UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO-MATEMATICAS

APLICACION DE UN AMBIENTE DE MANIPULACION SIMBOLICO

TESIS

QUE PARA OBTENER EL GRADO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

PRESENTA

MIREYA GARCIA RIOS

MONTERREY, N. L., SEPTIEMBRE DE 1988

Con cariño
para el prof.
Aurelio Kmz.
de su alumna
Mirlyn Garcia

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO-MATEMATICAS

APLICACION DE UN AMBIENTE DE MANIPULACION SIMBOLICO

TESIS

QUE PARA OBTENER EL GRADO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

P R E S E N T A

MIREYA GARCIA RIOS

MONTERREY, N. L., SEPTIEMBRE DE 1988

DEDICATORIAS

A mis padres:

Napoleón y Olga

Que me han apoyado moral y económicamente
siempre.
Porque me dieron el coraje suficiente para el
logro de ésta meta cuando ya casi desistía.

.. ..Porque los quiero mucho

A mis hermanos:

Mary. Marcos. Marisa. Martha y Miriam

Porque me han dado algo de sí.
...cada uno a su manera
el cual forma parte de este trabajo

A mis sobrinitos:

Mayra. Maricelita. Luis Angel. Olga Nayeli.
..... y los que estan por venir.

.....con todo mi corazón

RECONOCIMIENTOS

Al Dr. René Bañares Alcántara. porque gracias a él surgió el tema de tesis y el aceptarme para el desarrollo de la misma.

Al M.C. Eduardo Morales Manzanares. por su asesoría en el desarrollo de este trabajo y revisión de la misma.

Al Ing. Gilberto Reyes Barrera. por su apoyo desinteresado y asesoría por parte de la Universidad Autónoma de Nuevo León.

Al Director de la Facultad de Ciencias Fisico-Matemáticas. Fis. Luis Vicente García Glz.. así como a los catedráticos M.C. Juan Antonio Alanis Rdgz. y M.C. Juan Gilberto Solis Alanis. que me brindaron su apoyo.

Al Instituto de Investigaciones Eléctricas por las facilidades proporcionadas a lo largo de mi estancia.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT). por la ayuda económica brindada para el desarrollo de la tesis.

A ti Renaut. ....en especial.
por tus consejos y apoyo
a todo lo largo de la tesis.

.....con Amor

AGRADECIMIENTOS

A todas aquellas personas que cuando necesite un consejo o ayuda me lo proporcionaron.

Especialmente al M.C. Salvador Villarreal y M.C. Raúl Jacinto M. que con sus atinados consejos y comentarios ayudaron a la realización de la presente.

Al Ing. Carlos Ramirez e Ing. José Montoya, por darse tiempo en atenderme y explicarme el desarrollo de los proyectos en que trabajan. lo cual me permitió poder aplicar muMATH en esas áreas.

Por otro lado quiero agradecer al M.C. José Luis Cisneros. la asesoría para el manejo del editor en el cuál se escribió esta tesis . Gracias también a Abelardo que en algunos casos me saco de apuros en el manejo del editor y por la realización de algunos dibujos hechos con otro paquete. A Ramón por mandar imprimir rápido mi tesis ya que estaba siempre al final de la lista de espera.

A todo el cuerpo Administrativo del Departamento de Simulación que siempre me tendieron la mano cuando lo necesite:
Jane. Rocío. Chelita. Rosalba. Esther, Ofelia y Jaime.

A quienes que con su convivencia hicieron mi estancia más placentera.
Gracias:
Nazira. Miguel. Silvia. S. Izordia. Héctor. Bourget. Piri. De Lara.

.....y muchas otras personas.

con las cuales no convivi. pero que con un saludo. una sonrisa o un simple comentario ayudan a levantarnos el ánimo.

Contenido

Lista de Figuras

Capítulo 1

PANORAMA GENERAL

1.1 Introducción

Este trabajo de tesis intenta presentar como objetivo principal, la importancia de manejar un ambiente simbólico para su aplicación en cualquier área, principalmente en la de Ingeniería.

Para este propósito se utilizó el paquete simbólico muMATH 5 que era con el único que se contaba, además de que cualquier otro paquete de los existentes en el mercado es muy elevado su costo.

Como una parte de los objetivos, se tenía que conocer primeramente con que herramientas se contaba, de tal manera de tener una idea de que tipo de construcción de ambiente se podía hacer.

Una de las principales ventajas del paquete a manejar, era que tenía su propio lenguaje: muSIMP, con el cual se podían construir nuevas funciones o expander las ya existentes, no necesita mucha memoria y su costo no es tan elevado.

Pero si bien presentaba estas ventajas, había otras desventajas que fueron surgiendo conforme las diferentes aplicaciones realizadas, las cuales son comentadas en los capítulos subsecuentes.

muMATH presenta algunas deficiencias al tratar de utilizarlo en aplicaciones serias, ya que hay que conocerlo bien de tal manera de saber combinar ciertas funciones para poder obtener resultados finales a fin de satisfacer ciertas necesidades especificas. Se trató de automatizar al máximo toda una serie de procedimientos por medio de muSIMP, de tal manera de facilitar al usuario el uso del paquete. Para ésto, se tuvieron que crear nuevas funciones tanto genéricas como específicas.

1.2 Contenido

Primeramente, en este Capítulo se tratará de dar a conocer al usuario la forma en que surgio el problema y la manera en que éste será solucionado, dándole a conocer además por medio de los Antecedentes la importancia del tema, así como su desenvolvimiento en la Historia.

El Capítulo Dos. contendrá un resumen del paquete muMATH. usado para el desarrollo del ambiente de manipulación simbólico. con la intención de dar una idea a los principiantes el tipo de operaciones que maneja.

Se comentará en el Tercer Capítulo las aplicaciones prácticas de muMATH dentro del Departamento de Simulación. los problemas encontrados y como fueron solucionados.

En el Capítulo Cuarto se comentarán las líneas de investigación en ésta área así como las conclusiones de los logros obtenidos en el uso de éste paquete.

En los Apéndices A. B y C se describen las funciones creadas a partir de ciertas necesidades y se explican cada una de ellas.

En el Apéndice D. se comenta el sistema creado para su aplicación en el Departamento de Simulación en el Instituto de Investigaciones Eléctricas.

En los Apéndices E y F se desarrolla el contenido de los paquetes muMATH y el lenguaje muSIMP

1.3 Naturaleza del Problema

Se desea obtener un ambiente de manipulación simbólica en donde se facilite el manejo de expresiones matemáticas. las cuales pueden incluir:

- simplificación y solución de ecuaciones.
- Operaciones con arreglos y matrices.
- Simplificación logarítmica y trigonométrica.
- Diferenciación e integración de expresiones.
- Solución de sistemas de ecuaciones lineales. no lineales y diferenciales.

1.4 Objetivos

1. Conocer muMATH. que es un paquete de manipulación algebraica el cual permite crear un ambiente para la obtención de soluciones matemáticas.
2. Familiarizarse con muSIMP. que es un lenguaje en el cual está escrito muMATH el cual tiene la facilidad de crear nuevas funciones segun las necesidades y de expander las ya existentes.
3. Realizar aplicaciones prácticas para un mejor conocimiento del paquete en donde se puedan detectar tanto las ventajas como limitaciones del paquet

1.5 Antecedentes

Las primeras computadoras fueron calculadoras numéricas. pero antes de que estas máquinas fueran diseñadas ya se tenía la concepción abstracta de computación como procesamiento de simbolos.

Esta idea junto con la lógica matemática. la cual había venido rápidamente en desarrollo desde fines del siglo IX. fueron las dos fuerzas más importante para la evolución de la Inteligencia Artificial en el medio intelectual de los 30's y 40's.

Turing. que ha sido llamado el padre de la Inteligencia Artificial. inventó un modelo no numérico de computación. Las ideas de Turing junto con Church y otros proveyó la cadena entre la formalización del razonamiento y las máquinas computadoras a inventar. Estos veían que los números eran un aspecto no esencial de la computación.

El primer programa simbólico SAINT (Symbolic Automatic INTegrator) fue escrito por James R. Slagle en 1963. Este programa resuelve problemas elementales de integración simbólica. SAINT fue escrito en LISP y es corrido en una computadora IBM 7090.

Un segundo programa importante de integración simbólica fue SIN (Symbolic INTegrator) escrito por Joel Moses en 1967.

Desde entonces surgieron diferentes sistemas de propósito general para el cómputo algebraico implementado en computadoras grandes. Dos sistemas de éstos más importantes son:

MACSYMA diseñado por Carl Engleman. William Martin y Joel Moses en 1968. Es un sistema interactivo que corre en DEC 10 o 20. VAX. Honeywell serie 6000 y Symbolics LM2 ó 3600. Requiere un mínimo de 2 megabytes.

REDUCE es un sistema interactivo que corre en una IBM-360 o 370. DEC 10 o 20. VAX. Univac 1100. CDC Cyber. Burroughs 6700 y otros más. Requiere de un mínimo de cerca de 350 Kilobytes.

En la última década se han descubierto nuevos algoritmos para encontrar el mayor común divisor de polinomios. factorización de expresiones racionales. simplificación de sumatorias. integración simbólica y análisis asintótico.

Como se observa. la importancia de la manipulación simbólica se remota desde épocas de los 30's y 40's siendo un factor muy importante para el surgimiento de la Inteligencia Artificial.

La principal meta de investigación en esta área. es la de inventar y analizar nuevos algoritmos matemáticos y extender los algoritmos numéricos previamente conocidos al área de manipulación simbolica 1

Capítulo 2

PRINCIPIOS DE MUMATH Y MUSIMP

2.1 Introducción

muMATH se desarrolló para brindar un cómputo algebraico de propósito general de mayor disponibilidad para cualquier tipo de usuario. El sistema es interactivo y permite crear nuevas funciones por lo que se puede expander. muMATH está implementado para computadoras basadas en los microprocesadores 8086 y 8088 corriendo bajo el sistema operativo MS-DOS y para el 8080 y Z80 corriendo bajo el sistema operativo CP/M-80. El sistema puede resolver problemas matemáticos de complejidad media en un tiempo razonable.

muMATH y muSIMP fueron diseñados e implementados por David Stoutemyer y Albert Rich de Soft-Warehouse. Honolulú. Hawaii. El proyecto empezó en 1976 por Rich con el diseño y código de un intérprete de LISP en una microcomputadora 8080. Convencido del poder y utilidad de las matemáticas simbólicas. Stoutemyer colaboró con Rich para mejorar el intérprete tanto en velocidad como en habilidad numérica. El lenguaje muSIMP fué revisado minuciosamente para dar al usuario el poder de un lenguaje aplicativo como LISP pero con una sintáxis más natural. Finalmente. el sistema matemático simbólico muMATH-79 fué escrito en muSIMP. siendo el primer sistema de éste tipo disponible en microcomputadoras.

La versión 83 de muMATH provee al usuario con un mejor desarrollo matemático. Varios paquetes matemáticos nuevos fueron agregados incluyendo un paquete para límites. un paquete para sumatorias. un paquete para ecuaciones diferenciales ordinarias. y un paquete de álgebra de vectores y cálculos.

En el Apéndice E se encuentra el desarrollo de cada uno de los paquetes que contiene muMATH con explicaciones y ejemplos para un mayor entendimiento de cada una de sus funciones y el Apéndice F contiene un resumen del lenguaje de manipulación simbólico muSIMP.

2.2 Requerimientos del Sistema

muMATH ha sido implementado para una amplia variedad de microcomputadoras. Enseguida se enlistan las necesidades requeridas tanto en hardware como en software para correr muMATH.

A) IBM Computadora Personal

1. Una IBM PC con tablero y monitor.
2. Al menos 128 K de memoria RAM.
3. Al menos un diskette IBM PC. de 5 1/4 pulgadas.
4. El diskette debe contener el software IBM DOS versión 1.1 o 2.0.

B) Computadora $MS - DOS^{tm}$

1. Una microcomputadora basada en los microprocesadores 8086 o 8088 corriendo bajo el sistema operativo MS-DOS.
2. Al menos 128 K de memoria RAM.
3. Al menos un diskette capaz de leer el diskette maestro de muMATH.

C) Computadoras $Apple^{tm}$ II, IIe, o III con Z80 $SoftCard^{tm}$

1. Una computadora Apple II, IIe o III con monitor.
2. Memoria RAM de 64 K (es decir, 48 K de memoria en Apple II más 16 K RAMcard o Language Card).
3. Al menos un diskette de 5 1.4 pulgadas.
4. El diskette del Sistema Operativo CP/M-80 que es provisto con Z80 SoftCard.

D) Computadoras $CP\ M - 80^{tm}$

1. Una microcomputadora en los microprocesadores 8080. 8085 o Z80 corriendo bajo el disco del sistema operativo CP/M-80.
2. Al menos 56 K de memoria RAM.
3. Al menos un diskette capaz de leer el diskette maestro de muMATH.

2.3 Introducciór a muMATH y muSIMP

Las capacidades matemáticas de los lenguajes de programacion tradicionales tales como: ALGOL. APL. BASIC. FORTRAN. PASCAL. y PL I son esencialmente limitados a operaciones aritméticas. Programas escritos en esos lenguajes contienen fórmulas consistiendo de variables. funciones y operadores los cuales deben estar bien definidos y asignados a un valor numérico para que se puedan hacer los cálculos correspondientes.

En contraste, muMATH puede evaluar y simplificar fórmulas conteniendo variables que no se les ha asignado un valor numérico previamente. Tales variables son manejadas algebráicamente como cualquier matemático.

No es necesario tener cierto conocimiento de programación para usar muMATH en estos cálculos, únicamente tener presente las siguientes convenciones:

1. El operador "*" es usado para denotar multiplicación. Este puede ser omitido y ser reemplazado por un espacio excepto entre una variable y una expresión entre paréntesis.
2. El operador "^" es usado para denotar exponenciación o elevar una expresión a una potencia.
3. Las expresiones dadas por el usuario deben ser terminadas por un ";" seguido por un <RETURN>.

En los siguientes capitulos se dará una descripción detallada de como usar muMATH. También se incluirán unos pequeños ejemplos de como trabajar con muMATH para ir familiarizándose con el paquete.

Una vez que está encendida la computadora, insertar una copia del diskette muMATH que al menos contenga los archivos: MUSIMP.COM, MATSOL.SYS, y CALCULUS.SYS en el manejador ("drive") que se esté usando, a continuación teclear el comando:

MUSIMP MATSOL

Después de ésto, para indicar que ya ha cargado el ambiente aparece el signo: ?, en donde ésto indica que ya se pueden usar las facilidades de una calculadora matemática simbólica proporcionadas por MATSOL. Por ejemplo, para simplificar la fórmula:

$$2Y(Y^2 - Z) + 2Z(Y \quad Z)$$

introduzca la expresión equivalente enseguida del "prompt" en forma de signo de interrogación:

2Y * (Y^ 2 - Z) - 2Z *, (Y - Z);

y muMATH desplegará:

2 Y^3 - 2 Z^2

Otras transformaciones algebraicas incluyen la expansion de una expresión sobre un común denominador y factorizaciones parciales. Si se desea, una expresión simplificada puede ser salvada para usarse en subsecuentes expresiones, esto permite una secuencia complicada de operaciones interrelacionadas.

muMATH puede aceptar ecuaciones genuinas como expresiones, simplificando ambos lados independientemente. Diferentes tipos de ecuaciones pueden ser sumadas, multiplicadas, etc. en orden de obtener soluciones paso a paso. Este proceso puede ser automático si el paquete de Solucion de Ecuaciones (SOLVE) es cargado. Por ejemplo, para resolver la ecuación:

$$X(3 - X^2) = 4X(1 - C^2) \quad X$$

para X en términos de C. introduzca la expresión:

SOLVE(X * (3 + X^2) == 4X^(1 - C^2) - X, X):

y muMATH desplegará el conjunto de soluciones:

$$\{X == -2C,\ X == 2C,\ X == 0\}$$

El sistema también maneja arreglos. álgebra de matrices y determinantes. con valores de entrada no-numéricos permitidos.

Por ejemplo. para encontrar la inversa de la matriz:

$$\begin{vmatrix} 1 & 2Y \\ -2 & Y \end{vmatrix}$$

introduzca la expresión:

{[1, 2Y], [-2, Y]} ^-1:

muMATH desplegará la matriz inversa:

{[1/5, -2/5],
[2/5/Y, 1/(5Y)]}

Para correr el resto de los ejemplos en este capítulo se necesita cargar el paquete de cálculos matemáticos tecleando el comando LOAD. ésto es:

LOAD(CALCULUS):

Sumatorias y productos cerrados sobre limites numéricos y nonuméricos pueden encontrarse usando muMATH. Por ejemplo, para determinar:

$$\sum_{j=0}^{n-1} C^j$$

En términos de C y N. introduzca:

SIGMA(C^j, j, 0, n-1):

y muMATH desplegará:

(1 - C^n) / (1 - C)

muMATH tiene un extenso conjunto de transformaciones opcionales y automáticas para valores absolutos. logaritmos. exponenciales. expresiones trigonométricas e hiperbólicas. Por ejemplo. para simplificar:

$$2(\cos x)^2 \sin y - 2\cos x \cos y \sin x - \sin y$$

introduzca la expresión:

TRGEXPD(2(COSx)^2 SINy - 2 COSx COSy SINx - SINy, 30):

y muMATH desplegará:

SIN(2X - Y)

muMATH automatiza tediosos cálculos en diferenciales e integrales. Por ejemplo, para encontrar la integral de:

$$\int CX^2 + XSIN(X^2)$$

con respecto a X, introduzca la expresión:

INT(C X^2 + X SIN(X^2), X):

y muMATH desplegará:

C X^3/3 - (COS(X^2)) 2

Cálculos de la expansión de la serie de Taylor también se pueden hacer en muMATH. Por ejemplo, para determinar la expansión de la serie de Taylor de E^{sinx} truncada en el quinto grado, alrededor de x = 0, introduzca:

TAYLOR(#E^SINx, x, 0, 5):

y muMATH desplegará:

1 + X - X^2/2 - X^4/8 - X^5/15

La regla de L'Hopital es usada por el sistema para determinar límites indeterminados. Por ejemplo para determinar:

$$\lim_{x \to 0} \frac{a^x - a^{sinx}}{x^3}$$

introduzca la expresión:

LIM((a^x - a^SINx) x^3, x, 0):

y muMATH desplegará:

LN(a) 6

muMATH puede resolver ecuaciones diferenciales. Por ejemplo, para declarar que "Y" depende de "X" y que "'" denota diferenciación con respecto a "X", introduzca:

DEPENDS(y(x)) $

DIFVAR: 'x:

enseguida tecleé el comando:

SOLVE(x y'' - 2 (x - 1) y' + 2 (x - 1) == 0, y):

y muMATH desplegará:

y == =E^x ARB(1) SINx x E^x ARB(2) COSx x

muMATH realiza álgebra de vectores y cálculo de vectores en cualquier sistema de coordenadas curvilíneas ortogonales. Por ejemplo, para establecer coordenadas esféricas introduzca el comando:

NEWCOORDS(r SINtheta COSphi, r SINtheta SINphi, r COStheta, [r, theta, phi]);

luego para calcular la divergencia del vector:

[SINtheta, COSphi, r SINphi]

introduzca el comando:

DIV[SINtheta, COSphi, r SINphi]

y muMATH desplegará:

COStheta COSphi/(r SINtheta) − 2 SINtheta/r + COStheta/SINphi

muMATH usa exacta precisión en todos los cálculos numéricos. Por ejemplo, si sé introduce la expresión:

5/6 - 0.25;

muMATH desplegará:

7/12

Notesé que las respuestas fraccionales son reducidas al menor término y los resultados son siempre exactos. Como un ejemplo de la precisión de la aritmética de muMATH para calcular la octava potencia de factorial de diez introduzca:

10!^8;

y muMATH desplegará

3006798071416758059974231133043818496000000000000000000

El número y tamaño de expresiones que muMATH puede acomodar es limitado sólamente por la memoria RAM disponible en la computadora con que se esté trabajando.

muMATH es modular, de tal manera que no se necesita conocer un alto nivel de matemáticas para usar un nivel más bajo.

muMATH está escrito en el lenguaje de programación muSIMP, el cuál fué especialmente diseñado para manejar las demandas de procesamiento matemático simbólico. muSIMP es documentado para poder extender muMATH si es necesario. Además, los archivos fuentes de muMATH son provistos de tal manera que pueden ser modificados si se desea o ser usados como modelos para creación de otras extensiones.

2.4 Contenido del Sistema

2.4.1 Archivos Fuentes de muMATH-83

Son archivos de texto impresos en ASCII. Cada archivo provee un paquete modular de capacidades matemáticas. Así. sólo se necesita leer el archivo fuente requerido para resolver un problema matemático dado. Esto hace posible correr muMATH en computadoras de tamaño de memoria limitadas. Estos archivos son:

- ARITH.MUS aritmética racional
- ALGEBRA.ARI álgebra elemental
- EQN.ALG simplificación de ecuaciones
- SOLVE.EQN solución de ecuaciones
- ARRAY.ARI operaciones con arreglos
- MATRIX.ARR operaciones con matrices
- LINEQN.MAT ecuaciones algebraicas lineales simultáneas
- ABSVAL.ALG simplificación de valores absolutos
- LOG.ALG simplificación logarítmica
- TRG.ALG simplificación trigonométrica
- ATRG.TRG simplificación trigonométrica inversa
- HYPER.ALG simplificación trigonométrica hiperbólica
- DIF.ALG diferenciación simbólica y serie de Taylor
- INT.DIF integración simbólica
- INTMORE.DIF extensión de integración simbólica
- LIM.DIF funciones de límites
- SIGMA.DIF formas cerradas de sumatorias y productos
- ODE.SOL ecuaciones diferenciales ordinarias de 1er orden
- ODENTH.ODE ecuaciones diferenciales ordinarias de 2o orden
- ODEMORE.ODE extensión de los métodos de 1er orden de ODE
- VEC.ARR álgebra de vectores
- VECDIF.VEC cálculo de vectores

2.4.2 Archivos SYS de muMATH-83

Son imágenes de archivos de memoria no impresos. Un archivo SYS es creado por el comando SAVE y es cargado por el comando LOAD. Cada archivo SYS contiene las capacidades

matemáticas presentes en el tiempo que es salvado

- ALGEBRA.SYS incluye ARITH.MUS y ALGEBRA.ARI
- CALCULUS.SYS incluye ALGEBRA.SYS. LOG ALG. TRG.ALG. ATRG.TRG. DIF.ALG. INT.DIF, INTMORE.INT. LIM.DIF, y SIGMA.DIF
- MATSOL.SYS incluye ALGEBRA.SYS. EQN.ALG, SOLVE.EQN. ARRAY.ARI, MATRIX.ARR y LINEQN.MAT

2.4.3 Lecciones de muMATH

Son archivos de lecciones interactivas que enseñan como usar muMATH.

- CLES1.ALG aritmética racional y asignación de variables
- CLES2.ALG factoriales y potencias fraccionarias
- CLES3.ALG expansión de polinomios y factorización
- CLES4.ALG variables complejas y substitución de expresiones

Estas lecciones necesitan que el archivo ALGEBRA.SYS esté cargado antes de empezar con las lecciones.

2.5 Guía de Operación

Aquí se explican las técnicas básicas requeridas para usar muMATH y muSIMP.

Son requeridos tres pasos básicos para trabajar con muMATH por primera vez. Primeramente hay que cargar y correr el intérprete muSIMP. En seguida construir el sistema muMATH deseado leyendo los archivos fuentes. Finalmente, salvar el sistema muMATH resultante como un archivo SYS. El archivo SYS facilita volver a cargar el sistema cuando se desee.

2.5.1 Como cargar muSIMP

Primero hay que cargar el sistema operativo de la computadora en la forma normal. En seguida, si es necesario, cambiar de manejador ("drive", segun se encuentre el disco conteniendo una copia de MUSIMP.COM . Una vez que el "prompt" del sistema operativo aparezca en la pantalla teclear el comando:

MUSIMP

y oprimir la tecla <RETURN>. Después de algunos segundos, el mensaje del encabezado de MUSIMP:

muSIMP-83 4.nn (mm dd yy)
xxxxxxxxxxx versión
Copyright (c) 1982 The SOFT WAREHOUSE

es desplegado en la pantalla.

Abajo del mensaje de encabezado. el "prompt" de muSIMP será desplegado. El "prompt" consiste del signo de interrogación (?) seguido de un espacio y el cursor de la consola.

El "prompt" de muSIMP indica que el sistema está listo para aceptar una expresión dada. Una expresión en muSIMP es un comando para ser ejecutado o una expresión matemática a ser evaluada. La expresión dada después del "prompt" debe ser terminada por un punto y coma (;), un "ampersand" (&). o un signo de pesos ($). Un punto y coma se usa para imprimir resultados en notación matemática. Un "ampersand" se usa para imprimir resultados en notación de lista. El signo de pesos es usado para evitar desplegar los resultados.

Tan pronto como el <RETURN> sea dado. muSIMP lee y analiza sintácticamente la expresión. Si no hay error en la sintáxis. la expresión es evaluada. Cuando la fase de evaluación es completa. la cadena: "@:" es desplegada para anunciar la respuesta. Los siguientes son ejemplos de interacciones en muSIMP:

```
? 2 + 5 - 3 4;
@: -5

? JUAN = MARY;
@: FALSE

? MEMBER (MANZANA. '(PASA CIRUELA MANZANA NARANJA))&
@: (MANZANA. NARANJA)

? AGE: 34$

? AGE;
@: 34
```

Cada vez que el "prompt" de muSIMP es desplegado. muSIMP causa que la computadora emita un sonido. Esto permite saber que muSIMP está listo para aceptar más datos de entrada. Para evitar este sonido. se puede usar el comando:

BELL: FALSE$

Este comando coloca la variable de control BELL en falso lo cuál desactiva el sonido.

El comando de muSIMP:

SYSTEM();

Termina el ciclo de interacción de muSIMP. es decir. sale de muSIMP y regresa el control al sistema operativo.

2.5.2 Lectura de Archivos Fuente

Archivos fuentes son archivos de texto en ASCII conteniendo una secuencia de comandos en muSIMP. El archivo fuente correspondiente a cada paquete matematico contiene los comandos de muSIMP necesitados para establecer las capacidades matemáticas provistas por

ese paquete. El archivo fuente evita teclear los comandos cada vez que se necesite usar el paquete. Otra ventaja de los archivos fuentes es que se pueden editar usando el editor de muSIMP o cualquier otro editor de texto estándar.

Un comando RDS es usado para leer un archivo fuente en muMATH. El comando RDS (ReaD Select) cambia la fuente de entrada de datos de muSIMP del teclado a un archivo fuente dado. La forma de este comando es:

RDS('name. 'type. 'drive):

donde <name> es el nombre del archivo. <type> es el tipo de archivo y <drive> es la letra del manejador ("drive") en el cuál se encuentra el archivo. <drive> es un argumento opcional que toma automáticamente el manejador ("drive") donde se dió de alta el sistema.

Por ejemplo, para leer el paquete ARITH.MUS de muMATH una vez que se está en muSIMP. teclear:

? RDS('ARITH. 'MUS):

seguido por <RETURN>. Si ARITH.MUS se encuentra. el nombre del archivo es desplegado y los comandos son leídos y ejecutados. Después que cada comando es leído. un signo de interrogación es desplegado (?). Esto permite saber el tiempo que tarda muSIMP en leer el archivo. Si ARITH.MUS no se encuentra, FALSE es desplegado en la pantalla.

Al final de cada archivo fuente de muMATH hay una demostración corta que muestra las capacidades provistas por ese archivo. Un comentario encerrado en signos de tanto por ciento (%) describirá cada problema. Después que el problema y su respuesta fueron desplegados. el mensaje de opción:

Abort. Break. Continue. DOS?

es desplegado en la pantalla. muSIMP espera entonces por la selección deseada del usuario que puede ser cualquiera de las letras: A, B. C. o D.

A: aborta la demostración. El "prompt" de MUSIMP es desplegado. y el sistema espera por datos de entrada.

B: Suspende temporalmente la demostración. El "prompt" de MUSIMP es desplegado, y se pueden probar ejemplos específicos. Cuándo se este listo para continuar con la demostración. introduzca "underscore" y oprima <RETURN>.

C: *Continúa la demostración con el problema siguiente.*

D: Termina la demostración. MUSIMP es terminado y el control se regresa al sistema operativo residente.

para evitar la demostración automática de MUMATH. haga la asignacion:

DEMO: FALSE$

antes de usar el comando RDS para la lectura de los archivos fuentes.

Suponga que se quiere construir un sistema en muMATH con la capacidad para resolver ecuaciones algebraicas. Como se muestra en el diagrama de dependencia 2.1 al final de este capítulo. SOLVE.EQN requiere de los paquetes ARITH.MUS. ALGEBRA.ARI. y EQN.ALG.

Los siguientes pasos son los requeridos para construir un sistema de tal naturaleza.

```
? DEMO: FALSE$
? RDS('ARITH, 'MUS);
@: ARITH

? RDS('ALGEBRA, 'ARI);
@: ALGEBRA

? RDS('EQN, 'ALG);
@: EQN

? RDS('SOLVE, 'EQN);
@: SOLVE
```

Esta construcción requiere de varios minutos. De cualquier manera, esto se necesita hacer sólo una vez ya que este sistema puede ser salvado en un archivo con extensión SYS. Más adelante se explicará como cargar y salvar archivos SYS.

2.5.3 Archivos SYS

Un medio ambiente MUSIMP consiste de todos los valores de variables, definición de funciones, y valores de propiedad en el sistema. Estos valores y definiciones podrían ser dados directamente del teclado, ser leídos de un archivo fuente, o cargados de un archivo SYS. El comando de muSIMP SAVE, cuyo formato es:

SAVE('name,'drive)

crea un archivo SYS conteniendo una imagen de memoria compacta del medio ambiente existente al tiempo que el comando es ejecutado.

El comando LOAD:

LOAD('name,'drive)

reestablece este medio ambiente. También se puede cargar el ambiente al inicio cuando se invoca a muSIMP, poniendo el nombre del archivo después de MUSIMP:

MUSIMP <nombre del archivo> % Sin extensión %

Conforme un archivo fuente se vaya leyendo, muSIMP analiza sintácticamente los comandos en una forma interna para entonces ejecutarlos. Cuando el comando es una definición de función, la definición es seudocompilada en una forma compacta. Esto requiere buscar en todas las funciones previamente compiladas por cualquier subexpresión que puede ser compartida con las nuevas funciones

A pesar de la rapidez para esta optimización de espacio, la compilacion y análisis sintáctico de tales archivos fuentes requiere considerablemente más tiempo que cargar un archivo equivalente de imagen de memoria SYS.

Por ejemplo, el listado fuente para los archivos: ARITH.MUS y ALGEBRA.ARI, los dos más grandes paquetes de muMATH, son cerca de 20 hojas impresas. Para leer estos archivos toma

varios minutos usando el comando RDS contra solamente pocos segundos que lleva cargar el archivo SYS correspondiente. Esta diferencia de velocidad es una de las mejores razones para usar archivos SYS.

2.5.4 Restricciones de Memoria

El espacio de memoria es requerido para almacenar las definiciones de funciones y valores de propiedad contenidos en cada archivo fuente de muMATH. Este espacio es también requerido para almacenar resultados simbólicos y valores numéricos intermedios durante la evaluación y simplificación de una expresión. Ya que la computadora tiene una cantidad finita de memoria, hay que tener cuidado del espacio requerido para correr muMATH.

Es esencial guardar suficiente cantidad de espacio de memoria de tal manera que muMATH pueda almacenar resultados intermedios. Si durante la evaluación de una expresión, hay insuficiente espacio, el mensaje de error:

Memory Space Exhausted

será desplegado y la evaluación deberá ser abortada.

Aunque el error de espacio de memoria insuficiente no ocurra, puede tomar bastante tiempo para la solución de problemas si solo hay una pequeña cantidad de espacio de trabajo.

Como una regla general, es buena idea cargar solamente los paquetes requeridos para resolver un conjunto dado de problemas.

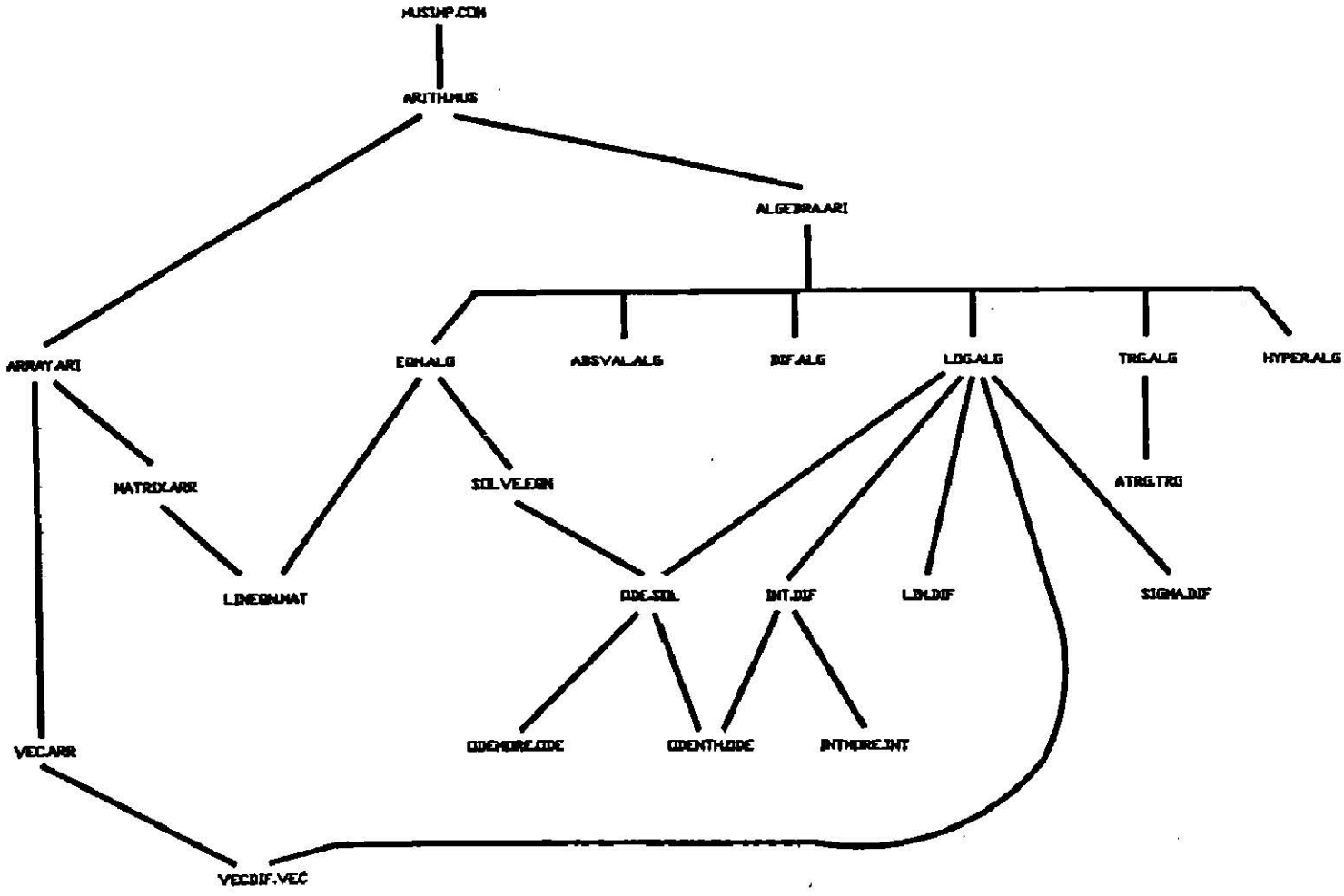


Figura 2.1: Diagrama de Dependencia

Capítulo 3

PROBLEMAS DE MUSIMP Y APLICACIONES PRACTICAS

3.1 Introducción.

Este capítulo se divide en dos partes. En la primera parte se comentan los problemas encontrados en general en las funciones de muSIMP y en el paquete muMATH. independientemente de alguna aplicación y la manera en que éstos fueron resueltos. Los problemas resueltos fueron los de mayor interés para la obtención de resultados en las áreas de aplicación y de mayor importancia. principalmente en el manejo de un ambiente de manipulación simbólico. Algunos otros problemas que se fueron detectando en el transcurso de las aplicaciones prácticas pero que no fueron de importancia. simplemente se ignoraron. Existieron sin embargo problemas que no hubo manera de resolver lo cual da a conocer en cierta manera los limites de complejidad del tipo de ecuaciones que el paquete puede manejar.

En la segunda parte se desarrollan las aplicaciones de muMATH en el área de simulación, para lo cual se tuvieron que crear una serie de herramientas que se pueden aplicar genéricamente como específicamente. Las aplicaciones a comentar estan divididas por proyecto para un mejor entendimiento. en donde éste a su vez se subdivide en: objetivo. desarrollo. aplicación y tendencias.

3.2 Problemas Encontrados

Estos se dividieron en:

1) Factorización
2) Potencias Fraccionarias
3) Lectura por Pantalla
4) Comunicación con otros lenguajes

5) Inversión de matrices

6) Memoria

3.2.1 Factorización.

El paquete Algebra Elemental contiene un subpaquete el cuál lleva el nombre de Funciones de Transformación Algebraicas. Este cuenta. como su nombre lo indica. con funciones de transformación para el manejo de expresiones matemáticas simbólicas. La única función para tratar de factorizar una expresión dada es FCTR. Para expresiones un poco más complicadas FCTR no podía simplificar adecuadamente. Se sabía con seguridad que la expresión se podía factorizar, pero tal parecía que el problema consistía en que muMATH no podía "ver" por ejemplo. las conveniencias de sacar cierto factor común en el numerador para que se eliminara con uno en el denominador o el factorizar una expresión con respecto a cierta variable. de tal manera de reducir al máximo la expresión.

Debido a ésto. se empezó a experimentar con las variables de control. las cuales tienen control sobre las transformaciones usadas para simplificar una expresión. Es decir. se les puede asignar cierto valor específico lo cual producirá que cierta expresión sea distribuida o factorizada con respecto a otra subexpresión, dependiendo del valor que tome la variable de control correspondiente.

Los valores que toman las variables de control pueden ser alguno de los siguientes números primos: 2. 3. 5 o un múltiplo de estos primos. Valores positivos causan expansión, mientras que valores negativos producen reducciones sobre las expresiones.

Se empezó a trabajar con estas variables para entender más su funcionamiento y se notó que un buen uso de ellas podían producir resultados muy compactos. Por ejemplo. uno de los resultados intermedios que se tenía al estar practicando con diferentes ecuaciones era:

$$FOO: 5A*(-1-A)\hat{}3-4B1*(-1-A)\hat{}3-(A-B1)(2A*(-1-A)\hat{}2-2(-1-A)\hat{}2)-(-1-A)\hat{}3$$

el cual se escogió como ejemplo ya que no esta muy complicado y se pueden observar mejor los cambios que va sufriendo. Primeramente, la expresión se asignó a la variable FOO para luego poder seguir manejándola con respecto a esta variable. siendo ésto mucho más sencillo. Antes de hacer esta asignación se le colocó a la variable de control NUMNUM el valor de cero para evitar que muMATH automáticamente reacomodara la expresión. lo cual siginificaba que algunos términos los iba a expander y esto algunas veces no es conveniente. Es decir. las variables de control por default tienen asignados ciertos valores. de tal manera que al dar cierta expresión al sistema éste automáticamente lo reacomoda.

Si se introduce la expresión anterior al sistema sin cambiar ningún valor de NUMNUM (únicamente es necesario cambiar NUMNUM porque la expresion dada contiene solo numeradores) éste hará lo siguiente:

? FOO: $5A*(-1-A)\hat{}3$ - $4B1*(-1-A)\hat{}3$ - (A-B1) (2A * (−1 − A) ^2 - 2(-1−A)^2) - (-1−A)^3:

©: $-4B1*(-1-A)\hat{}3$ − (-B1−A) (2 (-1−A)^2 A - 2(-1 - A)^2) - 5(-1−A)^3 A - (-1−A)^3

si se checa esta expresión. el resultado que arroja muMATH es diferente a la forma original de la ecuación. Si por el contario se hace:

? NUMNUM: 0:
@: 0

? FOO: $5A * (-1 - A)$ ^3 - $4B1 * (-1 - A)$ ^3 + (A-B1) $(2A * (-1 - A)$ ^2 - 2(-1+A)^2) - (-1+A)^3:
@: $5A * (-1 + A)$ ^3 - $4B1 * (-1 - A)$ ^3 - (A-B1) $(2A * (-1 - A)$ ^2 - 2(-1−A)^2) - (-1−A)^3:

con este cambio a NUMNUM el sistema respeta la forma de entrada de la ecuación que era lo que se quería primeramente.

El siguiente paso era estar modificando NUMNUM para ver los efectos que producía en la expresión. Si se observa bien la ecuación. se nota que una factorización con respecto a $(-1 + A)^3$ reduciría mucho el tamaño de la ecuación. Un cambio a valor negativo de la variable NUMNUM producirá una reducción sobre las expresiones dadas. Como el término con respecto al cual se quiere factorizar es complejo. es decir. es una suma elevada a cierta potencia. una manera de darle a conocer al sistema que con respecto a esta subexpresión se quiere factorizar es darle un valor negativo grande. por decir -30. se procedió a cambiar este valor para ver los efectos:

? NUMNUM: -30$

?FOO: EVAL(FOO):
@: (-1+A)^3 (5A - (1−4B1) − 2(A-B1))

Lo que se hizo después de cambiar el valor de NUMNUM fue evaluar la expresión con este nuevo valor y volver a dejar el resultado en FOO. El resultado obtenido era el deseado. se factorizó la expresión con respecto a $(-1 + A)^3$. Si se observa la expresión a lado del término $(-1 - A)^3$ aún contiene elementos comunes. Lo que se desea ahora es expander esa parte para quitar los paréntesis internos y que el sistema pueda "ver" claramente los términos comunes. Según el manual para expander términos lineales es suficiente con colocarle un valor de 6 a NUMNUM por lo que se procede a hacer lo siguiente:

? NUMNUM: 6$

?_FOO: EVAL(FOO):
@: (-1−A)^3 (7A - (1−6B1))

con esto cambios a NUMNUM fueron suficientes para obtener un expresión bastante simplificada.

De esta forma fue como se estuvo trabajando con diferentes tipos de ecuaciones hasta conocer bien la forma en que afectaban los cambios de las variables de control sobre las expresiones.

Hasta aquí. se podía reducir relativamente bien ciertas expresiones. para esto se necesitaba conocer el uso de las variables de control y tenían que ser expresiones relativamente simples para poder identificar a "ojo" que tipo de cambio era el más conveniente. Lo que se buscó fue cambiar éste uso interactivo de las variables por un programa. de tal manera que la reducción

de una expresión fuera automático. de lo contrario. las aplicaciones que se pudieran hacer íban a estar muy restringidas e íban a requerir de un conocimiento exhaustivo del paquete.

Se empezó a buscar la manera entonces. de como ir diciendole a la máquina que tipo de valor deberían de ir tomando las variables de control y como ir escogiendo las expresiones. de tal manera que fuera reduciendo la expresión. Interactivamente se escogían los valores de las variables según la forma de la expresión, pero ésto no lo podía "ver" la máquina.

Pero si bien la máquina no podía "ver" la complejidad de una ecuación para saber que cambio aplicar, se encontró otra forma en que se podía decir bastante de las expresiones a factorizar: su tamaño. Checando en el ejemplo anterior. se tenía inicialmente una ecuación de ésta forma:

$5A*(-1+A)$ ^3 - $4B1*(-1-A)$ ^3 - (A-B1) $(2A*(-1-A)$ ^2 - 2(-1+A)^2) - (-1-A)^3

donde después de hacer una serie de cambios con NUMNUM la expresión quedó reducida a una de la forma:

(-1-A)^3 (7A - (1-6B1))

A simple vista. si dieran a escoger de éstas dos ecuaciones para hacer cálculos. sabiendo que es la misma expresión. se escogería sin dudar la más corta ya que es la más fácil de manejar. Por lo tanto la selección de una ecuación con respecto a otra íba ser de acuerdo a su tamaño. Ya que muMATH no contaba con una función para conocer el tamaño de una expresión, se creó ésta. LONGITUD es el nombre de la función la cual dá como resultado precisamente la longitud de su argumento. En el apéndice A está definida la función junto con ejemplos.

Una vez que se tuvo una maner. de ir seleccionado las diferentes expresiones que se van creando al ir modificando los valores de las variables de control. se procedió a buscar una forma óptima de ir modificando estas variables.

Se sabía con seguridad. al estar practicando con diferentes tipos de ecuaciones y estar modificando los valores de la variables de control. que había unas formas comunes de ir modificando estas variables. Existían tres formas que eran las más comunes para la mayoría de las ecuaciones.

Es decir, de la expresión original que se tenía de entrada. se hacían tres cambios en las variables de control. en donde en cada cambio se guardaba la expresión resultante en una variable de paso. Todas las expresiones obtenidas con estos cambios eran comparadas entre sí, incluyendo la original y se escogía la que tuviera la longitud más corta.

Esta expresión seleccionada entraba a otra sección. en donde se volvían hacer otras tres modificaciones. de nuevo éstas eran comparadas entre si y se escogía la más corta. De esta manera se íba seleccionado la expresión que se fuera reduciendo más. La selección paraba cuando la expresion elegida en la segunda sección era igual a la elegida en la primera seccion. porque ésto quería decir que ya no se podía reducir más la expresion.

Estas formas de modificación en las variables de control. al menos con las expresiones practicadas. era la más óptima. aunque hubo casos en que los resultados obtenidos no eran muy satisfactorios. es decir. si reducía la expresión pero aún habia manera de que se pudiera reducir más. Esto se debe a que las formas de modificación no abarcan todas las combinacione-

posibles y era donde caían algunos tipos de ecuaciones.

No se quería abarcar todas las combinaciones posibles de valores de las variables de control, porque ésto incrementaría mucho el tiempo al ir escogiendo la expresión más corta, además el rango que se abarcaba con las modificaciones que se tenían para la selección de una expresión, era bastante grande, al menos con el material con que se contaba para hacer éstas pruebas.

Una vez teniendo bien definido y probado la forma en que se debería ir seleccionando una expresión, se procedió a automatizar todo esto integrándolo en un programa llamado REDUCE.

Las pruebas iniciales realizadas con REDUCE funcionaron bastante bien, todo se había podido integrar. REDUCE tomaba de su argumento una expresión y arrojaba como resultado una expresión equivalente factorizada.

REDUCE demostro su capacidad en cierto tipo de problemas de una aplicación práctica muy específica, en la cual se cubrían ciertas necesidades. Las pruebas realizadas era con el material precisamente de esta aplicación, en donde el tipo de ecuaciones a manejar era limitado con respecto a toda una variedad que se puede encontrar.

Algunos ejemplos de expresiones factorizadas por REDUCE, así como la definición de su función aparece en el apéndice B.

3.2.2 Potencias Fraccionarias

El paquete de Aritmética Racional: ARITH.MUS, contiene un subpaquete para potencias fraccionarias el cual provee facilidades para la simplificación de números con potencias fraccionarias y exponenciales complejos. El problema es que cuando no existen raíces reales, muMATH no hace alguna aproximación, simplemente deja expresado el término, ejemplo:

```
? 6^(1/2);
@: 6^(1/2)

? 24^(1/3);
@: 2*3^(1/3)
```

Lo único que hace el sistema es reducir al máximo la expresión, como lo muestra el último ejemplo, es decir obtiene todas las raíces posibles y el resto lo deja expresado.

En este punto no se trato de solucionar el problema ya que para los intereses creados no era importante su solución.

3.2.3 Lectura por Pantalla

Los tipos de lectura con el que cuenta el paquete son:

RDS(filename.filetype.drive)
READCHAR(peek-flag)
CONCHAR()

SCAN()
READLIST()

También se cuenta con la función READLINE, la cual no es una función primitiva de muSIMP y para el uso de ésta se tiene que cargar la función por medio del comando LOAD.

El problema con que se topó es que primeramente ninguna de las funciones primitivas muSIMP podían leer expresiones del tipo:

4x + 5y
6x(3x + 2x^2)

CONCHAR() y SCAN() únicamente leen los primeros caracteres de una expresión. READLIST() lo que hace es no tomar el primer caracter de la expresión y el resultado que regresa va acompañado de FALSE. Ejemplo:

? VAR: READLIST():
4x − 5y <RETURN> % EXPRESIóN DADA POR EL USUARIO %
@: FALSE

? VAR:
@: x + 5 FALSE y

? VAR: READLIST():
4x + 5y:
@: FALSE

?@: 5y − x

Como se observa, se asigna a la variable VAR lo que se desea leer por medio del comando READLIST() para luego procesar los datos de entrada pero ninguna de las dos formas que se muestran sirvió a este propósito. La diferencia es que en el primer caso, después de dar la expresión a ser leída se le da un <RETURN>, en el segundo caso la expresión a leer es terminada con ":". Ninguno de los dos resultados es exactamente la expresión que se desea leer.

Una vez que se hicieron varias pruebas por medio del comando READLIST() y se comprobó que este comando no era satisfactorio, se procedió a cargar READLINE:

```
? LOAD('READLINE);
```

Los resultados con READLINE fueron buenos, al menos cualquier tipo de expresión que se deseaba leer la podía interpretar bien.

Los problemas empezaron cuándo se usaba a READLINE dentro de un función definida por el usuario. Esto es, se tenía cierta función principal en donde se deberían de leer ciertas condiciones iniciales para hacer luego cálculos posteriores. Las condiciones iniciales eran expresiones de esta naturaleza:

$$\frac{\partial}{\partial}(T_1 \quad TS_2$$

$$-\beta_2(T_1 - TS_1)$$

los resultados arrojados no eran los esperados, por lo que se tuvo que checar muniosamente todas las variables involucradas en los cálculos. Se comprobó entonces que el problema estaba en la lectura de las condiciones iniciales. es decir. el uso del READLINE dentro de una definición de función no funcionaba.

Se empezó a investigar de que manera se podía solucionar este problema. y después de una serie de pruebas. se encontró que el comando MATCH podía solucionar el problema combinándolo con SCAN() de la siguiente manera:

```
MATCH(SCAN(),';)
```

Esta combinación lee una expresión hasta encontrarse con el operador ";". El problema en esta forma de lectura era. primero: que para poder detectar el ";" de una expresión hay que darle un segundo carácter después de este comando. el otro problema es que el resultado de la lectura es guardado en la forma WHEN <expresión leída> EXIT. Ejemplo:

? MATCH(SCAN(),';):
4x + 5y;;<RETURN>

@: WHEN 4x + 5y EXIT

Note como la expresión a leer es terminada con doble ";" y la forma del resultado es diferente. Para la solución del primer problema no se pudo hacer nada. así que la expresión a ser leída tenía que ser terminada con doble ";". El segundo problema se pudo resolver fácilmente. extrayendo la expresión del WHEN-EXIT con el comando FIRST. es decir haciendo:

? FIRST(MATCH(SCAN(),';):
4x − 5y;;<RETURN>

@: 4x + 5y

De ésta forma se había implementado una forma de lectura y se procedió a probarla desde una definición de función. tal parecía que algunos comandos primitivos no funcionaban igual usandolos interactivamente que desde una función definida por el usuario.

Las pruebas con este tipo de lectura funcionaron bien. la única nota aclaratoria que se agregó para el uso de este programa. donde las condiciones iniciales de entrada son dadas por el usuario. era que cualquier expresión de entrada debería ser terminada con doble ";". Un ejemplo para el uso de esta aplicación se desarrollará en la seccion de aplicaciones prácticas.

3.2.4 Comunicaciones con otros lenguajes y lectura de archivos

muSIMP tiene un comando: EXECUTE para la comunicación con otros lenguajes o con el Sistema Operativo. es decir. estando dentro del ambiente muSIMP se puede comunicar hacia afuera y regresar a muSIMP sin perder el ambiente en el que se encontraba antes. Es decir.

todos los cálculos o valores de variables estan aún disponibles. Ejemplo, para ver el directorio desde dentro de muSIMP:

? EXECUTE("COMMAND.COM","DIR W"):

éste desplegará el directorio del manejador ("drive") que esté cargado y regresará a muSIMP.

Para ver el contenido de alguna función desde muSIMP:

? EXECUTE("COMMAND.COM","TYPE <filename>.<filetype>"):

y la función se desplegará regresando luego al ambiente muSIMP.

Se trató de hacer una interfase con muSIMP a Turbo-Pascal por medio del comando EXECUTE. Los cálculos obtenidos eran guardados en un archivo y éstos eran leídos por medio del comando RDS para luego ser utilizados en cálculos posteriores. Inicialmente no había ningun problema tanto de comunicación como de lectura de archivos. Después de hacer varias pruebas, la lectura de los archivos de los resultados obtenidos con la interfase en Pascal no empezaban a funcionar, es decir, no se podían leer.

Se probaron varias soluciones al problema, pero la única que funcionaba era cambiarle de nombre al archivo donde se dejaban los resultados de la interfase con Pascal cada vez que no quisiera leerlos. Esta solución no era nada satisfactoria, pero tal parecía que era dañado el archivo después de usarlo varias veces, ya que después de cambiarle de nombre y hacer varias pruebas ocurría el mismo problema: al principio funcionaba bien pero después empezaba a fallar.

Se aclara que el problema aquí era, que tanto la interfase como la lectura de archivos estaban contenidos en un programa principal. Este problema no existe interactivamente, es decir, si se hace manualmente desde cierto ambiente muSIMP cada una de las instrucciones que se tenían en el programa principal, éstas eran realizadas satisfactoriamente.

Tal parecía que el sistema estaba orientado para un uso interactivo mejor que un uso automático, pero lo cual limitaba mucho sus beneficios.

La manera que se pudo resolver este problema es que los cálculos obtenidos en la interfase con Pascal se lograron hacer en muMATH, realmente el problema no se solucionó, simplemente se buscó otra manera de hacer las cosas las cuales finalmente beneficiaron, en el sentido de tener una forma más compacta y más rápida de realizar estos cálculos, ya que de otra manera, se tenían que estar leyendo archivos y estar pasando de un lenguaje a otro.

3.2.5 Inversión de Matrices

El paquete MATRIX.ARR de muMATH, cuenta con la facilidad de manejo de matrices simbólicas. Específicamente se trató con el punto de inversion de matrices.

Se empezó a trabajar con matrices de 2x2 del tipo:

$$\begin{pmatrix} 2s & s & 1 \\ s-1 & 3s & \end{pmatrix}$$

donde la solución arrojada checaba perfectamente con los resultados esperados. Cuando se empezó a trabajar con matrices de expresiones más complicadas, el sistema arrojaba resultados, pero los términos eran demasiado grandes. Un tipo de matriz simbólica un poco más complicada sería:

$$\begin{pmatrix} \frac{806s-.264}{s^2+1.15s-.202} & \frac{-(15.0s+1.42)}{s^3-12.8s^2-13.6s-2.36} \\ \frac{1.95s^2-2.12s+.490}{s^3+9.15s^2+9.39s+1.62} & \frac{7.14s^2+25.8s+9.35}{s^4-20.8s^3+116.4s^2+111.6s-18.8} \end{pmatrix}$$

en donde la matriz inversa correspondiente se encuentra en el apéndice C.

Los resultados arrojados contenían muchos términos comunes, pero por la misma complejidad de la ecuación muMATH no podía simplificar.

Existía también la certeza que REDUCE no podía factorizar la expresión. Se había comprobado que REDUCE no era óptimo, ya que su área de aplicación había sido limitado.

Se procedió entonces a optimizar REDUCE, ya que lo que se quería era una función que factorizara cualquier tipo de expresión. Realmente el problema aquí era el manejar el tamaño de la ecuación más que su complejidad.

Una de las ventajas que tiene muSIMP es que almacena su información en forma de listas y existen formas para ir recorriendo éstas. De aquí surgió la idea de ir recorriendo la expresión nodo por nodo hasta llegar a una expresión cuyo tamaño no fuera muy grande, por decir una de longitud aproximadamente de 100 caracteres era muy buen tamaño para ser manejada por REDUCE. Una vez encontrada una expresión de este tamaño se aplicaría REDUCE y así sucesivamente se recorrería toda la lista nodo por nodo hasta llegar al final de la misma.

De esta manera surgió ANALIZA que es la que va controlando los niveles de profundida con ayuda de una subrutina ANALIZA1, una vez que llega al nodo cuya longitud no sea mayor de 100 aplica REDUCE.

Se empezaron a hacer las primera pruebas con la matriz arriba mencionada invertida, cuyos elementos eran demasiado grandes. Se iba tomando elemento por elemento de la matriz y se daban como argumento a ANALIZA, los resultados arrojados eran mucho más chicos que los originales.

Se procedió a hacer la comprobación multiplicando los elementos reducidos de la matriz inversa con la matriz original y se esperaba obtener la matriz identidad, pero el resultado fue diferente. Esto quería decir que había un error, si no era al momento de hacer los cambios de los elementos reducidos era en el manejo de los niveles de profundidad de la expresion.

Se siguió el desarrollo de ANALIZA por medio de un "trace". Un "trace" es una facilidad que proporciona muSIMP para hacer un seguimiento de las funciones que va evaluando. Este "trace" se mandó imprimir para checar paso a paso los resultados que se iban obteniendo y ver que era lo que no estaba funcionando bien.

Se comprobó entonces que el error estaba al manejar los niveles de profundidad. Una vez que se corrigió el error se procedió a volver a probar ANALIZA. Se le dieron los mismos elementos de la matriz inversa y los resultado obtenidos fueron diferentes a los primeros.

Estos nuevos resultados eran como una tercera parte de los originales, lo cual era aún una reducción bastante buena.

Se procedió a hacer la comprobación, o sea a multiplicar la nueva matriz inversa reducida por la matriz original y se obtuvó una matriz diferente a la identidad. Se pensó que posiblemente, debido a que los elementos obtenidos de esta multiplicación eran grandes, no estaba clara la reducción.

Se procedió a aplicar ANALIZA a cada uno de los elementos del resultado de la multiplicación de las matrices y efectivamente se íba obteniendo la matriz identidad.

La definición de ANALIZA, así como la reducción de la matriz inversa se muestran en el apéndice C.

Cuando se realizó la prueba para una matriz de 4x4 con los siguientes elementos:

$$\begin{pmatrix} \frac{1.0}{1-4s} & \frac{0.7}{1+5s} & \frac{0.3}{1-5s} & \frac{0.2}{1-5s} \\ \frac{0.6}{1-5s} & \frac{1.0}{1+4s} & \frac{0.4}{1+5s} & \frac{0.35}{1-5s} \\ \frac{0.35}{1+5s} & \frac{0.4}{1+5s} & \frac{1.0}{1-4s} & \frac{0.6}{1-5s} \\ \frac{0.2}{1+5s} & \frac{0.3}{1+5s} & \frac{0.7}{1-5s} & \frac{1.0}{1-4s} \end{pmatrix}$$

Los resultados obtenidos fueron exageradamente grandes, tan grandes que el tiempo que se tardó la máquina en arrojar el resultado fue aproximadamente de dos horas. Se trató de reducir cada uno de los elementos por medio de ANALIZA, pero no hubo suficiente capacidad de memoria lo cual era de esperarse.

Se realizó un programa para inversión de matrices intentando optimizar el ya existente, pero no se tuvo éxito, ya que los resultados obtenidos diferían muy poco con respecto al tamaño de los obtenidos con muMATH. Esto es debido a que al hacer inversiones de matrices simbólicas, los resultados son muy grandes, a menos que se tengan muchos términos comunes o que contengan muchos ceros los elementos de la matriz.

Por el momento, lo único que se puede hacer es restringir la inversión de matrices simbólica a no ser mayor de una de 3x3 para poder manejar las expresiones o en su defecto extender la capacidad de memoria. Para el proyecto donde surgió la necesidad de aplicar inversión de matrices, generalmente se trabaja con matrices de 2x2.

Una vez teniendo un mejor programa para reducción de expresiones se procedió a utilizar a ANALIZA en lugar de REDUCE en diferentes tipos de ecuaciones. En algunos casos aplicados, se notaron ciertas diferencias de factorizaciones hechas con ANALIZA y las obtenidas manualmente al comparar resultados. Estas diferencias se ejemplificaran enseguida para conocer el tipo de factorización que ANALIZA realiza.

La ecuación a manejar es:

$2T1 * (2((A - (1 - A)\,'2)\ (-1 - A)\ \hat{}3 - (1 - (A - (1 - A)\ 2)\ (A - B1))\ (-1 - A)\hat{}2) - 2$
$(1 - (A - (1 - A)\ 2)/(A - B1))\ (-1 - A)\hat{}2 - 2\ ((-1 - A)\ (A - B1))) - 32\ T2\ (-1 - A)\hat{}2$
$- 4T3 * (A - B2)(A - (1 - A)\ 2)\ ((1 - A)\ \hat{}3\ (1 - B2))$

El resultado de ANALIZA al reducir la expresión anterior fue:

(-8 + 18A - 10B1)/((-1 + A)^2 (A - B1)) T1 + T2 32 (-1 + A)^2 - T3 (2 (-1 + A) (A - B2))/(1 - A)^3 (1 - B2)

el cual acomodado de manera diferente para poder compararlo con el obtenido manualmente es:

$$\frac{-8-18A-10B1}{(-1-A)^2(A-B1)}T1 - \frac{32}{(-1+A)^2}T2 - \frac{2(-1+A)(A-B2)}{(1-A)^3(1-B2)}T3$$

El obtenido manualmente fue:

$$(\frac{10}{(A-1)^2} - \frac{8}{(A-1)(A-B1)})T1 - \frac{32}{(A-1)^2}T2 - \frac{2(A-B2)}{(1-A)^2(1-B2)}T3$$

En el primer término. la única diferencia es que ANALIZA obtuvo un común denominador pero en sí la expresión es la misma. En la segunda expresión no hay ninguna diferencia. En la tercera expresión la obtenida manualmente está mejor factorizada pero por un cambio de signos que se hizo. es decir se tiene:

$$+\frac{2(-1+A)(A-B2)}{(1-A)^3(1-B2)}T3$$

si se cambia de signo la expresión en el numerador (-1−A) por (1-A) y se cambia de signo a la expresión completa para no alterarla. esto favorece para su reducción, es decir:

$$-\frac{2(1-A)(A-B2)}{(1-A)^3(1-B2)}T3$$

de esta manera se elimina el termino común (1-A) y queda la expresión:

$$-\frac{2(A-B2)}{(1-A)^2(1-B2)}$$

que es el resultado obtenido manualmente. Este tipo de factorización ANALIZA no está capacitado a resolver ni algun otro manejo de éste tipo. ANALIZA simplementa reacomoda de diferentes maneras los términos con los que cuenta. elimina términos comunes y escoge la expresion más corta.

Sin embargo. se podría implementar éste tipo de factorizacion pero el cual repercutiría en un mayor tiempo de ejecución y en memoria. Para los intereses creados. el tipo de factorización que realiza ANALIZA es satisfactorio.

3.2.6 Memoria

El problema de restricción de memoria no es grave. al menos para la mayoria de los casos. Lo único que hay que tener cuidado es en la construccion de ambientes específicos. es decir.

para lograr un ambiente se tienen que ir cargando los paquetes muMATH como ya se explicó en el Capítulo 2.

Si se crea un ambiente muy grande o sea que contenga muchos paquetes de muMATH, va a consumir mucha memoria y por consecuencia reduce el área de trabajo. Por ésto se recomienda la construcción de ambientes únicamente con lo necesario a los fines específicos.

Por otro lado no se pueden cargar dos ambientes al mismo tiempo. es decir no se pueden hacer dos o más LOAD para crear un nuevo ambiente, ya que quedaría el ambiente del último LOAD que se hizo. Lo que se puede hacer, cuando se quiere construir un sistema con paquetes de niveles más altos. es cargar un sistema construido parcialmente con el comando LOAD y el resto de los paquetes leerlos con el comando RDS. lo cual ahorra mucho tiempo.

Un problema de memoria que se encontró y que no se pudo resolver fue. para la reducción de los elementos de matrices inversas mayores de 3x3 ya que sus elementos eran demasiado grandes y no hubo suficiente capacidad de memoria.

3.3 Aplicaciones Prácticas de muMATH en el área de Simulación

3.3.1 Introducción

En esta sección se desarrollará a fondo las aplicaciones de muMATH en el área de Simulación. Se subdividirá la sección por nombre de proyecto. Cada proyecto contendrá a la vez el objetivo de éste. la aplicación realizada con muMATH y las tendencias posibles en un futuro.

3.3.2 PROYECTO 1:

Determinación de Curvas de Arranque de Plantas
atendiendo a esfuerzos térmicos

OBJETIVO:

Optimizar la curva de arranque de una turbina específicamente. Para el arranque de una turbina, se tiene cierta curva predeterminada que describe las temperaturas en que la turbina debe funcionar con respecto al tiempo.

Desde el inicio de arranque de una turbina (14 rpm) hasta la carga plena (300 MW) se lleva un tiempo aproximado de 8 a 12 hrs. Esta curva de arranque es proporcionada al momento de la compra del equipo y existen altas probabilidades de que el tiempo se pueda reducir sin dañar por otro lado el equipo. Los beneficios son bastante provechosos lo que justificó los gastos necesarios para la realizacion de este proyecto.

DESARROLLO:

Una de las ecuaciones con las que se trabaja en el proyecto tiene la forma:

$$\frac{\partial T}{\partial \tau} = \frac{\partial^2 T}{\partial R^2} + \frac{1}{R}\frac{\partial T}{\partial R}$$

la cual modela el fenómeno de conducción de calor en sólidos en la dirección radial de un cilindro de pared gruesa. Para resolver este modelo se hace una aproximación por polinomios [2] que cumplan ciertas condiciones. La distribución de temperaturas se aproxima de la siguiente forma:

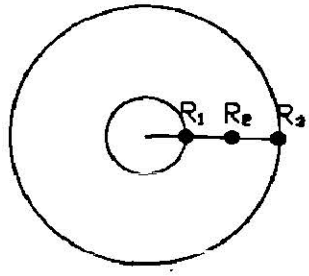
$$T = P_1(r) * T_1 + P_2(r) * T_2 + P_3(r) * T_3 + \cdots + \tag{3.1}$$

los cuales se satisfacen para cualquier valor de r. En el caso de tres estaciones, que es con el que se han empezado a hacer pruebas, se tomaron los siguientes valores de r :

$R_1 = A$

$R_2 = \frac{1+A}{2}$

$R_3 = 1$



con las condiciones iniciales de frontera:

$$\left|\frac{\partial T}{\partial R_1}\right|_{R_1=A} = 0 \qquad \left|\frac{\partial T_3}{\partial R_3}\right|_{R_3=1} = -\beta_2(T_3 - TS_1)$$

en donde los polinomios toman la forma:

$$P_1(r) = \frac{(r-\frac{1+a}{2})(r-1)^2(r-b_1)}{(\frac{a-1}{2})(a-1)^2(a-b_1)}$$

$$P_2(r) = \frac{16(r-a)^2(r-1)^2}{(a-1)^4}$$

$$P_3(r) = \frac{(r-a)^2(r-\frac{1+a}{2})(r-b_2)}{(1-a)^2(\frac{1-a}{2})(1-b2)}$$

al substituir en la ecuación 3.1 queda:

$$T = \frac{(r-\frac{1-a}{2})(r-1)^2(r-b_1)}{(\frac{a-1}{2})(a-1)^2(a-b_1)} - \frac{16(r-a)^2(r-1)^2}{(a-1)^4} - \frac{(r-a)^2(r-\frac{1-a}{2})(r-b_2)}{(1-a)^2(\frac{1-a}{2})(1-b2)}$$

en seguida se procede a obtener la primera y segunda derivada de T con respecto a "r" para representar el modelo y substituir para "r" los tres valores correspondientes. esto es:

$$\frac{\partial T_1}{\partial R_1} = \left|\frac{\partial^2 T_1}{\partial R_1^2}\right|_{R_1=A} - \left|\frac{1}{R_1}\frac{\partial T_1}{\partial R_1}\right|_{R_1=A}$$

$$\frac{\partial T_2}{\partial R_2} = \left|\frac{\partial^2 T_2}{\partial R_2^2}\right|_{R_2=\frac{1+A}{2}} - \left|\frac{1}{R_2}\frac{\partial T_2}{\partial R_2}\right|_{R_2=\frac{1-A}{2}}$$

$$\frac{\partial T_3}{\partial R_3} = \left|\frac{\partial^2 T_3}{\partial R_3^2}\right|_{R_3=1} - \left|\frac{1}{R_3}\frac{\partial T_3}{\partial R_3}\right|_{R_3=1}$$

de tal manera que la ecuación original se transforma en un sistema de tres ecuaciones con tres incógnitas de esta naturaleza:

$$\frac{\partial T_1}{\partial \tau} = \frac{-22}{(1-a)^2}T_1 - \frac{32}{(1-a)^2}T_2 - \left[\frac{10}{(1-a)^2} - \frac{2}{1-a}\beta_2\right]T_3 - \frac{2}{1-a}\beta_2 TS$$

$$\frac{\partial T_2}{\partial \tau} = \left[\frac{8}{(1-a)^2} - \frac{3}{1-a^2}\right]T_1 - \frac{16}{(1-a)^2}T_2 - \left[\frac{8}{(1-a)^2} - \frac{3}{1-a^2} - \frac{(3-a)\beta_2}{2(1-a^2)}\right]T_3 - \frac{3-a}{2(1-a^2)}\beta_2 TS$$

$$\frac{\partial T_3}{\partial \tau} = \frac{-10}{(1-a)^2}T_1 - \frac{32}{(1-a)^2}T_2 - \left[\frac{26}{(1-a)^2} - \frac{(9-a)\beta_2}{1-a}\right]T_3 - \frac{9-a}{1-a}\beta_2 TS$$

Este sistema se resuelve para ciertas condiciones iniciales de: a, β_2 y TS que representan respectivamente el radio interno del cilindro. la temperatura inicial interna del cilindro y la temperatura ambiente.

Una vez que se substituyen estos valores, se obtiene un sistema de ecuaciones de la siguiente forma:

$$\begin{array}{rrrr} -88.00T_1 & +128.00T_2 & -60.00T_3 & +180.00 \\ +28.00T_1 & -64.00T_2 & -44.33T_3 & -75.00 \\ -40.00T_1 & +128.00T_2 & -173.00T_3 & -765.00 \end{array}$$

el cual se resuelve para diferentes tiempos. en donde para cada valor se van conociendo las temperaturas en la parte externa del cilindro (T_1). en la parte media (T_2) y en el interior del cilindro (T_3).

Estos diferentes valores que se van adquiriendo. se van pasando a un paquete de graficación en donde se va a ir obteniendo la curva de distribución de temperaturas y de allí se obtiene la curva de distribución de esfuerzos que es la meta final.

La curva de distribución de esfuerzos va a ir señalando la temperatura en que la turbina debe estar funcionando contra el tiempo.

APLICACION:

La manera en que MuMATH interviene en este proyecto es que originalmente el desarrollo para obtener el sistema de ecuaciones se realizaba manualmente. es decir; se obtenía la primera y segunda derivada. se substituían cada uno de los diferentes valores de "r" y finalmente se trataba de reducir la expresión lo más posible. El problema crecía cuando el número de estaciones aumentaba a cinco porque se obtenían polinomios mas grandes:

$$P_1(r) = \frac{32(r-1)^2(r-\frac{1+3a}{4})(r-\frac{1-a}{2})(r-\frac{3+a}{4})(r-b1)}{3(a-1)^5(a-b_1)}$$

$$P_2(r) = \frac{2048(r-1)^2(r-a)^2(r-\frac{1-a}{2})(r-\frac{3-a}{4})}{9(a-1)^6}$$

$$P_3(r) = \frac{256(r-1)^2(r-a)^2(r-\frac{1+3a}{4})(r-\frac{3-a}{4})}{-(a-1)^6}$$

$$P_4(r) = \frac{2048(r-1)^2(r-a)(r-\frac{1+3a}{4})(r-\frac{1-a}{2})}{9(a-1)^6}$$

$$P_5 = \frac{32(r-a)^2(r-\frac{1-3a}{4})(r-\frac{1-a}{2})(r-\frac{3-a}{4})(r-b2)}{3(1-a)^5(1-b_2)}$$

el cual no era nada fácil obtener sus derivadas parciales. substituir para cada valor de "r" y construir el sistema de ecuaciones.

Se creó entonces un programa en MuSIMP que con las facilidades que proporcionaba muMÀTH derivara y substituyera en los polinomios.

Los resultados obtenidos no eran óptimamente factorizados. Precisamente aquí fue donde surgió el primer problema comentado en la sección correspondiente.

Para las pruebas del polinomio de tres condiciones los problemas de factorización fueron resueltos con REDUCE. Aunque para aplicarlo se hizo por partes para obtener una buena reducción. Para la solución del polinomio de cinco condiciones se tenían muchos problemas de reducción. ya que las ecuaciones a manejar eran demasiado grandes.

Se procedió a crear la interfase para el polinomio de tres condiciones. primero porque ya se tenían resultados desarrollados manualmente y esto serviría para compararlos con los obtenidos con muMATH. Segundo. los problemas de reducción para el polinomio de cinco condiciones se habían resuelto superficialmente desarrollándose por partes.

Los resultados arrojados por el sistema, para el polinomio de tres condiciones fueron muy satisfactorios. inclusive se pudo detectar que en el desarrollo manual había unos pequeños errores.

Mientras que el sistema para el polinomio de tres condiciones ya estaba funcionando se seguía con problemas para el de cinco condiciones. paralelamente a ésto había surgido el problema de factorización para las matrices inversas. Fue cuando se pensó en optimizar REDUCE y surgió en su lugar ANALIZA.

Los problemas de factorización se resolvieron. pero existió otro problema al querer integrar todo en un sistema: no había suficiente capacidad de memoria.

Buscando la manera de solucionar este problema. se encontró otra forma de resolver la situación. Esta nueva forma de solución restringía en cierta manera al sistema y lo convertiría para su aplicación en casos bien específicos. Por otro lado. el beneficio obtenido era que el tiempo de ejecución era muy rápido. inclusive mejor que para el de tres condiciones. También los resultados íban a ser muy beneficiosos, ya que se esperaba que dieran una mejor aproximación del comportamiento de las curvas a graficar. Cabe mencionar que éstas pruebas no se habían podido desarrollar manualmente por el tamaño de los polinomios.

Esta nueva forma de solución para el caso del polinomio de cinco condiciones, a diferencia del de tres condiciones, es que las condiciones iniciales de frontera únicamente son para cuatro casos diferentes. En el de tres condiciones puede darse cualquier condicion de frontera. A partir de éstas condiciones. se calculan las constantes B1 y B2 que forman parte de las variables de los polinomios. Estas constantes si se calculan para el caso del polinomio de tres condiciones. para el caso de cinco condiciones éstas constantes se asignan. dependiendo del valor de la condición inicial de frontera dado. Este cambio fue necesario por la cantidad de memoria que se consume en el cálculo de las constantes.

Otro cambio en el programa para el polinomio de cinco condiciones es que las constantes B1 y B2 no son desplegadas en pantalla como en el caso del polinomio de tres condiciones. También otra diferencia. es que en el programa del polinomio de cinco condiciones no se hace ninguna reducción intermedia de las expresiones a manejar. es decir no se manda llamar a ANALIZA para que reduzca los términos. ya que ésto consume mucha memoria y tiempo. Por el contrario. se manejan los términos grandes y hasta que no se substituyen los datos iniciales de las variables: a. β_1, β_2, TS, etc.,que hacen que la expresión se reduzca un poco. entonces se aplica ANALIZA.

Al substituir estos datos iniciales en el polinomio de cinco condiciones se utiliza la instrucción EVAL, ya que al leer los datos directamente, se dejan en las variables correspondientes a substituir y únicamente se la da un EVAL a la expresión que contenga estas variables para que la evalúe con estos datos. A diferencia del programa para el polinomio de tres condiciones. en donde al leer los datos iniciales se dejan en unas variables de paso. donde después se substituyen estos valores con la instrucción EVSUB por las variables correspondientes: a. β_1, β_2, TS, etc..en la expresión a evaluar. La diferencia de éstas dos formas es que EVSUB consume más memoria que EVAL ya que evalúa y substituye.

Una vez que se checó que todo funcionara bien. se integró esta parte al sistema del polinomio de tres condiciones. Ahora el sistema funcionaba para cualquiera de los dos casos: para un polinomio de tres condiciones o de cinco condiciones. El usuario introduce solamente las condiciones iniciales de frontera y obtiene de resultado el sistema de ecuaciones.

En el apéndice D se encuentra todo el desarrollo de este sistema con comentarios.

TENDENCIAS:

Se piensa que MuMATH también pueda resolver el sistema de ecuaciones obtenidas. pero se tendrían que crear ciertas funciones específicas para ello. Si ésto es posible se haría una interfase más directa de muMATH con el graficador de la distribución de temperaturas. Esto sería muy beneficioso ya que los planes son obtener en tiempo real las gráficas del comportamiento de la turbina.

3.3.3 PROYECTO 2:

Control Multivariable de una unidad termoeléctrica
empleando el arreglo inverso de Nyquist.

OBJETIVO:

Lograr desacoplar las variables de tal manera que al modificar una variable de entrada para alterar su correspondiente variable de salida las otras no sean afectadas. Es decir se quiere lograr una correspondencia uno a uno entre variables de control y variables de salida.

DESARROLLO:

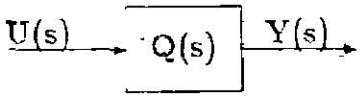
El análisis de un sistema modelado utilizando el arreglo inverso de Nyquist [4] se basa en una representación de funciones de transferencia[1] de la siguiente manera:

$$Y(s) = Q(s)U(s) \tag{3.2}$$

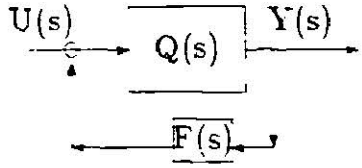
donde:

$$q_{ij} \quad \frac{a_0^{ij} - a_1^{ij} s - \cdots + a_n^{ij} s^{n^{ij}}}{b_0^{ij} - b_1^{ij} s - \cdots + b_m^{ij} s^{m^{ij}}} \qquad \begin{array}{l} \forall a_k^{ij} \quad cte.\ real\ y\ k = 1.2.\cdots, n \\ \forall b_k^{ij} \quad cte.\ real\ y\ k = 1.2.\cdots, n \end{array}$$

Gráficamente este modelo se puede representar por:



en donde se requiere de un compensador F(s) para mejorar la respuesta dinámica del sistema.



Esta nueva representación de transferencia para la estructura de compensación viene dada por la función:

$$H(s) = (I - Q(s)F(s))^{-1} \quad Q(s)$$

La forma más simple para el análisis de esta función es obtener la inversa de la ecuación anterior, resultando:

[1] Una función de Transferencia es una ecuación matemática que relaciona a la entrada y la salida de cualquier proceso

$$\hat{H}(s) - \hat{Q}(s) - F(s)$$

en donde gráficamente ésto sería:

$$\underline{Y(s)} \longrightarrow H(s) \xrightarrow{U(s)}$$

El problema aqui es obtener $\hat{Q}(s)$ a partir de una Q(s) que puede tener la siguiente forma:

$$\begin{pmatrix} \frac{806s-.264}{s^2+1.15s+.202} & \frac{-(15.0s-1.42)}{s^3+12.8s^2-13.6s-2.36} \\ \frac{1.95s^2-2.12s-.490}{s^3+9.15s^2+9.39s-1.62} & \frac{7.14s^2-25.8s+9.35}{s^4+20.8s^3-116.4s^2-111.6s-18.8} \end{pmatrix} \tag{3.3}$$

Para la obtencion de $\hat{Q}(s)$ que consiste en invertir la matriz descrita se emplearon varias técnicas ya que no se contaba con un sistema que pudiera invertir matrices simbólicamente.

Se partió de una forma de representación llamada de variables de estado. Esto es, el sistema modelado representado en variables de estado es de la forma:

$$\dot{X}(t) = AX(t) + Bu(t) \tag{3.4}$$

$$Y(t) = CX(t) + Du(t) \tag{3.5}$$

donde:

$X(t)$	*es el vector de estados* $\in R^n$
A	*es una matriz de nxn real e independiente de t*
B	*es una matriz de nxm*
C	*es una matriz de lxn*
D	*es una matriz de lxm*
$u(t)$	*es el vector de variables de control* $\in R^n$
$Y(t)$	*es el vector de variables de salida* $\in R^r$

Sobre esta nueva representación se aplican una serie de procedimientos para lograr un sistema invertido en variables de estado de la siguiente forma.

$$\dot{Z}(t) = \hat{A}Z(t) + \hat{B}Y(t) \tag{3.6}$$

$$u(t) = \hat{C}Z(t) - \hat{D}Y(t) \tag{3.7}$$

Ahora como paso final, queda representar este sistema invertido en variables de estado a una función de transferencia $\hat{Q}(s)$. Para ésto se aplica en 3.6 y 3.7 la transformada de Laplace quedando:

$$SZ(s) = \hat{A}Z(s) - \hat{B}Y(s) \tag{3.8}$$

$$U(s) = \hat{C}Z(s) - \hat{D}Y(s) \tag{3.9}$$

donde despejando a Z de 3.8 nos queda:

$$Z = (sI - \hat{A})^{-1}BY(s)$$

sustituyendo en 3.9:

$$U(s) = \hat{C}(sI - \hat{A})^{-1}\hat{B}Y(s) - \hat{D}Y(s)$$

factorizando queda:

$$U = [\hat{C}(sI - \hat{A})^{-1}\hat{B} - \hat{D}]Y(s) \tag{3.10}$$

La ecuación 3.10 nos muestra ya a $\hat{Q}$ que es la relación:

$$[\hat{C}(sI - A)^{-1}\hat{B} - \hat{D}$$

en donde ahora hay que resolver $(sI - \hat{A})^{-1}$ por medio de otro método ya que se vuelve a caer en el mismo problema de invertir una matriz simbólica pero la cual es mucho menos compleja que la inicial ya que "I" es la matriz identidad. "s" es una variable y "A" es numérica.

El método que se aplica es el de Leverrier-Souriau [3] para resolver a $(sI - A)^{-1}$ llamada la resolvente de A. esto es:

$$(sI - A)^{-1} = \frac{adj(sI - A)}{det(sI - A)}$$

donde $det(sI_n - A)$ es llamado el polinomial característico de A y es igual a:

$$s^n - d_1 s^{n-1} - \cdots - d_{n-1}s - d_n$$

y $adj(sI_n - A)$ es igual a:

$$R_1 S^{n-1} + R_2 S^{n-2} + \cdots + R_{n-1} S + R_n$$

donde los coeficientes R_i y d_i (i = 1.2,..,n) pueden encontrarse con la relación recursiva:

$$\begin{array}{llll} R_1 = \sum n & & d_1 = -tr(R_1 A) \\ R_2 = R_1 A - d_1 I_n & & d_2 = \frac{-1}{2} tr(R_2 A) \\ & & \vdots \\ R_{n-1} = R_{n-2} A - d_{n-2} I_n & & d_{n-1} = \frac{-1}{n-1} tr(R_{n-1} A) \\ R_n = R_{n-1} A - d_{n-1} I_n & & d_n = \frac{-1}{n} tr(R_n A) \\ R_{n+1} = R_n A - d_n I_n = 0 & & \end{array}$$

El "trace" de A denotado por tr(A) siendo A una matriz de nxn es la suma de los elementos de la diagonal principal de A esto es:

$$tr(A) = \sum_{i=1}^{n} a_{ij}$$

Resueltas estas matrices se procede a obtener la deseada $\hat{Q}(s)$ invertida la cual se substituye en la ecuación:

$$\hat{H}(s) = \hat{Q}(s) + F(s)$$

Una vez obtenida $\hat{H}(s)$ se aplica la metodología del arreglo inverso de Nyquist para el desacoplamiento y compensación del sistema.

Resumiendo todo el proceso teníamos inicialmente una Q(s) en donde para invertirla para obtener $\hat{H}(s)$ se hace un cambio a variables de estado (A B C D) donde se aplica una serie de procedimientos según Rosenbrock para obtener un sistema invertido en variables de estado $(\hat{A}\hat{B}\hat{C}\hat{D})$ en donde aquí para poder resolver a $(sI - \hat{A})^{-1}$ se aplica otra metodología de Leverrier que finalmente resuelve esta inversión de matriz simbólica que viene siendo la $\hat{Q}(s)$ anhelada.

APLICACIONES:

Como se ve se aplica toda una serie de metodologías para poder invertir una matriz simbólica en donde ésto solo viene siendo un paso para poder aplicar finalmente el próposito del proyecto que es el desacoplamiento del sistema por medio del arreglo inverso de Nyquist

MuMATH invierte matrices simbolicamente y en un tiempo reducido. El problema que se presentó, ya comentado en la sección de problemas encontrados, es que al invertir matrices simbólicas mayores de 3x3 se tiene problemas al manejar sus elementos ya que son demasiados grandes y no hay forma de poder reducir las expresiones ya que no hay suficiente capacidad de memoria.

Al menos para esta aplicación, por lo general se trabajan con matrices de 2x2 o 3x3 en donde los elementos obtenidos de la matriz inversa se les aplica ANALIZA para la reducción de sus términos. la cual es bastante buena. En el apéndice C se encuentra la matriz 3.3 inversa con sus elementos reducidos.

TENDENCIAS:

Las tendencias es hacer una interfase amigable para matrices no mayores de 3x3 con la ocpión de poder invertirlas u obtener sus derivadas parciales. También otra opción en esta interfase sería el obtener un analisis de estabilidad para cualquier expresión factorizada.

Capítulo 4

CONCLUSIONES

4.1 Líneas de Investigación

Existe toda una amplia gamma de aplicaciones en el área de la manipulación simbólica, desde obtener un simple resultado simbólico o el de realizar cierta programación simbólica para obtener resultados específicos hasta toda una aplicación formal como es el caso donde se aplica a la Inteligencia Artificial. Precisamente, el lenguaje muSIMP que es con el que está escrito muMATH esta basado en el lenguaje típico de Inteligencia Artificial LISP pero con una sintáxis más natural.

Dentro del área de Análisis Numérico en el Depto. de Simulación se utiliza mucho el análisis de estabilidad de cualquier expresión factorizada, esto es debido a que existen problemas fuertes al momento de graficar esta expresión, ya que para cierto rango de valores se dispara y es que matemáticamente no es igual una expresion original a una factorizada. Se tiene en proyecto realizar un sistema, el cual pueda factorizar cualquier tipo de expresión simbólica y ademas obtenga el rango de estabilidad de la misma para evitar problemas al momento de gráficar ésta.

Una aplicación fuerte de manipulación simbólica en el Depto. de Simulación en el Instituto de Investigaciones Eléctricas, es un sistema de programación automática para el desarrollo de simuladores, en el cual un módulo de este sistema estará interactuando con muMATH para todo lo referente a la manipulación algebraica.

También se están desarrollando sistemas expertos[1] en el sector educativo, los cuales enseñan como resolver problemas algebraicos aplicando métodos tradicionales de solución. ésto permite que el estudiante vaya comprendiendo mejor la aplicacion del método.

[1] Un sistema experto es un programa de computadora que resuelve ciertos problemas específicos, como si fuera un experto humano.

4.2 Conclusiones

Resumiendo. se puede decir que los resultados obtenidos fueron satisfactorios en las diferentes áreas aplicadas. Se tuvo sin embargo, que enfrentar varios tipos de problemas de los cuales se libraron los principales para obtener resultados finales.

Los programas simbólicos creados para poder satisfacer las diferentes necesidades fueron: LONGITUD y ANALIZA así como un sistema amigable para la solución de un sistema de ecuaciones parciales.

Los problemas presentados fueron primeramente por memoria insuficiente al tratar de reducir expresiones demasiado grandes por medio de ANALIZA. Este tipo de expresiones tan grandes se pueden obtener únicamente al invertir una matriz mayor de 3x3, para cualquier otro tipo de expresion no existió este problema. Una posible solución a este problema es trabajar en una computadora con mayor capacidad de memoria principal. tomando encuenta que el tiempo que se lleve ANALIZA en reducir sus elementos va a ser grande.

Otros problemas presentados fueron para el desarrollo del sistema amigable, más que nada en la realización de lecturas por medio de pantallas. Estos problemas fueron debidas a presentación y uniformidad para poder dar mayor facilidad al usuario. muMATH no es muy amigable. pero cuenta con las funciones básicas requeridas para éste tipo de desarrollos.

El tipo de problema genérico mas fuerte solucionado fue la factorización por medio de ANALIZA, ya que sin éste las aplicaciones realizadas en los dos proyectos mencionados en las aplicaciones prácticas hubieran sido imposibles.

Otra muy importante aplicación de ANALIZA es en la reducción de matrices inversas no mayores de 3x3. que aunque restringida esta aplicación tiene un uso muy importante, una de ellas ya comentada en el segundo proyecto. el cual ahorra toda una serie de procedimientos al modelador de sistemas de control.

Hay muchas otras aplicaciones. desde una simple interacción directa como el obtener una derivada o una integral hasta toda una interacción dentro de sistemas expertos. El único problema para una aplicación más fuerte en éste paquete sería la memoria pero se puede resolver expandiendo la capacidad de ésta o bien usando un sistema más potente para la manipulación simbólica como es MACSYMA.

Una ventaja fuerte que presenta éste paquete contra los problema encontrados es que tiene su propio lenguaje muSIMP. lo cual lo hace versátil ya que se puede expander según las necesidades, puede servir para fines educativos. no ocupa mucha memoria. corre en PC's y su costo no es muy elevado.

Bibliografía

[1] *" Handbook of Artificial Intelligent"*.
Avron Barr & Edward A. Feigenbaum. Kaugman. Los ALtos, Calif., 1981.

[2] *"Approximate Solution to Thermal-Shock Problems in Plates, Hallow Spheres, and Cylinders with Heat Transfer at Two Surfaces"*.
A. Mendelson & S.S Manson. ASME (American Society Mechanical Engineers). Cleveland. Ohio. 1954.

[3] *"Multivariable System Theory and Design"*.
R.V. Patel & Neil Munro, Pergamon Press. 1982.

[4] *"Computer-Aided Control System Design"*.
H.H. Rosenbrock, Academic Press. 1974.

[5] *"Microsoft MUMATH Symbolic Mathematics Package for MS-DOS Operating System"*,
David Stoutemyer & Albert Rich. 1976.

Encuadernación EL MODELO
Diego de Montemayor 904 Nte. y Arteaga
Teléfono 74-62-37