

El objetivo principal en el desarrollo del simulador digital fue dar al usuario la mayor interacción posible, la poca flexibilidad del sistema operativo DOS para manejar diversas aplicaciones simultáneamente, así como la incompatibilidad entre diversos lenguajes para intercomunicarse, limitó el desarrollo completo de las perspectivas de interacción planeadas para la implementación del simulador. La mezcla de los ambientes Windows y DOS provocó varios problemas, uno de ellos el switcheo entre el ambiente

gráfico de la ambientación en Windows con el ambiente DOS de los algoritmos en modo texto. Este problema fue resuelto con una utilería propia de Windows para el manejo de aplicaciones no-windows, denominada "*Program Information Files* (PIF)", la cual traslada la salida de programas de ambiente DOS a una ventana propia de Windows, solución que aún siendo aceptable presenta un retraso apreciable en la visualización de resultados de los algoritmos numéricos, limitando así, una posible interacción directa del usuario con los algoritmos.

Con la finalidad de compensar las deficiencias del sistema operativo DOS en el manejo simultáneo de aplicaciones, el ambiente Windows introdujo el concepto de multitareas lo cual ha traído ventajas significativas en el empleo e intercomunicación de diferentes aplicaciones a un tiempo. El manejo de multitareas causó conflictos en el desarrollo del simulador, ya que en el momento en que la ambientación ejecuta los programas de algoritmos numéricos Windows los toma como una nueva aplicación que operará simultáneamente con la ambientación, el problema estriba en que esta última recibe la información de que el algoritmo término su ejecución antes de que realmente finalice, lo cual ocasiona problemas al usuario ya que buscará resultados que aún no existen. El manejo de esta restricción ha quedado como responsabilidad del usuario, el cual fácilmente puede darse cuenta si verdaderamente terminó la ejecución del algoritmo para continuar.

4.7 PROGRAMA DE DEMOSTRACION

Ha sido incluido en la parte posterior de este trabajo un diskette de computadora, el cual contiene un programa de demostracion del ambiente generado. Los requerimientos mínimos para el manejo óptimo de este programa son los siguientes: Computadora personal PC-386, monitor VGA, mouse, 2 MBytes de memoria RAM, así como el ambiente Windows versión 3.1 o mayores, ejecutado en modo extendido. El programa incluido no contiene los algoritmos numéricos implementados, solamente simula su ejecución con el fin de mostrar la interacción del ambiente.

4.8 CONCLUSIONES

No es fácil expresar en palabras la verdadera versatilidad del simulador interactivo, la cual solamente se aprecia en su total plenitud con el uso del simulador. Aún así se han tratado de dejar claras las ventajas que proporciona el simulador al usuario. La gran flexibilidad en su manejo, así como la diversidad de opciones permitidas para interactuar con los elementos y parámetros del sistema, el contar con diferentes algoritmos en el mismo entorno, ya que sólo basta con seleccionar el más adecuado o el deseado para el análisis requerido, el conjugar diversas características de solución de los algoritmos que ayuden a una mejor comprensión del comportamiento de los mismos o a resolver sistemas de difícil convergencia, todas estas facilidades que le brinda el simulador al usuario no dejan duda de la flexibilidad y bondades logradas en el mismo.

Las restricciones y limitaciones encontradas en el manejo del simulador digital son mínimas, algunas de ellas pueden ser resueltas mediante el empleo de software adicional, para otras será necesario adecuar los avances de nuevo software que reduzcan estas limitaciones.

CAPITULO 5
MODULO PARA EL ANALISIS DE CARGABILIDAD

En este capítulo se presentan los aspectos básicos involucrados en el análisis de cargabilidad de líneas de transmisión, con énfasis en la influencia que en ésta determinación tienen la capacidad de corto circuito (CCC) de los SEP entre los cuales se conectará la línea, y la capacidad de soporte de reactivos de ambos sistemas. Se presenta un algoritmo numérico para el análisis de cargabilidad, en base a un estudio de flujos modificado, así como su inclusión como módulo adicional al simulador digital mostrando la interacción que se logra en esta nueva aplicación.

5.1 INTRODUCCION

En la planeación y la operación de los sistemas eléctricos de potencia se requiere del análisis de cada uno de sus elementos en forma separada, verificando la capacidad, disponibilidad y confiabilidad de los mismos, así como del sistema en su conjunto. El crecimiento de los SEP en base a estudios de aumento de demanda, el diseñar enlaces ya sea para interconectar diferentes sistemas o áreas de un mismo sistema, así como el aumentar la capacidad de transmisión de enlaces son aspectos básicos de la planeación. En el ámbito de operación la distribución de los flujos en el sistema, así como la cantidad de potencia a proporcionar por cada generador, son factores que requieren un análisis detallado para operar el sistema de una manera confiable, economica y segura. Operar el sistema con un margen de seguridad ante contingencias, así como el proporcionar un servicio de calidad dentro de rangos aceptables, son factores de análisis fundamentales para lo cual el estudio de *cargabilidad* es de gran relevancia.

5.2 CARGABILIDAD DE ENLACES

El análisis de cargabilidad permite conocer la capacidad máxima de transmisión de potencia a través de una línea de transmision bajo las restricciones siguientes: límite térmico, caída de tension máxima permisible y por último el margen de estabilidad

establecido para la operación segura del sistema. Para sistemas de potencia robustos con capacidad de corto circuito arriba de los 15,000 MVA, operados incluso a niveles de tensión de extra y ultra alto voltaje (345 KV, 700 KV, 1200 KV) fue desarrollada[27,28,29] una *curva característica de cargabilidad*; mostrada en la Figura 5.1. En esta curva se muestra la capacidad máxima permisible de transmisión de potencia, a través de una línea de transmisión, en función de su longitud y en base a los límites establecidos.

La curva de cargabilidad ha sido separada en tres regiones, la primera enmarca como factor limitante para la transmisión la capacidad térmica; la segunda región señala como factor limitante la caída de voltaje en la línea, observándose lo anterior para longitudes menores de 320 Km (aproximadamente 200 millas). La tercera sección para líneas de mayor longitud como factor limitante lo establece el margen de estabilidad. El límite térmico de transmisión difícilmente es rebasado en sistemas débiles[14], en cambio para sistemas robustos resulta un factor limitante en líneas de longitud muy corta de alrededor de 80 Km (50 millas). En los sistemas eléctricos de Centroamérica caracterizados por formar SEP débiles[14] ciertas líneas de 115 KV están limitadas por el factor térmico, debido a que son líneas de una longitud muy corta y forman parte de redes malladas.

La seleccion de los diferentes límites de transmisión se basa en los criterios siguientes:

A. *Límite térmico*

Está dado por la capacidad de la línea para transmitir corriente a un nivel tal que el calentamiento producido por efecto Joule, no distorsione las características de elasticidad y rigidez mecánicas del conductor. Factores como el viento y la temperatura ambiente juegan un papel importante en la determinacion de éste límite.

B. *Límite por caída de tensión*

Está dado por la caída de tensión máxima permisible en la línea, de tal forma que

el voltaje en el nodo de recepción se mantenga dentro de un rango de calidad de servicio adecuado. La capacidad de soporte de reactivos en ambos extremos de la línea resulta un factor importante en la determinación de éste límite. Un límite del 5% en caída de voltaje en el ámbito de planeación es aceptable; en cambio este límite resulta muy restrictivo para el análisis de operación, requiriendo un margen más amplio.

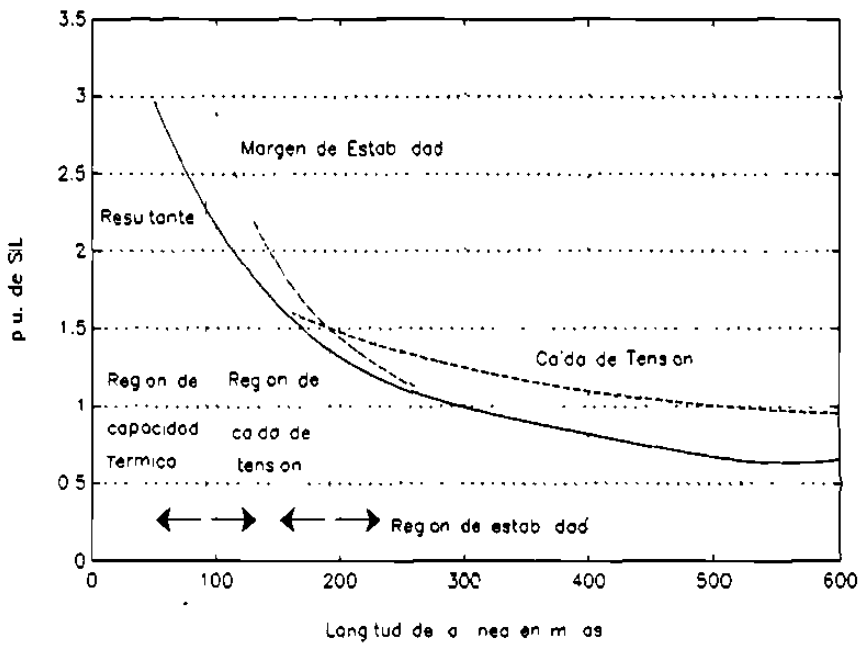


Fig. 5.1 *Curva de cargabilidad de R D Dunlop.*

C. *Límite de estabilidad*

Este límite está dado por un margen de seguridad tal que permita al sistema soportar contingencias sin llegar a colapsarse, éste margen es conocido como margen de estabilidad (ME) y se establece en base a una potencia generada de operación P_o, especificada normalmente alrededor de un 70% de la potencia máxima disponible, lo cual representa un ME del 30% permitiendo una separación angular máxima entre máquinas equivalentes de un extremo y otro de la línea de 44 grados eléctricos. La expresión (5.1) presenta el cálculo del margen de estabilidad bajo una potencia de operación especificada. De la misma forma (5.2) muestra el cálculo de la desviación

angular máxima entre máquinas, en base a la potencia de operación P_o especificada.

$$ME\% = \frac{P_{max} - P_o}{P_{max}} X\ 100 \qquad (5.1)$$

$$\delta = Sen^{-1}\left[\frac{P_o\ X_{eq}}{V_E\ V_R}\right] \qquad (5.2)$$

D. *Circuito Equivalente para Análisis de Cargabilidad.*

El modelo utilizado para el análisis de cargabilidad de una línea de transmisión que unirá dos sistemas aislados o áreas de un mismo sistema físicamente aisladas entre sí, será el presentado en la Figura 5.2, suponiendo un sistema balanceado.

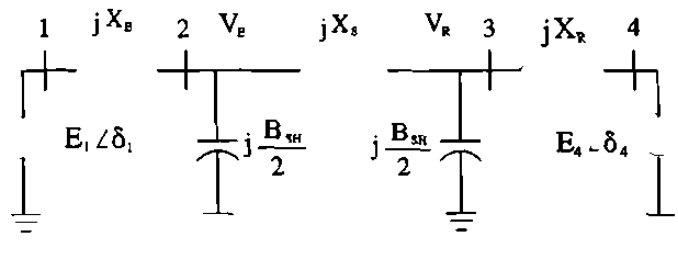


Fig. 5.2 *Sistema de prueba para análisis de cargabilidad*

Como se observa en la Figura 5.2 los sistemas a interconectar han sido reemplazados por su equivalente Thevenin; obtenido en base a la capacidad de corto circuito de cada sistema, visto desde los nodos de envío (I) y recepción (R) entre los cuales se conectará la línea. Las magnitudes de los voltajes $E_1\angle\delta_1$ y $E_4\angle\delta_4$ normalmente son especificados a valores de $E_1=1.0$ y $E_4=0.95$, éstos voltajes son indicativos de la capacidad de soporte de reactivos de cada sistema. El ángulo de una de las máquinas equivalentes δ_1 ó δ_4 se selecciona como referencia, asignándole un valor de cero grados; para el caso presentado se selecciona el nodo 4 como referencia, así $\delta_4=0°$. Los parámetros X_F y X_R representan las reactancias de los equivalentes Thévenin de envío

y recepción respectivamente. Los parámetros R_S, X_S y $B_{SH}/2$ son los correspondientes al modelo π equivalente de la línea de transmisión.

5.3 ALGORITMO PARA ANALISIS DE CARGABILIDAD

El algoritmo de *cargabilidad* implementado parte de un estudio de flujos de carga convencional adaptado específicamente para acomodar las particularidades del problema no-lineal. Haciendo referencia a la Figura 5.2, se define el nodo 1 como nodo de voltaje controlado con magnitud $E_1=1.0$, el nodo 4 será el compensador con valores de magnitud $E_4=0.95$ y ángulo $\delta_4=0°$, el nodo de recepción (R) se define como nodo de carga y por último el nodo de envío (t) será un nodo de carga con su voltaje controlado en la unidad, o sea $V_F=1.0$, lo cual se logra por medio del voltaje del generador 1 y empleando factores de sensitividad.

5.3.1 FACTOR DE SENSITIVIDAD

El método de flujos de carga empleado es el Desacoplado Rápido (DR)[30], el cual ha sido modificado para mantener la magnitud del voltaje del nodo de envío $V_E=1.0$, utilizando factores de sensitividad. El análisis de sensitividad para el modelo Q-V realizado en la sección 3.5 del capítulo 3 se tomará como base para éste desarrollo. Del modelo de potencia reactiva para el método DR presentado en (3.25) y esquematizado en la Figura 5.3, interesa obtener la sensitividad $\Delta V_1/\Delta V_2$ ya que dada la solución para ΔV_2 en una iteración del estudio de flujos se debe determinar el cambio ΔV_1 necesario para mantener V_2 en un valor unitario; en base al cambio ΔV_2 requerido.

Una vez conocido el cambio ΔV_2, se plantea el modelo incremental reactivo en el cual se desea variar ΔV_1 por medio de la inyeccion de potencia reactiva ΔQ_1 en el nodo 1, tal que anule el cambio ΔV_2 producido en la iteración actual del estudio de flujos, manteniendo así V_2 en un valor unitario. Bajo éstas condiciones y estableciendo que en la solución se tendrá inyeccion cero en los nodos 2 y 3, o sea $\Delta Q_2=0$ y $\Delta Q_3=0$, la expresión (3.25) toma la forma presentada en (3.26). El objetivo de este desarrollo

es encontrar la relación $\Delta V_1/\Delta V_2$ independiente de los cambios ΔV_3 y ΔQ_1, por tanto aplicando *Inversión Parcial de Shipley*[17] y tomando como pivote al elemento B_{33} de (3.26) se obtiene (3.27).

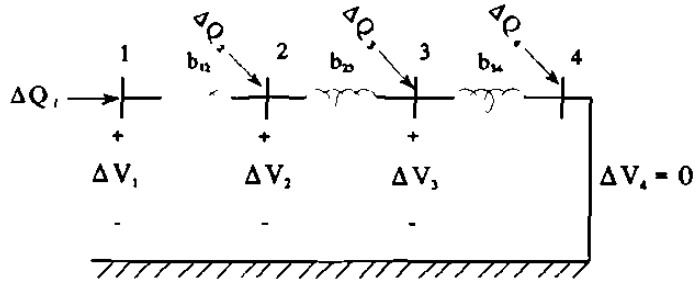


Fig. 5.3 *Modelo reactivo para el análisis de sensitividad*

De la expresión (3.27) desarrollando la segunda ecuación se obtiene la relación (5.3).

$$B_{21}\Delta V_1 + \left[B_{22} - \frac{B_{32}B_{23}}{B_{33}}\right]\Delta V_2 = 0 \qquad \textbf{(5.3)}$$

Por último despejando el cambio ΔV_1 se obtiene

$$\Delta V_1 = -\frac{\left(B_{22} - \dfrac{B_{23}B_{32}}{B_{33}}\right)}{B_{21}}\Delta V_2 \qquad \textbf{(5.4)}$$

El término ($-[B_{22} - B_{23}\, B_{32} / B_{33}]\quad B_{21}$) de (5.4) representa el factor de sensitividad, con el cual se calcula el cambio ΔV_1 para ajustar el voltaje V_1 y así mantener la magnitud de V_2 en la unidad. El cambio ΔV_3 provocado al voltaje V_3 puede obtenerse a partir de la tercer ecuación de la relación (3.27), o bien dejar que éste se ajuste por medio del proceso iterativo del estudio de flujos.

El algoritmo mostrado presenta buenas características, con una convergencia en 4 o 5 iteraciones lo cual permite obtener rápidamente puntos de la curva de cargabilidad para ser mostradas gráficamente, además permite una fácil y rápida implementación. La Figura 5.4 muestra el diagrama de flujo correspondiente para la modificación del algoritmo de flujos DR para el análisis de cargabilidad, empleando sensitividad.

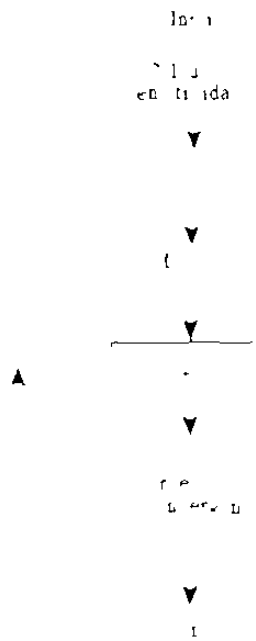


Fig. 5.4 *Diagrama de modificacion al metodo D R para analisis de cargabilidad*

5.4 DESARROLLO DEL ANALISIS DE CARGABILIDAD

El análisis de cargabilidad se inicia con el cálculo de los parámetros de la línea de transmisión, en base a su longitud y el nivel de tensión al cual operará. Se obtienen las reactancias de los equivalentes Thévenin de ambos sistemas de acuerdo a la capacidad de cortocircuito establecida, tanto en el extremo de envío como en el extremo de recepción. Se establecen los límites de operación deseados, respecto a margen de estabilidad y máxima caída de tensión en la línea, como se muestra en la Figura 5.5.

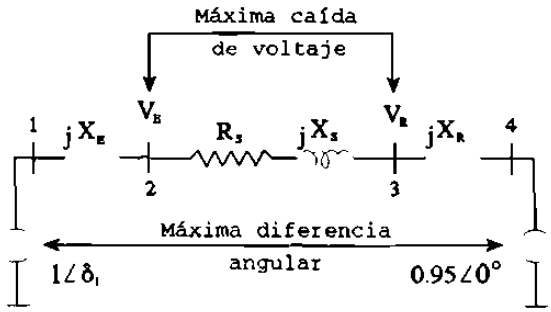


Fig. 5.5 *Límites de transmisión para estudio de cargabilidad.*

A. *Procedimiento para máxima caída de tensión.*

Se inicia el estudio asignando una potencia de generación en el nodo 1, la cual será absorbida por el compensador o generador conectado en el nodo 4. Se realiza la corrida de flujos, usando el factor de sensitividad para mantener $V_E = 1.0$, y del resultado obtenido se compara el voltaje en el nodo de recepción V_R, con el valor permitido por la caída de tensión especificada ($1.0 - \Delta V_{esp}$). Si la caída de tensión no ha sido excedida, se incrementa la potencia generada en el nodo 1 y se realiza una nueva corrida de flujos. El valor de potencia generada para el cual se alcanzó el límite de caída de tensión permisible representa el valor de potencia máxima de operación dado por esta restricción.

B. *Procedimiento para máxima diferencia angular.*

Al igual que en el procedimiento anterior, se inicia con una potencia de generación en el nodo 1. Se realiza la corrida de flujos, y del resultado obtenido se compara el ángulo δ_1, correspondiente al nodo 1, con el valor máximo especificado por el margen de estabilidad. Si el ángulo δ_1 no ha excedido el valor límite especificado, se incrementa la potencia generada en el nodo 1 y se realiza otra corrida de flujos. El valor de potencia generada para el cual se alcanzó el límite de estabilidad, representa el valor de potencia máxima de operación dado este límite.

De los valores de potencia de transmisión máxima obtenidos tanto por límite de caída de tensión como por el límite de estabilidad, se toma el menor de ellos como el límite de cargabilidad de la línea de transmisión para la longitud y nivel de tensión de operación especificados, y para los límites de operación establecidos.

5.5 PRUEBAS DE AMBIENTACION PARA EL ANALISIS DE CARGABILIDAD

El módulo de análisis de *Cargabilidad* implementado ha sido adicionado al simulador digital como una nueva aplicación. Tal y como se muestra en la Figura 5.6 el simulador digital fue integrado al ambiente Windows como un grupo de aplicaciones, denominado **Windows-SimSiP**, conteniendo los módulos de flujos de carga y de análisis de cargabilidad.

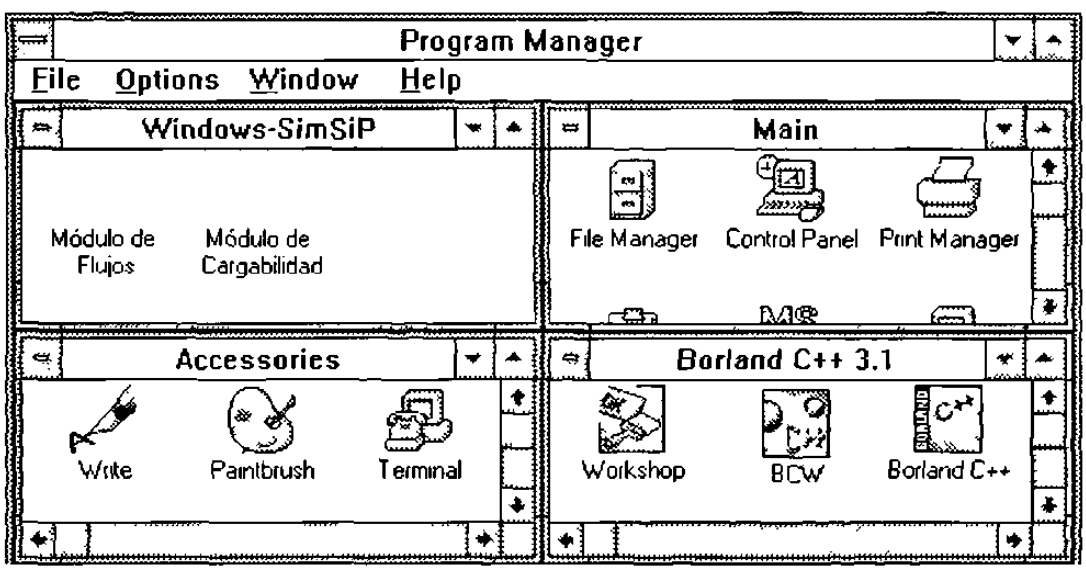


Fig. 5.6 *Integracion del Simulador Digital Interactivo como un grupo de aplicaciones hacia Windows*

Ambos módulos incluidos en el simulador trabajan en forma independiente, pero comparten internamente módulos adicionales como el editor de texto, así como las herramientas propias del ambiente Windows.

5.5.1 AMBIENTACION

El módulo de cargabilidad cuenta con diferentes opciones para la interacción con el usuario, tales como: selección del archivo de datos, cambio de condiciones de operación en generadores, etc. La Figura 5.7 muestra el menú principal de opciones del módulo de cargabilidad, mismas que serán descritas en los apartados siguientes.

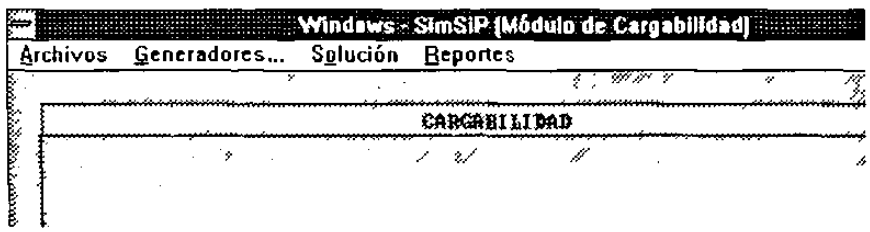


Fig. 5.7 *Menu principal, modulo de cargabilidad*

5.5.2 MANEJO DE CARACTERISTICAS DE GENERADORES

Esta opción permite interactuar con los generadores del sistema, dando acceso a cambios en la potencia real generada, por medio de la cual se busca la potencia de transmisión máxima en la línea de transmisión; de la misma forma se tiene acceso a la especificación de límites de generación de potencia reactiva, así como al cambio del voltaje especificado en los generadores. La Figura 5.8 muestra la ventana de diálogo que permite seleccionar tanto el nodo de generación como el tipo de cambio deseado.

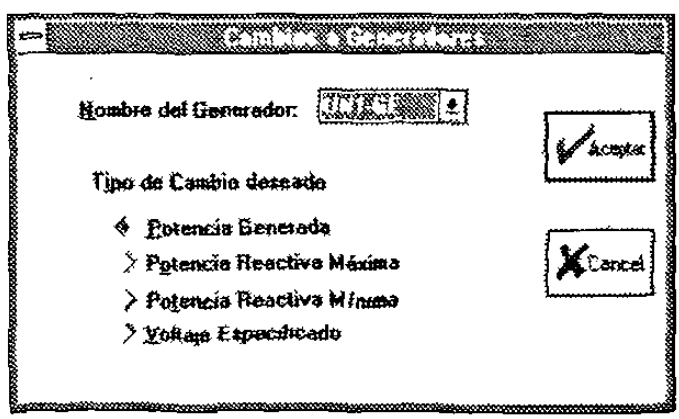


Fig. 5.8 *Cambios a Generadores*

Aceptar la selección en la ventana de diálogo de la Figura 5.8 lleva a un nuevo diálogo el cual muestra el valor actual del parámetro a cambiar permitiendo introducir el nuevo valor deseado, el diálogo correspondiente se muestra en la Figura 5.9.

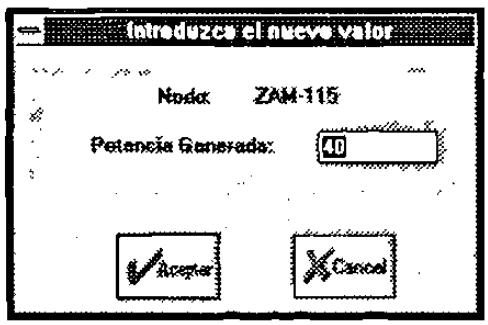


Fig. 5.9 *Acepta cambio de Generadores*

5.6 ALGORITMO DE SOLUCION AL PROBLEMA DE CARGABILIDAD

Esta opción le permite al usuario realizar dos tareas básicamente, como son: la solución directa del estudio de cargabilidad después de la cual son activadas las opciones de reportes, así como especificar los parámetros de solución deseados como: tolerancias, iteraciones máximas, etc. La Figura 5.10 muestra la ventana de diálogo que permite al usuario dar los parámetros y características de solución requeridas.

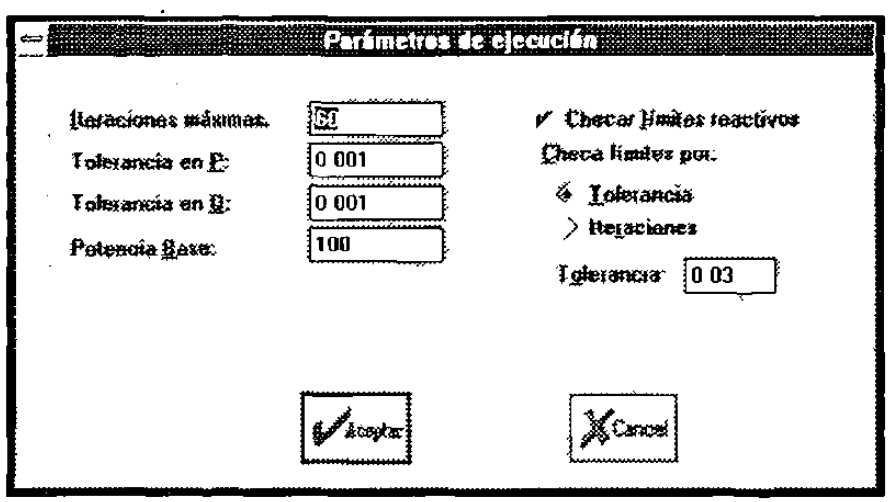


Fig. 5.10 *Parametros de ejecucion*

Como puede observarse en la Figura 5.10 el usuario tiene la posibilidad de conjugar diferentes características de la solución cambiando iteraciones máximas, las tolerancias tanto real como imaginaria y la potencia base. De la misma forma puede decidir si es conveniente checar los límites de generación de potencia reactiva y a partir de qué iteración o tolerancia se debe realizar ésta prueba. La versatilidad que proporcionan estas opciones le permiten al usuario realizar un análisis más completo sobre condiciones extremas de cargabilidad en la línea de transmisión, estudiar problemas de robustez en los sistemas de envío y recepción o bien problemas de capacidad de soporte de reactivos.

5.7 SELECCION DEL TIPO DE REPORTE

Los reportes proporcionados por el algoritmo de cargabilidad son esencialmente los mismos que los presentados por los algoritmos de flujos, como son: reporte de flujos completo, reporte nodal y reporte de generadores. El reporte de flujos es el que proporciona la información necesaria para el análisis de cargabilidad, éste reporte contiene la información del ángulo nodal δ_1 y de la magnitud de voltaje en el nodo de recepción V_R los cuales son requeridos para checar los límites establecidos. En la Figura 5.11 se muestra el reporte de flujos para el análisis de cargabilidad, señalar los valores básicos a observar para comprobar límites de caída de tensión en el nodo de recepción y el ángulo nodal en el generador 1, o sea V_R y δ_1 respectivamente.

El reporte de flujos en la Figura 5.11 muestra la potencia máxima de transmisión, para una línea de 230 KV con una longitud de 300 Km, considerando una caída de tensión máxima permisible del 5%; con el voltaje en el nodo de recepción de V_R=0.95 pu, el ángulo del nodo 1 tiene un valor de δ_1=19.4° lo cual lo mantiene dentro de límites para un ME=30% establecido.

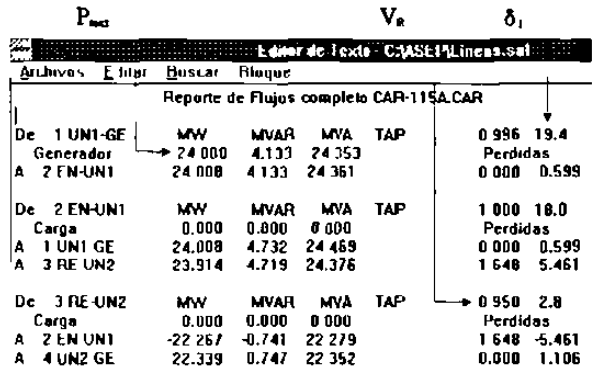


Fig. 5.11 *Reporte de flujos para análisis de cargabilidad.*

Finalmente mediante el análisis de una línea de 230 KV conectada a dos sistemas con capacidades de cortocircuito de 1,000 y 1,500 MVA en envío y recepción respectivamente, se obtuvieron para distintas longitudes de la línea los valores de potencia máxima de transmisión, los cuales en forma separada fueron graficados para obtener la curva de cargabilidad mostrada en la Figura 5.12, la cual muestra la degradación que presenta la cargabilidad de la línea al conectarse a SEP débiles; respecto a la curva presentada en la Figura 5.1 para sistemas robustos.

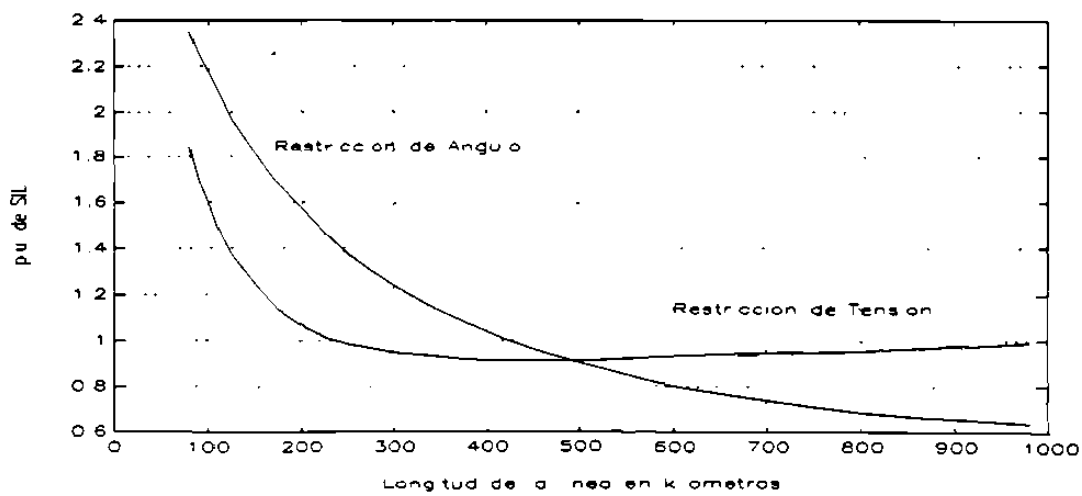


Fig. 5.12 *Curva de cargabilidad para una linea de 230 KV*

5.8 CONCLUSIONES

Se han presentado las diferentes opciones con las que cuenta el módulo de análisis de cargabilidad implementado en el simulador digital interactivo, el manejo y modificación de parámetros es esencial para el análisis de cargabilidad ya que permite de una forma rápida y sencilla buscar la potencia de transmisión máxima para los límites establecidos. Problemas de soporte de reactivos en los sistemas de envío y recepción pueden ser detectados automáticamente por el algoritmo, brindándole al usuario alternativas numéricas de solución.

Se presentaron dos aspectos relevantes, uno de ellos la implementación del algoritmo de análisis de cargabilidad, estudio que es importante tanto en el ámbito de planeación de SEP como en su operación. Se mostró un desarrollo eficiente y fundamentado del algoritmo demostrando la facilidad de implementación, presentándose además los aspectos relevantes de éste análisis. Otro punto importante es la relevancia que tiene la programación por módulos dentro del simulador digital, la flexibilidad que representa para la integración de nuevos módulos; teniendo acceso a herramientas de uso común las diferentes aplicaciones. Adicionalmente la interacción que se logró en el nuevo módulo demuestra que éste tipo de programación es una excelente alternativa para la implementación é inserción de nuevas aplicaciones al simulador.

CAPITULO 6
CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

En el presente trabajo se ha presentado el desarrollo de un *Simulador Digital Interactivo* para el análisis de SEP en estado estable mediante el empleo de herramientas de programación modernas. Por medio de estas herramientas fue posible generar, dentro del simulador digital, un ambiente interactivo amigable al usuario, con gran flexibilidad y de fácil manejo, para brindar al usuario una interacción plena con los algoritmos, así como con elementos y parámetros del SEP. Se permite una selección directa de los diferentes algoritmos de análisis y el empleo de distintos dispositivos entrada/salida habilitados en forma interna. Adicionalmente, se integró un sistema completo de ayuda en línea dentro de cada uno de los módulos del simulador digital, lo cual proporciona al usuario una mayor flexibilidad en su manejo.

La investigación realizada para el desarrollo del simulador digital interactivo comprendió la consideracion y el análisis de diversos aspectos relevantes que llevaron a las siguientes conclusiones:

- La simulación digital de los SEP, en las diferentes fases de planeación, operación y control, es un factor determinante para la operación segura y con calidad de servicio, así como para permitir un crecimiento sostenido y apropiado del mismo.

- La utilizacion en línea o fuera de línea del simulador digital define las características de software y hardware a utilizar en su implementación, así como los modelos de los elementos del SEP y los algoritmos numéricos a implementar.

- Desde el punto de vista de hardware, las estaciones de trabajo y las computadoras personales representan a las plataformas de desarrollo con mayor accesibilidad, tanto por su costo como por sus capacidades gráficas, de memoria, velocidad de procesamiento, etc.

- El empleo de algoritmos numéricos robustos garantiza una solución confiable ante diferentes condiciones de operación, sin embargo resulta necesario incluir algoritmos o procedimientos auxiliares para resolver casos de difícil convergencia.

- La flexibilidad e interacción brindada por un simulador digital representan factores determinantes de la aplicación real, apoyados en el buen desempeño y funcionamiento de los algoritmos numéricos usados.

- El desarrollo de la ambientación del simulador digital basado en el manejo de la programación orientada a objetos permitió dar al usuario gran flexibilidad en su manejo.

- Es importante presentar listas al usuario, con los nombres de los elementos del SEP, que permitan una selección directa de los mismos, evitando la necesidad de memorizar diferentes nombres y claves del sistema.

- La selección directa de elementos y opciones en el simulador digital permite al usuario analizar diversas condiciones de operación del SEP de una forma rápida y sencilla.

- La activación de diferentes algoritmos para el estudio de flujos de carga permite contar con diferentes alternativas para la solucion de casos con problemas de convergencia.

- La variedad de reportes de resultados con que cuenta el simulador digital permite conocer en forma detallada la condición de operación que se analiza.

- La integración del módulo de cargabilidad muestra la posibilidad de incluir nuevas aplicaciones al simulador digital en forma rápida y sencilla.

- El sistema de ayuda en línea implementado incrementa la flexibilidad y facilidad de manejo del simulador.

6.2 APORTACIONES

En el presente trabajo se han establecido las bases necesarias para el desarrollo de una ambientación interactiva de los algoritmos numéricos utilizados en el análisis en estado estable de los SEP, sin importar el lenguaje de programación en que sean desarrollados. Se aprovechan así los desarrollos generados con anterioridad y con los cuales se tiene experiencia, así como la característica de haber sido probados ampliamente.

Se generó la estructura básica modular del simulador digital interactivo mediante la implementación de algoritmos de flujos de potencia y cargabilidad. El desarrollo modular permite insertar nuevas aplicaciones, en módulos independientes que pueden interactuar con los ya existentes, integrando un simulador de gran capacidad e importancia para el análisis de SEP; tanto en estado estable en su forma presente, como en estado transitorio en un futuro cercano.

La aportación principal de este trabajo es el desarrollo y la implementación del simulador digital, el cual brinda al usuario flexibilidad en la interacción y permite realizar el análisis de los SEP de una forma directa y amigable; motivando al uso más frecuente del simulador. La posibilidad de consultar y modificar los elementos y parámetros del sistema, así como el especificar distintas características de solucion a los algoritmos le

dan pleno control al usuario, siendo lo más importante que todo se encuentra integrado en un mismo ambiente. Ningún tipo de selección para la entrada/salida o de edición de datos requerirá que el usuario abandone el ambiente.

6.3 RECOMENDACIONES

En el Programa Doctoral se ha desarrollado una gran cantidad de aplicaciones para el análisis de SEP, tanto en estado estable como en estado transitorio, la recomendación en este sentido es que sean integradas al simulador digital, con lo cual se logran dos objetivos: principalmente hacer más útiles y amigables las implementaciones para los alumnos y profesores del Programa, así como el extender las opciones del simulador digital con una mayor diversidad de aplicaciones, las cuales lo convertirían en una herramienta versatil y poderosa de análisis.

Se requiere complementar el simulador con un ambiente de manejo gráfico que permita el diseño de diagramas unifilares del SEP, así como una interacción directa con elementos del diagrama mismo. El ambiente gráfico puede usarse tanto en la presentación de resultados como en la conexión o desconexión de elementos, cambio de parámetros de los mismos, etc., logrando una mayor interacción para el usuario con su objeto de trabajo.

Para incrementar la flexibilidad en el manejo de información entre aplicaciones es necesario diseñar y emplear bases de datos relacionales.

A fin de lograr una interacción más completa entre los algoritmos y la ambientación, para el manejo de contingencias, detener simulaciones, etc., se requiere programar algoritmos en un lenguaje compatible con el de la ambientación lo cual apunta al uso de C++ para también tomar ventaja de la Programación Orientada a Objetos.

REFERENCIAS

[1] J. Arrillaga and C. P. Arnold, "*Computer of Power Systems,*" John Wiley & Sons, New York, 1990, (1a edición).

[2] Andreas F. Neyer, Felix F. Wu and Karl Imhof, "*Object Oriented programming for Flexible Software: Example of a Load Flow,*" IEEE Transactions on Power Systems, Vol. 5, No. 3, August, 1990.

[3] J. M. Undril, F. P. deMello, T. E. Kostyanik and R. Mills, "*Interactive Computation in Power System Analysis,*" IEEE Proc., Vol. 62, No. 7, July, 1974.

[4] J. V. Oldfield, "*Graphical Input Output Techniques in Power System Computation,*" Proceedings del PSCC, Report 2.7, 1974.

[5] A. F. Glim, "*Disk-oriented Data Storage and Retrieval System Applied to Power System Analysis,*" Proceedings del PSCC, Report 2.7, 1974.

[6] A. R. Benson, "*The Interface Between Systems and People,*" IEEE, 1974.

[7] M. Rafian, M. J. H. Sterling, M. R. Irving, "*Real Time Power Simulation,*" IEEE Proc., Vol. 134, Pt. C, No. 3, May, 1987.

[8] M. Foley, Y. Chen, A. Bose, "*A Real Time Power System Simulation Laboratory Environment,*" IEEE/PES, Winter Meeting, 1990, Paper 90 WM 073-7 PWRS.

[9] N. Pahalawatha, C. P. Arnold, M. Shurety, "*A Power System CAD Package for the Work Station and Personal Computer Environment,*" IEEE Transactions on Power Systems, Vol. 6, No. 1, February, 1991.

[10] G.E. Young, "*Software Design and Implementation for Real-Time Distributed Systems,*" IFAC Advances in Control Education, Boston, Massachusetts, June, 1991.

[11] Ken Kato, H. Rod Fudeh, "*Performance Simulation of Distributed Energy Management Systems,*" IEEE Transactions on Power Systems, Vol. 7, No. 2, May 1992.

[12] J. L. Scheidt, S. E. Miller, S. A. Klein, M. K. Enns and S. C. Savulescu, "*Future Role of High Level Languages in Power System Control Centers,*" IEEE Transactions on Power Systems, Vol. 3, No. 3, August, 1988.

[13] Olle I. Elgerd, "*Electric Energy Systems Theory: An Introduction,*" Mc. Graw-Hill Book Co., 1983.

[14] Florencio Aboytes, Editor, "*Control de Voltaje en Sistemas Eléctricos de Potencia,*" Centro Nacional de Control de Energía de la Comisión Federal de Electricidad de México, Monterrey, N. L., Mayo 1991.

[15] Alberto Avalos, Oscar Chacón y Salvador Acha, "*Método Simplex Aplicado al Problema de Flujos,*" Reunión de Verano de Potencia IEEE, RVP-91, 1991.

[16] B. Stott and O. Alsac, "*Fast Decoupled Load Flow,*" *IEEE Transactions on Power Apparatus and Systems*, PAS-93, 1974, pp. 859-867.

[17] Salvador Acha Daza, "*Inversión Parcial de* Y_{bus}," Escuela de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo, Morelia, Mich., Febrero 1979.

[18] L.L. Freris, and A.M. Sasson, "*Investigation of the load-flow problem,*" Proc. Inst. Electr. Eng. (IEE), Vol 115, No. 10, October 1968.

[19] W.F. Tinney, and C.E. Hart, "*Power Flow solution by Newton's method,*" IEEE Trans. Power Appar. Syst., PAS-86, pp. 1449-1456, 1967.

[20] Salvador Acha Daza, "*Curso de Análisis Avanzado de Sistemas Eléctricos de Potencia,*" Doctorado en Ingeniería Eléctrica, Universidad Autónoma de Nuevo León, Agosto 1990.

[21] G.W. Stagg, A.H. El-Abiad, "*Computer Methods in Power System Analysis,*" Mc. Graw-Hill Book Co., 1968.

[22] Alberto Avalos, Salvador Acha, "*Aspectos Básicos y Metodología para el Análisis de Cargabilidad,*" SIEEEM 93, IEEE Seccion Monterrey, Octubre 1993.

[23] Alberto Avalos, "*Técnicas para el manejo de nodos PV en el Método Desacoplado Rápido,*" Tesis de Licenciatura, Escuela de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo, Morelia, Mich., Agosto 1990.

[24] Isaías Elizarraraz, "*Curso Sistemas de Potencia*," Escuela de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo, Morelia, Mich., 1988.

[25] T. A. Green and A. Bose, "*Open Systems Benefit Energy Control Centers,*" IEEE Computer Applications in Power, Vol 5, No. 2, pp. 45-50, April 1992.

[26] Tsutomu Oyama, Tatsuya Kitahara and Yasuo Serizawa, "*Parallel processing for power system analysis using band matrix*," IEEE Transactions on Power Systems, Vol 5, No 3, pp. 1010-1016, August 1990.

[27] Richard Gutman, "*Application of line loadability concepts to operating studies*," IEEE Transactions on Power Systems, Vol. 3, No. 4, pp. 1426-1433, November, 1988.

[28] Simpson Linke, "*Surce Impedance Loading and Power Transmission Capability Revisited.*"

[29] R. D. Dunlop, R. Gutman and P. P. Marchenko, "*Analytical Development of Loadability Characteristics for EHV and UHV Transmission Lines*," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-98, No.2 March/April, 1979.

[30] B. Stott and O. Alsac, "*Fast Decoupled Load Flow*," IEEE Transactions on Power Apparatus and Systems, PAS-93, 1974, pp. 859-867.

[31] Pappas and Murray, "*Borland C++ HandBook*," Mc. Graw-Hill Book Co., Second Edition, 1992.

[32] Keith Weiskamp, Loren Heiny and Bryan Flamig, "*Object-Oriented Programming with Trubo C++*," John Wiley and Sons, 1991.

[33] Bruce Eckel, "*Aplique C++*," Mc. Graw-Hill Book Co., 1991.

[34] Paquete de Software "*Borland C++ Application Frameworks, 3.1.*"

[35] Paquete de Software "*Quick Basic, Versión 3.0.*"

[34] Paquete de Software "*Turbo Pascal, Versión 6.0.*"

[35] Simulador Interactivo de Sistemas de Potencia SISP-PC, Mayo, 1993. Desarrollado por el Departamento de Investigación y Desarrollo de la CFE, con sede en Monterrey, N.L.