UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA EN CIENCIAS DE LA TIERRA

"PROGRAMACIÓN AVANZADA PARA INGENIEROS PETROLEROS"

T E S I S
QUE PARA OBTENER EL TITULO DE:
I N G E N I E R O P E T R O L E R O
P R E S E N T A:

JUAN CARLOS SABIDO ALCÁNTARA

DIRECTOR DE TESIS
INGENIERO GUILLERMO TREJO REYES

CIUDAD UNIVERSITARIA, MÉXICO D.F. OCTUBRE 2007

Los débiles no luchan, los más fuertes luchan una hora, los que aún son más fuertes, luchan unos años, pero, los más fuertes de todos luchan toda su vida. Estos son los indispensables.

Bertolt Brech (1896 – 1956)

Dedicatorias y Agradecimientos.

*A ti **MAMÁ ROSA** por todo tú apoyo, esfuerzo, dedicación, comprensión y amor, te agradezco lo que te haz esforzado, te agradezco tus incontables horas de trabajo para darme un lugar donde dormir y un plato para comer, te agradezco el que a pesar del dolor nunca te has rendido, te agradezco que a pesar de las presiones y de las críticas nunca dejaste de apoyarme y creer en mi, te agradezco que seas mi **MADRE**, eres la mejor y no podría querer a nadie más para que ocupe ese lugar, este logro te lo dedico porque simplemente sin ti nunca hubiera ocurrido, **GRACIAS**. Lo logramos **MAMÁ, TE QUIERO MUCHO**.*

*A ti **PAPÁ**, que aunque hace diez y seis años falleciste siempre has estado a mi lado con tus enseñanzas, tus regaños y tus lecciones, la más importante, siempre levantarme por fuerte que sea el golpe, espero que desde donde estés **PAPÁ** nos observes y te sigas sintiendo muy orgulloso. Te sigo extrañando.*

*A ti **MARLENE**, por estar conmigo tanto tiempo, por ayudarme siempre que has tenido la oportunidad de hacerlo, por soportar los malos ratos, por ser paciente, por comprender mi situación económica y nunca exigir nada, por perdonarme tantos errores, por ser amable, cariñosa y amorosa, por apoyarme en las decisiones que tomo y solapar mis ocurrencias y desplantes. No te preocupes, siempre encontraremos la manera de salir adelante y seguir juntos, **GRACIAS** por amarme tanto.*

*A ti **PATRICIA**, por ser mi hermana, porque aunque al principio no estabas muy de acuerdo en que siguiera estudiando terminaste aceptándolo y brindándome todo tu apoyo, gracias por esos dos sobrinos que tanto quiero y por los que daría todo. Lo logre **GORDA**.*

*A ti **NATALIA**, por llegar a mi vida, cuando te vi por primera vez supe que te iba a querer siempre, gracias por hacerme reír, por quererme tanto, por bautizarme con mi nombre de **KON**, te quiero mucho **GORDITA**.*

*A ti **JESÚS**, por estar conmigo, por quererme tanto, y por ser tan loco como eres, te quiero **QUECHUCHO**.*

*A ti **TÍO MIGUEL**, por siempre querernos tanto a mi hermana y a mí, por apoyarme y respetarme en mis decisiones, **GRACIAS** tío.*

*A ti **TÍO FERNANDO**, que al igual que mi tío Miguel siempre me has respetado.*

*A ti **ERIC RAMÍREZ VARGAS** (Ramvar), porque tu obsequio fue de gran importancia para desarrollar las habilidades y el conocimiento que permitió desarrollar este trabajo, **GRACIAS**.*

*A ti **RICARDO CHAGOYA**, por siempre apoyarme, ofrecerme tu ayuda, prestarme tus apuntes y copias para así ahorrar un poco de dinero y por todo el aprendizaje que juntos adquirimos a lo largo de nuestra estancia en la Facultad, sobre todo por tu amistad, ya sonríe, tiriri tiriri tiriri, **GRACIAS**.*

*A ti **JORGE IVÁN RUÍZ GASTELUM**, por ser como eres, por creer en mí, por tus recomendaciones para obtener trabajo, por tu ayuda, apoyo y sobre todo tu amistad, **GRACIAS**.*

*A ti **VÍCTOR RICHAUD** por haber sido tan amble conmigo, tener tantas atenciones y tan buenas intenciones siempre, **GRACIAS**.*

*A ti **JOSÉ LUÍS HERNÁNDEZ**, por tu apoyo para la realización de este trabajo, sin tu ayuda hubiera sido más difícil, **GRACIAS**.*
*A ti **RICARDO SERRANO ALQUICIRA**, por tu amistad y tu apoyo, por las molestias que te has tomado, por siempre acordarte de mí cuando hay alguna oportunidad, **GRACIAS**.*

*A ti **MIGUEL QUIROZ** y a ti **DIANCO SANTOS** por apoyarme en los momentos más difíciles que viví durante mi estancia dentro de la Facultad, **GRACIAS**.*

*A ti **MARCOS EDUARDO TORRES**, por seguir acordándote de mí a pesar del tiempo y la lejanía, nos sigue esperando el paracaídas.*
*A ti **PEDRO FLORES**, por los sobrinos que me regalaste, por soportar los malos modos y aún así seguir siendo amable, **GRACIAS**.*

*A ti **LEOBARDO HERNÁNDEZ VALDIVIA** por tu amistad y apoyo, **GRACIAS.***

*A usted señora **GRACIELA CASTILLO**, por recibirme en su casa, por ser tan amable y atenta, por su apoyo y confianza, por su hija, **GRACIAS**.*

*A usted señor **ROLANDO CASTILLO**, por aceptarme sin preguntas ni condiciones, por recibirme en su casa, **GRACIAS.***

*Al **INGENIERO MARTÍN CARLOS VELÁZQUEZ FRANCO**, por ser quien inicialmente acepto dirigir este trabajo, y quien me apoyo hasta el último momento para realizar las correcciones necesarias para tener un mejor trabajo de tesis, muchas **GRACIAS**.*

*Al **INGENIERO GUILLERMO TREJO REYES**, por aceptar apoyarme tomando el lugar de director de tesis, y por todas las atenciones que ha tenido con mi persona, muchas **GRACIAS**.*

*A mis sinodales, los ingenieros Manuel Juan Villamar Vigueras, Guillermo Trejo Reyes, Mario Becerra Zepeda, Oscar Rolando Gómez Santos y Roque Francisco Riquelme Alcántar, que amablemente revisaron este trabajo y lo enriquecieron con sus observaciones, **GRACIAS**.*

A Gilberto Castillo (Místico), Omar Rubio, Antonio Santamaría (Toñito), Mariel, Juan José (Juanjo), Carlos Gavira, Walter Scott, Roberto Zúñiga, Juan Carlos Vite, Aimara Trejo, Raymundo, Jorge Rojo, Esther Rosado, Francisco Rosado, Cristian Pérez Gil, David Guzmán Arévalo, Jonathan Segura, y a todas los compañeros que sin querer omito, y que de alguna u otra forma estuvieron cerca de mi camino a lo largo de este tiempo.

*Por ultimo, quiero agradecer a todas las personas que siempre han estado en mi contra, que nunca han creído en mí, y que esperaban el momento de verme derrotado, **GRACIAS**, porque solo consiguieron hacerme mas grande y mas fuerte.*

CONTENIDO.

Página

LISTA DE FIGURAS. **XI**

LISTA DE TABLAS. **XIII**

OBJETIVO Y RESUMEN. **1**
Objetivo. 1
Resumen. 1

INTRODUCCIÓN. **3**
Definiciones Básicas. 3
Computadora. 3
Programa. 4
Lenguaje De Programación. 4
Compilador. 4
Algoritmo. 4
Diagrama De Flujo. 4

I. CONCEPTOS BÁSICOS. **5**
I.I Características de un Programa. 5
I.I.I Debe Funcionar. 5
I.I.II No Debe Tener Dificultades. 5
I.I.III Debe Ser Bien Documentado. 5
I.I.IV El Programa Debe Ser Eficiente. 5
I.II Etapas De Programación. 6
I.II.I Identificación Del Problema Y Definición Del Objetivo. 6
I.II.II Análisis Del Problema. 6
I.II.III Descripción Matemática. 6
I.II.IV Desarrollo De Solución. 6
I.II.V Algoritmo. 6
I.II.VI Prueba De Escritorio. 6
I.II.VII Construcción De Solución En Forma De Programa. 6
I.II.VIII Prueba E Interpretación. 7
I.II.IX Documentación. 7

I.III Tipos De Programación. 8
I.III.I Programación Estructurada. 9
I.III.I.I Partes De Un Programa Estructurado. 9
I.III.II Programación Orientada a Objetos. 10
I.III.II.I Mecanismos Básicos De La POO. 10
I.III.II.II Propiedades De Los Objetos. 11
I.IV Diagramas De Flujo. 11
I.IV.I Razones Para El Uso De Diagramas De Flujo. 11
I.IV.II Símbolos Utilizados En La Construcción De Diagrama De Flujo 11
I.IV.III Reglas Para La Construcción De Diagramas De Flujo. 13
I.IV.IV Etapas En La Construcción De Un Diagrama De Flujo. 14
I.V Aritmética De Las Computadoras. 14
I.V.I Sistemas De Números. 14
I.V.I.I Sistema Decimal. 14
I.V.I.II Sistema Binario. 14
I.V.I.III Sistema Hexadecimal. 15
I.V.I.IV Ejemplos. 15
I.V.II Números De Punto Fijo (Enteros o INTEGER). 15
I.V.III Números De Punto Flotante (Reales). 16
I.V.III.I Notación Científica Normalizada. 16
I.V.III.II Representación De Números Con Punto Flotante. 17
I.V.IV Machine Epsilon. 19

II. SOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES. 21

II.I Introducción. 21
II.II Conceptos Básicos. 21
II.II.I Método De Cramer. 22
II.II.II Conceptos De Norma, Número De Condición Y Estabilidad. 23
II.II.II.I Norma De Una Matriz. 23
II.II.II.II Número De Condición Y Error En La Solución. 24
II.II.II.III Estabilidad. 24
II.II.III Solución Numérica. 24
II.II.III.I Número De Operaciones Fundamentales. 24
II.II.III.II Necesidades De Almacenamiento. 24
II.II.III.III Rango De Aplicabilidad. 24
II.II.IV Métodos Directos Y Métodos Indirectos (Iterativos). 25
II.II.IV.I Métodos Directos. 25
II.II.IV.II Métodos Indirectos. 25
II.II.V Tipos De Matrices Utilizadas En Ingeniería. 25
II.II.VI Conceptos De Matriz Diagonal, Triangular Superior Y Triangular Inferior. 26
II.II.VI.I Matriz Diagonal. 26
II.II.VI.II Matriz Triangular Superior. 27
II.II.VI.III Matriz Triangular Inferior. 28
II.III Método De Eliminación Gaussiana. 28
II.III.I Algoritmo. 29

II.IV Método De Descomposición LU. 29
II.IV.I Método Doolittle. 30
II.IV.I.I Algoritmo Para Descomponer Una Matriz A En LU. 30
II.IV.I.III Algoritmo Para Solucionar Un SEL Factorizado En LU. 31
II.IV.II Método Crout. 32
II.IV.III Método Choleski. 32
II.V Método De Jacobi. 32
II.V.I Algoritmo. 33
II.V.II Ejemplo. 34
II.VI Método De Gauss – Seidel. 36
II.VI.I Algoritmo. 36
II.VI.II Convergencia Del Método De Jacobi Y Gauss - Seidel. 36
II.VII Método De Relajación Sucesiva (SOR). 36
II.VII.I Algoritmo. 36
II.VIII Subrutina NSPIV.14,15 37
II.VIII.I Algoritmo. 37
II.VIII.II Subrutina De Solución. 38

III. SOLUCIÓN DE SISTEMAS DE ECUACIONES NO LINEALES. 41

III.I Introducción. 41
III.II Conceptos Básicos. 41
III.II.I Ecuación No Lineal. 41
III.II.II Sistema De Ecuaciones No Lineales. 42
III.III Solución De Una Ecuación No Lineal. 42
III.III.I Métodos Para Resolver Una Ecuación No Lineal. 42
III.III.II Método De Newton – Raphson. 43
III.III.II.I Algoritmo. 43
III.III.II.II Ejemplo. 44
III.III.III Método De La Secante. 45
III.III.III.I Algoritmo. 45
III.IV Solución De Un Sistema De Ecuaciones No lineales. 46
III.VI.I Método De Newton – Raphson Para Sistemas De n*n. 46
III.IV.I.I Algoritmo. 46

IV. INTERPOLACIÓN NUMÉRICA 49

IV.I Introducción. 49
IV.II Lineal y Doble Interpolación. 49
IV.II.I Interpolación Lineal Para Un Conjunto De Datos Conocidos. 49
IV.II.I.I Algoritmo. 50
IV.II.I.II Ejemplo. 51
IV.II.II Interpolación Lineal Para Una Función Dada. 51
IV.II.II.I Algoritmo.2 52
IV.II.III Doble Interpolación. 52
IV.II.III.I Algoritmo. 52
IV.III Método De Lagrange. 53

IV.III.I Algoritmo. 54
IV.III.II Ejemplo. 55
IV.IV Método Del Spline Cúbico. 55
IV.IV.I Algoritmo. 55
IV.IV.II Ejemplo. 60
IV.V. Mínimos Cuadrados. 61
IV.V.I Regresión Lineal. 61
IV.V.I.I Algoritmo. 61
IV.V.I.II Ejemplo. 62
IV.V.II Ajuste De Curvas No Lineales Por Mínimos Cuadrados. 64
IV.V.II.I Algoritmo. 64
IV.V.II.II Ejemplo. 65

V. DERIVACIÓN E INTEGRACIÓN NUMÉRICA. 69

V.I Introducción. 69
V.II Derivación Numérica. 69
V.II.I Derivadas De Datos. 69
V.II.I.I Algoritmo. 69
V.II.I.II Ejemplo. 70
V.II.III Método Con Serie De Taylor. 71
V.II.III.I Algoritmo. 71
V.II.IV Derivada con Spline Cubico. 73
V.II.IV.I Algoritmo. 73
V.II.IV.II Ejemplo. 74
V.III Integración Numérica. 75
V.III.I Formulas De Newton – Cotes. 75
V.III.I.I Regla Del Trapecio. 75
V.III.I.I.I Algoritmo. 75
V.III.I.I.II Ejemplo. 76
V.III.I.II Regla de Simpson 1/3. 77
V.III.I.II.I Algoritmo. 77
V.III.I.II.II Ejemplo. 78
V.III.I.III Regla de Simpson 3/8. 78
V.III.I.III.I Algoritmo. 78
V.III.II Cuadratura Gaussiana. 78
V.III.II.I Algoritmo. 79
V.III.II.II Ejemplo. 80
V.III.III Integración con Spline Cúbico. 81
V.III.III.I Algoritmo. 81
V.III.III.II Ejemplo. 81

VI. ECUACIONES DIFERENCIALES ORDINARIAS. 83

VI.I Introducción. 83
VI.II Métodos De Euler. 83
VI.II.I Método De Euler Hacia Adelante. 83
VI.II.I.I Algoritmo. 83

VI.II.I.II Ejemplo. 84
VI.II.II Método De Euler Hacia Adelante Para Un Conjunto De EDO. 85
VI.II.II.I Algoritmo. 85
VI.II.II.II Ejemplo. 85
VI.II.III Método De Euler Modificado. 86
VI.II.III.I Algoritmo. 86
VI.II.IV Método De Euler Hacia Atrás. 87
VI.II.IV.I Algoritmo. 87
VI.III Métodos De Runge – Kutta. 87
VI.III.I Método De Runge – Kutta De Segundo Orden. 88
VI.III.I.I Algoritmo. 88
VI.III.I.II Ejemplo. 88
VI.III.II Método De Runge – Kutta De Tercer Orden. 91
VI.III.II.I Algoritmo. 91
VI.III.III Método De Runge – Kutta De Cuarto Orden. 92
VI.III.III.I Algoritmo. 92

VII. ECUACIONES DIFERENCIALES PARCIALES. 93

VII.I Introducción. 93
VII.II Ecuaciones Parabólicas. 94
VII.II.I Ecuaciones de Conducción de Calor. 94
VII.II.II Condiciones Iniciales y de Frontera. 95
VII.III Método Explícito, Implícito y Crank-Nicholson. 96
VII.III.I Método Explícito. 97
VII.III.II Método Implícito. 98
VII.III.III Método Crank – Nicholson. 98
VII.IV Ecuaciones Elípticas. 99
VII.IV.I Ecuaciones Para Flujo Calorífico De Estado Estable. 100

VIII. APLICACIONES DE INGENIERÍA PETROLERA. 103

VIII.I Introducción. 103
VIII.II Equilibrio Líquido – Vapor. 103
VIII.II.I Algoritmo. 103
VIII.III Balance Materia En Un Proceso De Separación Gas – Aceite. 105
VIII.III.I Algoritmo. 106
VIII.III.II Ejemplo. 107
VIII.IV Determinación De Litología Y Porosidad Por Combinación De Los Registros De Porosidad. 112
VIII.IV.I Registros De Porosidad. 112
VIII.IV.II Determinación De Litología Y Porosidad. 113
VIII.V Perfiles De Producción, Reservas De Hidrocarburos, Ganancias Y Número Óptimo De Pozos. 113
VIII.V.I Cálculo Del Perfil De Producción De Crudo De Un Pozo. 114
VIII.V.II Cálculo Del Volumen Mensual De Crudo Producido. 114
VIII.V.III Cálculo Del Volumen De Crudo Entre t=0 y t=n. 115

VIII.V.IV Cálculo Del Perfil De Producción De Gas De Un Pozo, Volumen Mensual De Gas Producido Y Volumen De Gas Entre t=0 y t=n. 115
VIII.V.V Reserva De Hidrocarburos. 115
VIII.V.VI Ganancia Con Un Pozo O Con N Pozos De Crudo. 115
VIII.V.VII Número Óptimo De Pozos De Crudo. 116
VIII.V.VIII Ganancia Con N Pozos De Gas. 116
VIII.V.IX Número Óptimo De Pozos De Gas. 117
VIII.V.X Ejemplo. 117
VIII.VI Pruebas De Incremento. 121
VIII.VI.I Algoritmo. 121
VIII.VI.II Método de Horner. 122
VIII.VI.II.I Ejemplo. 123
VIII.VII Elipse De Esfuerzos Triaxiales. 127
VIII.VII.I Tuberías. 127
VIII.VII.II Fuerzas De Diseño De Las Tuberías. 127
VIII.VII.II.I Esfuerzo De Ruptura. 127
VIII.VII.II.II Esfuerzo De Colapso. 128
VIII.VII.II.III Esfuerzo De Tensión (Axial). 128
VIII.VII.III Esfuerzo Triaxial. 128
VIII.VII.III.I Algoritmo. 129
VIII.VII.III.II Ejemplo. 132
VIII.VIII Ecuación General Para Dos Métodos Explícito, Implícito y Crank – Nicholson y Solución Numérica De La Ecuación De Difusión. 134
VIII.VIII.I Solución De La Ecuación General. 135
VIII.VIII.II Ejemplo. 136
VIII.IX Solución Analítica De La Ecuación De Difusión Para Flujo Radial Y Yacimiento Infinito. 143
VIII.IX.I Desarrollo. 143
VIII.IX.II Algoritmo. 145
VIII.IX.III Ejemplo. 147

CONCLUSIONES Y RECOMENDACIONES. 151

Conclusiones. 151
Recomendaciones. 152

AI. PROGRAMACIÓN EN FORTRAN®. 153

AI.I Elementos Básicos De Programación En FORTRAN®. 153
AI.I.I Inicio De Un Programa En VISUAL FORTRAN®. 153
AI.I.II Variables. 153
AI.I.III Operadores Lógicos, Aritméticos Y De Comparación 154
AI.I.III.I Operadores Lógicos. 154
AI.I.III.II Operadores Aritméticos. 155
AI.I.III.III Operadores De Comparación. 155
AI.I.IV Jerarquía Operacional. 155
AI.II Estructura De Un Programa En FORTRAN®. 155

AI.II.I Sección De Declaración. 155
AI.II.II Sección De Ejecución. 156
AI.II.III Sección De Terminación. 156
AI.II.IV Compilación Y Ejecución De Un Programa. 156
AI.II.V Ejemplo. 157
AI.III Estructuras De Control. 158
AI.III.I Estructuras De Decisión. 158
AI.III.II Estructuras De Repetición O Ciclos. 159
AI.III.III Ejemplos De estructuras De Decisión Y Repetición. 160
AI.IV Arreglos. 162
AI.IV.I Concepto De Vector Y Matriz En Programación. 163
AI.IV.II Manejo De Variables En Forma De Arreglo. 163
AI.IV.II.I Declaración De Arreglos Con Dimensión Fija. 163
AI.IV.II.II Declaración De Arreglos Con Dimensión Variable (Memoria Dinámica). 163
AI.IV.II.III Memoria Dinámica. 163
AI.IV.II.IV Ejemplo. 164
AI.IV.III Lectura E Impresión De Datos. 164
AI.IV.III.I Lectura En Pantalla. 165
AI.IV.III.II Lectura De Archivos Externos. 165
AI.IV.III.III Impresión En Pantalla. 167
AI.IV.III.IV Impresión En Archivos Externos. 168
AI.IV.III.V Ejemplo. 168
AI.IV.IV Operaciones Con Arreglos. 169
AI.IV.IV.I Suma De Matrices En FORTRAN®. 169
AI.IV.IV.II Multiplicación De Matrices En FORTRAN®. 170
AI.IV.IV.III División De Matrices FORTRAN®. 170
AI.V Programación Estructurada. 170
AI.V.I Módulos. 170
AI.V.II Funciones. 171
AI.V.III Subrutinas. 172
AI.V.IV Sentencia Contains. 173
AI.V Recomendacion. 174

AII. PROGRAMACIÓN EN VISUAL BASIC 6.0®. 175

AII.I Elementos Básicos De Programación en VISUAL BASIC 6.0®. 175
AII.I.I Inicio De Un Programa En VISUAL BASIC 6.0®. 175
AII.I.II Objetos. 176
AII.I.II.I Como Insertar Objetos. 176
AII.I.II.II Propiedades. 176
AII.I.III Código. 177
AII.I.IV Operadores Lógicos, Aritméticos Y De Comparación 177
AII.I.IV.I Operadores Lógicos. 177
AI.I.IV.II Operadores Aritméticos. 178
AI.I.IV.III Operadores De Comparación. 178
AII.I.V Ejemplo. 178

AII.I.V.I Resumen De Objetos. 178
AII.I.V.II Código De La Aplicación. 179
AII.I.V.III Explicación. 179
AII.I.VI Ejecución Del Programa. 179
AII.II Variables Y Procedimientos. 180
AII.II.I Variables. 180
AII.II.I.I Tipos. 180
AII.II.I.II Declaración. 180
AII.II.II Procedimientos. 181
AII.II.II.I Procedimientos Para Manipular Cadenas De Caracteres 181
Función Str. 181
Función Val. 181
AII.II.II.II Procedimientos Para Manipular Expresiones Numéricas. 181
Funciones Trigonométricas 181
Funciones Logarítmica Y Exponencial. 182
Función Int. 182
Función Abs. 182
Función Sqr. 182
AII.II.II.III Procedimientos Varios. 182
Función Rnd. 182
AII.III Estructuras De Control. 182
AI.III.I Estructuras De Decisión. 182
AI.III.II Estructuras De Repetición o Ciclos. 184
AII.IV Arreglos. 184
AII.IV.I Manejo De Variables En Forma De Arreglo. 185
AII.IV.I.I Declaración De Arreglos Con Dimensión Fija. 185
AII.IV.II.II Declaración De Arreglos Con Dimensión Variable (Memoria Dinámica). 185
AII.IV.II.III Memoria Dinámica. 185
AII.V Objetos Avanzados. 186
AII.V.I MSFlexGrid. 187
AII.V.I.I Propiedades Del MSFlexGrid. 187
AII.V.I.II Escritura En MSFlexGrid. 188
AII.V.I.II.I Escritura Mediante Teclado. 188
AII.V.I.II.II Lectura De Datos De Un MSFlexGrid. 189
AII.V.I.II.III Renglones Y Columnas De Un MSFlexGrid. 189
AII.V.I.III Uso De Arreglos Con MSFlexGrid. 189
AII.V.I.IV Ejemplo. 191
AII.V.I.IV.I Resumen De Objetos. 192
AII.V.I.IV.II Código De La Aplicación. 192
AII.V.II PictureBox. 193
AII.V.II.I Propiedades Del PictureBox. 193
AII.V.II.II Métodos Gráficos. 193
AII.V.II.III Graficas En PictureBox. 195
AII.V.II.IV Ejemplo. 196
AII.V.III OptionButton. 196

AII.V.III.I OptionButton Con IF…THEN…ELSE. 196
AII.V.III.I OptionButton Con el Evento Click. 197
AII.V.IV CheckBox. 197
AII.V.VI.I CheckBox Con IF…THEN…ELSE. 197
AII.VI Recomendación. 198

AIII. MATLAB 7.0®. 199

AIII.I Introducción. 199
AIII.II Espacio De Trabajo. 199
AIII.III Asignación De Valores. 200
AIII.IV Valores Y Matrices Especiales. 202
AIII.IV.I Valores Especiales. 202
AIII.IV.II Matrices Especiales. 202
AIII.IV.II.I Elementos Continuos. 202
AIII.IV.II.II Matriz Transpuesta. 202
AIII.IV.II.III Matriz Identidad. 203
AIII.IV.II.IV Matriz Inversa. 203
AIII.IV.II.V Determinante. 203
AIII.V Operaciones Con Escalares Y Arreglos. 204
AIII.V.I Operaciones Con Escalares. 204
AIII.V.II Operaciones Con Arreglos. 204
AIII.V.III Operadores Lógicos. 205
AIII.V.IV Operadores De Comparación. 205
AIII.VI Gráficos Con MATLAB 7.0®. 206
AIII.VI.I Gráficos 2D. 206
AIII.VI.I.I Función plot(). 206
AIII.VI.I.II Función title. 206
AIII.VI.I.III Función xlabel e ylabel. 206
AIII.VI.I.IV Función Grid. 206
AIII.VI.I.V Ejemplo. 206
AIII.VI.II Gráficos 3D. 207
AIII.VI.II.I Función 3plot. 207
AIII.VI.II.I.I Ejemplo. 207
AIII.VI.II.II Funciones meshgrid y mesh. 207
AIII.VI.II.II.I Ejemplo. 207
AIII.VII Programación Con MATLAB 7.0®. 208
AIII.VII.I Archivos *.m. 208
AIII.VII.II Lectura Y Escritura. 208
AIII.VII.II.I Función input. 208
AIII.VII.II.II Función disp. 208
AIII.VII.III Estructuras De Control. 208
AIII.VII.III.I Estructuras De Decisión. 209
AIII.VII.III.I Estructuras De Repetición O Ciclos. 209
AIII.VII.IV Lectura De Datos Por Medio De Archivos Externos. 210
AIII.VII.V Ejemplo 1. 210

AIII.VII.VI Ejemplo 2 211

AIV. POLINOMIOS DE LEGENDRE. 213

AIV.I Algoritmo. 213

AV. MALLAS CARTESIANAS Y RADIALES. 215

AV.I Introducción. 215

AV.II Malla Cartesiana. 218

AV.II.I Nodos Distribuidos. 218

AV.II.I.I Algoritmo. 218

AV.II.II Nodos Centrados. 219

AV.II.II.I Algoritmo. 219

AV.III Malla Radial. 219

AV.III.I Algoritmo. 220

Método 1. 220

Método 2. 220

AVI. ECUACIÓN DE DIFUSIÓN PARA FLUJO DE FLUIDOS EN MEDIOS POROSOS. 221

AVI.I Introducción. 221

AVI.II Ecuación General. 221

AVII. ECUACIÓN DE DIFUSIÓN FLUJO RADIAL. 223

AVII.I Introducción. 223

AVII.II Ecuación General. 223

AVIII. MÉTODO DE THOMAS. 227

AVIII.I Introducción. 227

AVIII.II Algoritmo. 228

AIX. INTEGRAL EXPONENCIAL. 229

AVIII.I Algoritmo. 229

AX. APLICACIÓN. 231

AX.I Introducción. 231

AX.II Presentación. 231

AX.III Archivo. 231

AX.III.I Abrir. 232

AX.III.II Reporte. 232

AX.III.III Ejemplo. 232

AX.III.IV Salir. 232

AX.IV Acceso a las Aplicaciones. 232

AX.V Recomendaciones. 232

REFERENCIAS. 233

LISTA DE FIGURAS.

Página

I.1 Etapas de Programación. 7
I.2 Diagrama de Flujo: Terminal. 11
I.3 Diagrama de Flujo: Proceso. 12
I.4 Diagrama de Flujo: Entrada/Salida. 12
I.5 Diagrama de Flujo: Decisión. 12
I.6 Diagrama de Flujo: Conector. 12
I.7 Diagrama de Flujo: Flujo de Datos. 12
I.8 Diagrama de Flujo: Procedimiento. 13
I.9 Diagrama de Flujo: Impresión. 13
I.10 Diagrama de Flujo: Dirección. 13
I.11 Diagrama de Flujo: Líneas. 13

III.1 Método Newton - Raphson. 43
III.2 Método de la Secante. 46

IV.1 Recta de Interpolación a Partir de Dos Puntos. 50
IV.2 Recta de Interpolación a Partir de Una Función. 52
IV.3 Doble Interpolación. 53
IV.4 Función Ajustada a un Número n de Datos. 53
IV.5 Gráfica Ejemplo de Regresión Lineal. 63
IV.6 Recta de Ajuste Regresión Lineal. 64
IV.7 Gráfica Ejemplo Mínimos Cuadrados. 66
IV.8 Polinomio de Ajuste Mínimos Cuadrados. 67

V.1 Aproximación Progresiva. 72
V.2 Aproximación Regresiva. 72
V.3 Aproximación Central. 73
V.4 Regla del Trapecio. 75
V.5 Ejemplo Regla del Trapecio. 76

VI.1 Ejemplo Runge – Kutta Segundo Orden. 89
VII.1 Varilla Conducción de Calor. 94
VII.2 Método Explícito. 97
VII.3 Método Implícito. 98
VII.4 Método Crank – Nicholson. 99
VII.5 Ecuación Flujo Calorífico de Estado Estable. 100

VIII.1 Batería de Separación. 105
VIII.2 Perfil de Producción de Aceite. 119
VIII.3 Perfil de Producción de Gas. 120

VIII.4 Gráfica de Horner. 122
VIII.5 Ejemplo Gráfica de Horner Semilog. 124
VIII.6 Ejemplo Gráfica de Horner Normal. 124
VIII.7 Ejemplo Grafica de Horner Recta Ajustada. 126
VIII.8 Esfuerzo Triaxial. 129
VIII.9 Elipse de Esfuerzos Triaxiales. 131
VIII.10 Ejemplo Elipse de Esfuerzos Triaxiales. 134
VIII.11 Gráfica Presión vs Longitud. 142
VIII.12 Gráfica Semilog Presión vs Longitud. 149

AI.1 Programa Nuevo. 153
AI.2 Ventana Output. 157
AI.3 Programa en Ejecución. 158
AI.4 Archivos de Datos para Vector. 166
AI.5 Archivos de Datos para Matriz. 166
AI.6 Insertar Módulo. 170

AII.1 Nuevo Proyecto. 175
AII.2 Espacio de Trabajo. 175
AII.3 Formulario. 176
AII.4 Barra de Objetos. 176
AII.5 Pantalla de Código. 177
AII.6 Ejemplo en Ejecución. 179
AII.7 MSFlexGrid. 187
AII.8 Componentes. 187
AII.9 Renglón y Columna Fija. 189
AII.10 Renglón y Columna Libres. 189
AII.11 Sistema Cartesiano PictureBox. 195
AII.12 Ejemplo en Ejecución. 196
AII.13 OptionButton. 196
AII.14 CheckBox. 197

AIII.1 Espacio de Trabajo. 199
AIII.2 Botón Start. 200
AIII.3 Ejemplo Gráfica 2D. 206
AIII.4 Ejemplo Gráfica 3D. 207
AIII.5 Ejemplo Superficie 3D. 207

AV.1 Simulador Unidimensional. 215
AV.2 Simulador Unidimensional Radial. 216
AV.3 Simulador Areal. 216
AV.4 Simulador Transversal. 216
AV.5 Simulador Tridimensional. 217
AV.6 Nodos Distribuidos. 217
AV.7 Nodos Centrados. 218
AV.8 Malla Radial. 219
AX.1 Presentación. 231
AX.2 Archivo. 231
AX.3 Menú Principal. 232

LISTA DE TABLAS.

Página

I.1 # de Bits. 15
I.2 Distribución de Bits para Precisión de 32 Bits. 17
I.3 Distribución de Bits para Precisión de 64 Bits. 18

II.1 Operaciones Elementales del Método de Cramer. 23
II.2 Métodos Directos. 25
II.3 Métodos Indirectos. 25

III.1 Ejemplo Newton – Raspón $x_0 = 1$. 44
III.2 Ejemplo Newton – Raspón $x_0 = 1.4$. 45
III.3 Ejemplo Newton – Raspón $x_0 = -2.2$. 45

IV.1 Tabla de Valores. 49
IV.2 Tabla de Valores con Uno Desconocido. 50
IV.3 Ejemplo Interpolación Lineal. 51
IV.4 Ejemplo Método Lagrange. 55
IV.5 Ejemplo Spline Cúbico. 60
IV.6 Ejemplo Regresión Lineal. 62
IV.7 Valores A y Z de Regresión Lineal. 63
IV.8 Ejemplo Mínimos Cuadrados. 65
IV.9 Elementos Matriz y Vector para Mínimos Cuadrados. 66
IV.10 Coeficientes de Potencias. 67

V.1 Ejemplo Derivada con de Datos. 70
V.2 Ejemplo Derivada con Spline Cúbico. 74
V.3 Resultados Derivada con Spline Cúbico. 74
V.4 Derivadas y Error. 75
V.5 Puntos de Gauss. 79

VI.1 Ejemplo Euler Hacia Adelante. 85
VI.2 Resultados Runge – Kutta Segundo Orden $1[seg]$. 90
VI.3 Resultados Runge – Kutta Segundo Orden $1[seg] \le t \le 10[seg]$. 90

VIII.1 Ejemplo Balance de Materia. 108
VIII.2 1ª Etapa de Separación. 109
VIII.3 2ª Etapa de Separación. 109

VIII.4 3ª Etapa de Separación. 110
VIII.5 Fracción Liquido - Vapor. 110
VIII.6 Registros de Porosidad. 112
VIII.7 Ejemplo Perfiles de Producción. 117
VIII.8 Perfil de Producción. 118
VIII.10 Resultados Perfil de Producción. 120
VIII.11 Ejemplo Método de Horner. 123
VIII.12 Ejemplo Método de Horner Regresión Lineal. 125
VIII.13 Ejemplo Elipse de Esfuerzos Triaxiales. 132
VIII.14 Resultados Elipse de Esfuerzos Triaxiales. 133
VIII.15 Ejemplo Ecuación General. 136
VIII.16 Ejemplo Ecuación General Sistema Internacional. 137
VIII.17 Posición de Nodos. 137
VIII.18 Coeficientes y Presión $t_{acum} = 0.1[día]$. 139
VIII.19 Coeficientes y Presión $t_{acum} = 10.1[día]$. 140
VIII.20 Coeficientes y Presión $t_{acum} = 1,010.1[día]$. 140
VIII.21 Coeficientes y Presión $t_{acum} = 101,010.1[día]$. 141
VIII.22 Coeficientes y Presión $t_{acum} = 1,101,010.1[día]$. 141
VIII.23 Ejemplo Solución Analítica. 148
VIII.24 Resultados Ejemplo Solución Analítica. 148

AI.1 Operadores Lógicos. 154
AI.2 Operadores Aritméticos. 155
AI.3 Operadores Comparación. 155

AII.1 Operadores Lógicos. 177
AII.2 Operadores Aritméticos. 178
AII.3 Operadores Comparación. 178
AII.4 Objetos para Ejemplo. 178
AII.5 Tipos de Variables. 180
AII.6 Claves de Objeto. 186
AII.7 Objetos para Ejemplo MSFlexGrid. 192

AIII.1 Valores Especiales. 202
AIII.2 Operaciones Escalares. 203
AIII.3 Operaciones con Arreglos. 203
AIII.4 Operadores Lógicos. 205
AIII.5 Operadores Comparación. 205
AIII.6 Funciones Gráficas 2D. 206

OBJETIVO Y RESUMEN.

Objetivo.

Proporcionar al alumno una herramienta de aprendizaje y consulta para lograr el máximo aprovechamiento de su curso de Programación Avanzada considerado en el sexto semestre de la carrera de Ingeniería Petrolera de la Facultad de Ingeniería de la UNAM. Así mismo aportar un asistente para el profesor encargado de impartir dicha clase.

Resumen.

El presente trabajo se realizó buscando ocupar el lugar de apuntes para la asignatura de Programación Avanzada del plan de estudios de Ingeniería Petrolera de la UNAM aprobado por el Consejo Técnico de la Facultad de Ingeniería y por el Consejo Académico del Area de las Ciencias Físico Matemáticas y de las Ingenierías en el año 2005, sin embargo no buscan sustituir, mas bien complementar, los apuntes elaborados por Roberto Ariel Guzmán Guzmán bajo el titulo *"Programación Avanzada Aplicada a Ingeniería Petrolera"* en el año 2001 como tema de tesis, y dirigida por el Dr. Néstor Martínez.

La columna que sostiene este esfuerzo es el programa de estudio de la materia compuesto por seis temas y que son tratados ampliamente en este trabajo, sin embargo, dicho temario se amplió con dos capítulos (I y VIII), obteniendo el siguiente contenido:

I. Conceptos Básicos.
II. Solución de Sistemas de Ecuaciones Lineales.
III. Solución de Sistemas de Ecuaciones No Lineales.
IV. Interpolación Numérica.
V. Derivación e Integración Numérica.
VI. Ecuaciones Diferenciales Ordinarias.
VII. Ecuaciones Diferenciales Parciales.
VIII. Aplicaciones de Ingeniería Petrolera.

En el capitulo I se abordan los conceptos que cualquier programador debe conocer, del capitulo II al VII se abordan los temas que componen el plan de estudios de la asignatura, estos son métodos numéricos, y se presentan los

algoritmos que permiten su aplicación, en el capitulo VIII se proponen y detallan algunas aplicaciones de las tres ramas de la Ingeniería Petrolera (Perforación, Yacimientos y Producción), que el alumno conoce a lo largo de la carrera, apropiadas para elaborar programas capaces de resolver los problemas que ellas presentan, y que pueden o no requerir de una solución numérica de las presentadas en los capítulos II al VII.

Por último se incluye una breve conclusión y algunas recomendaciones a los alumnos y al profesor de la asignatura.

En la sección de anexos se incluyen dos pequeños manuales dirigidos a las personas con nulos o pocos conocimientos de programación y que orientan al lector en sus primeros pasos como programador, estos manuales son sobre el uso de VISUAL FORTRAN® y MATLAB 7.0 ® contemplados en el temario de la asignatura, adicionalmente y buscando aportar un mayor conocimiento al alumno se incluye otro de VISUAL BASIC 6.0 ®, el resto de los anexos están dirigidos a complementar los temas tratados a lo largo de los ocho capítulos.

INTRODUCCION.

Como Ingeniero Petrolero no es suficiente ser usuario del software existente en el mercado, software que va desde el OFFICE® hasta las potentes herramientas que se utilizan en la industria como:

- Simuladores de yacimientos.
- Diseño de instalaciones superficiales.
- Análisis PVT de las muestras de hidrocarburos.
- Interpretación Sísmica.
- Interpretación de Registros Geofísicos.

Es necesario conocer por lo menos las técnicas básicas de programación que permitan entender el funcionamiento de dicho software, esto permitirá detectar posibles fuentes de error al momento de utilizarlos e inclusive poder ser capaces de realizar observaciones al proveedor de dichas herramientas.

Para esto y antes de comenzar a programar es necesario conocer las siguientes definiciones básicas para cualquier programador.

Definiciones Básicas.

Computadora.

Se trata de cualquier máquina que automatiza cálculos. Esta definición limita el desarrollo actual de las computadoras, pues también son de gran utilidad para realizar actividades repetitivas y monótonas donde se trabajan datos que han sido introducidos previamente, se procesan y se devuelven otra serie de datos, además se utilizan como medios de entretenimiento pues son capaces de reproducir video, música, e incluso juegos de video.

Para definir una computadora se debe tener en cuenta las dos partes que la componen:

1. ***Software:*** Se refiere a los programas que se instalan en la computadora y que permiten realizar las diferentes actividades con la computadora.

2. ***Hardware:*** Se refiere a la parte física de la computadora como son el monitor, teclado CPU, ratón, impresora, circuitos cables, este resulta completamente inútil sin la existencia del software y viceversa.

En resumen, una computadora es un conjunto de componentes físicos (hardware) que el usuario puede manipular por medio de otro conjunto de componentes virtuales (software) con la finalidad de realizar desde cálculos sencillos hasta el procesamiento de grandes cantidades de información en trabajos largos, complejos y repetitivos en un periodo de tiempo muy pequeño.

Adicionalmente se pueden utilizar para acceder a la información disponible en Internet y para el entretenimiento.

Programa.

Es un conjunto de instrucciones que permiten a la computadora realizar alguna acción específica, estas acciones abarcan cálculos simples hasta procesos iterativos con gran cantidad de datos, además de funciones de entretenimiento y navegación por Internet.

Lenguaje de Programación.

Es el que permite a los desarrolladores de software, expresar las soluciones que han desarrollado en términos de un lenguaje que después a través de un compilador será interpretado por la computadora.

Compilador.

Es un sistema que convierte el código escrito en algún lenguaje de programación en una aplicación que la computadora pueda ejecutar.

Algoritmo.

Es una secuencia de pasos lógicos necesarios para llevar a cabo una tarea específica, como la solución de un problema. El algoritmo es la idea lógica que está atrás del problema.

Diagrama de Flujo.

Es la representación visual o gráfica de un algoritmo, el diagrama de flujo emplea una serie de bloque y de flechas cada uno de los cuales representa una operación en particular o un paso en el algoritmo. Las flechas representan la secuencia en la cual se realizan las operaciones. Mas adelante se tocaran algunos elementos importantes para la realización de diagramas de flujo.

I. CONCEPTOS BÁSICOS.

I.I Características de un programa.

I.I.I Debe funcionar.

La característica más simple e importante de un programa es que funcione.

I.I.II No debe tener dificultades.

Hay que anticipar las situaciones en las que se va a emplear el programa con el fin de evitar errores, esto puede ocurrir cuando el usuario introduce información errónea al programa, para esto el programador deberá asegurarse de obligar al usuario a utilizar solo información correcta y coherente con las capacidades y funciones del programa.

I.I.III Debe ser bien documentado.

La documentación es necesaria para ayudar a comprender o a utilizar un programa. Puede realizarse de dos formas:

1. Externa: Lo que comúnmente se conoce como “Manual de referencia”, permite aprender a utilizar el programa.

2. Interna: La que se incluye en muchas aplicaciones como “Ayuda”, permite resolver dudas sobre le funcionamiento del programa.

Adicionalmente para trabajos de mantenimiento y actualización es conveniente tener documentada cada parte de la estructura del programa, con el fin de que estos trabajos no sean complicados de realizar.

I.I.IV El programa debe ser eficiente.

Que un programa sea fácil de leer y de comprender es importante para su mantenimiento y modificación.

Para lograr cada uno de estos puntos se recomienda seguir las siguientes etapas durante la programación.

I.II Etapas de programación.

I.II.I Identificación del problema y definición del objetivo.

En primer lugar se tiene que establecer si un problema realmente existe y se tiene que determinar que es lo que se pretende evaluar a través de la solución de dicho problema.

I.II.II Análisis del problema.

Una vez establecida la existencia y la necesidad de solución del problema es necesario conocer a fondo el problema que se pretende resolver antes de proceder a desarrollar la solución a este.

I.II.III Descripción matemática.

Es necesario el desarrollo de una descripción matemática pues esta es la única manera en que la computadora será capaz de resolver el problema, esta deberá incluir todas las justificaciones teóricas y las manipulaciones matemáticas que permitan una mayor simplificación del modelo y así reducir la complejidad de este.

I.II.IV Desarrollo de solución.

Se deben conocer y entender los diferentes métodos que permiten dar solución al problema y así poder elegir adecuadamente el que mejor funciona o poder incluir todas las diferentes formas de solución en la aplicación que se pretende desarrollar. Aquí el análisis numérico juega un papel importante pues dependiendo de la descripción matemática se podrá elegir mejor la herramienta numérica que permita a la computadora resolver el modelo.

I.II.V Algoritmo.

El desarrollo del algoritmo computacional involucra la transformación del modelo matemático en un método numérico de tal forma que la computadora pueda interpretarlo. Se necesita el desarrollo de una organización lógica para su aplicación en la computadora. Para esto se puede hacer uso de los diagramas de flujo que pueden ayudar en dicha organización.

I.II.VI Prueba de escritorio.

Consiste en realizar los pasos de la solución del problema de manera escrita o manual con la finalidad de verificar que la solución propuesta devolverá los resultados deseados.

I.II.VII Construcción de solución en forma de programa.

Esta etapa es mecánica, pues consiste en la construcción en forma de un programa, la solución desarrollada en la etapa anterior siguiendo la lógica de programación respetando las reglas del lenguaje se vaya a utilizar.

I.II.VIII Prueba e interpretación.

Es responsabilidad de los programadores realizar las pruebas necesarias que garanticen el correcto funcionamiento del programa. Estas se deben hacer con los casos más representativos y que simulen cada una de las posibilidades de solución del problema que se busca resolver a fin de comparar los resultados producidos por la computadora mediante el programa realizado con otros obtenidos por medio de alguna otra herramienta.

I.II.IX Documentación.

Permite que el usuario final pueda manipular de manera correcta el programa. Además puede ayudar en el mantenimiento del mismo. Es lo que comúnmente se conoce como manual.

Estas etapas se resumen en el siguiente diagrama de flujo:

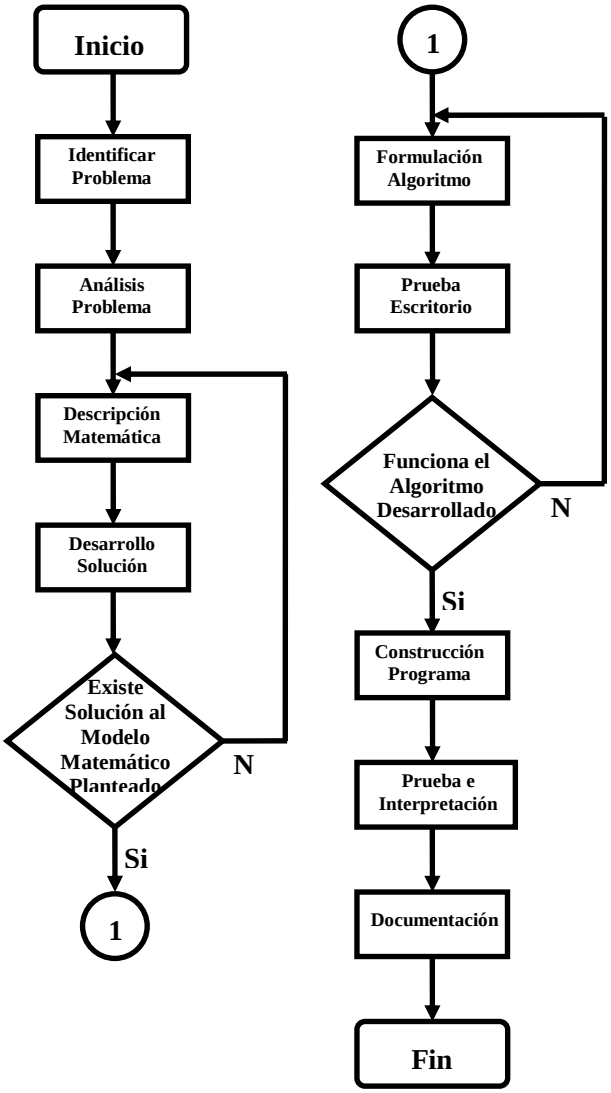


Figura I.1 Etapas de Programación

I.III Tipos de programación.

En la actualidad existen muchas opciones para desarrollar un programa, la gran variedad de lenguajes de programación ofrecen diversas ventajas y al mismo tiempo desventajas para el programador.

Además es importante reconocer la existencia de dos corrientes en la programación, primero la programación estructurada y después la programación orientada a objetos.

La primera es la respuesta al tipo de programación que se realizaba al inicio de la utilización de las computadoras, esta se conocía como programación libre y hacia uso de algo conocido como ***salto incondicional,*** se trata de una instrucción de programación con la cual, el contador del programa toma un valor nuevo, que el programador indica. Este método se empleó en las primeras técnicas de programación. Por ejemplo algunas líneas de FORTRAN® en programación libre serian:

```
PROGRAM PRIMERO
IMPLICIT NONE
        REAL(8)::A,B,C
        WRITE(*,*)'INDICA A'
        READ(*,*)A
        100 WRITE(*,*)'INDICA B'
        READ(*,*)B
        C=A/B
        WRITE(*,*)C
        IF C=2.5 THEN
        GOTO 100
        END IF
END PROGRAM
```

Cuando la ejecución del programa llega a la instrucción ***GOTO***, la siguiente sentencia ejecutada será la que se encuentre en: ***100***. El abuso de esta, aparentemente ágil sentencia, da lugar a problemas graves en programas robustos y complejos. El salto incondicional no es necesario en un lenguaje de programación de alto nivel, ya en los años 60 se dieron cuenta de ello, desarrollándose técnicas de programación que no lo utilizaban, sin embargo los lenguajes de programación aun la tienen como recurso de programación, pero las técnicas de programación así como los expertos recomiendan e incluso prohíben su uso.

En la industria petrolera se utiliza de manera muy importante el lenguaje FORTRAN® apoyándose en algún otro software como MATLAB® para la parte gráfica que permite interpretar los resultados obtenidos con los programas realizados en FORTRAN®.

En este trabajo se abordarán los elementos básicos para utilizar los lenguajes de programación FORTRAN® y MATLAB® contemplados en el programa de la asignatura Programación Avanzada del sexto semestre de la carrera de Ingeniaría Petrolera de la Facultad de Ingeniería de la UNAM, además se incluirá una sección de Visual Basic 6.0® con la finalidad de contribuir a un mayor desarrollo del lector.

I.III.I Programación estructurada.

La programación estructurada es una técnica en la cual la estructura de un programa se realiza tan claramente como sea posible mediante el uso de tres estructuras lógicas de control:

1. Secuencia: Sucesión simple de dos o mas operaciones.
2. Selección: Es una condicional de una o mas operaciones.
3. Iteración: Repetición de una operación mientras se cumple una condición.

Esto hace innecesario el uso de la instrucción o instrucciones de transferencia incondicional (GOTO).

Los tres tipos de estructuras lógicas de control pueden ser combinados para producir programas que manejen cualquier tarea de procesamiento de información.

Un programa estructurado esta compuesto de segmentos, los cuales pueden estar constituidos por unas pocas instrucciones o por una página o más de codificación.

Una característica importante de un programa estructurado es que puede ser leído en secuencia, desde el comienzo hasta el final sin perder la continuidad de la tarea que cumple el programa, lo contrario de lo que ocurre con otros estilos de programación. Esto es importante debido a que, es mucho más fácil comprender completamente el trabajo que realiza una función determinada, si todas las instrucciones que influyen en su acción están físicamente contiguas y encerradas por un bloque.

La facilidad de lectura, de comienzo a fin, es una consecuencia de utilizar solamente tres estructuras de control y de eliminar la instrucción de desvío de flujo de control, excepto en circunstancias muy especiales tales como la simulación de una estructura lógica de control en un lenguaje de programación que no la posea.

I.III.I.I Partes de un programa estructurado.

El código de un programa elaborado de forma estructurada cuenta con cuatro secciones principales:

1. Declaración de Variables: En esta sección se declara todas las variables de entrada y salida que se utilizaran a lo largo de todo el programa.
2. Ejecución de las Instrucciones: En esta sección se indica el manejo que se dará a las variables declaradas mediante las operaciones indicadas.
3. Formato de los datos de entrada y salida.
4. Terminación: En esta sección se indica el término del programa.

Fortran es un lenguaje de programación desarrollado en los años 50 y activamente utilizado desde entonces. Y su nombre lo toma del acrónimo de "Formula Translation".

Fortran se utiliza principalmente en aplicaciones científicas y análisis numérico. Desde 1958 ha pasado por varias versiones, entre las que destacan FORTRAN® II, FORTRAN® IV, FORTRAN® 77, FORTRAN® 90, FORTRAN® 95 y FORTRAN® 2003, VISUAL FORTRAN®. Para este trabajo se revisara el paquete VISUAL FORTRAN® 6.0, que permite compilar códigos fuente escritos en FORTRAN® 90, es decir la programación es de tipo estructurada, en la sección de anexos se encuentra un manual básico para poder comenzar a programar en FORTRAN® 90 haciendo uso del VISUAL FORTRAN® 6.0.

I.III.II Programación orientada a objetos.

La programación orientada a objetos (POO) es una forma de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. Esta corriente surge a partir de la necesidad de volver la interacción del usuario con la computadora más amigable, además del surgimiento de sistemas operativos de tipo gráfico como lo son Windows ® y Mac OS ®.

I.III.II.I Mecanismos básicos de la POO.

Los mecanismos básicos de la programación orientada a objetos son:

- Objetos. Un programa tradicional se compone de procedimientos y datos. Un programa orientado a objetos se compone solamente de objetos. Un objeto es una entidad que tiene unos atributos particulares, las propiedades, y unas formas de operar sobre ellos, los métodos. Por ejemplo, una ventana del sistema operativo Windows® es un objeto. El color de fondo de la ventana, el alto, etc., son propiedades. Las rutinas, lógicamente transparentes al usuario, que permiten maximizar la ventana, minimizarla, etc., son métodos.
- Mensajes. Cuando se ejecuta un programa orientado a objetos, los objetos están recibiendo, interpretando y respondiendo a mensajes de otros objetos.
- Métodos. Un mensaje esta asociado con un procedimiento, de tal forma que cuando un objeto recibe un mensaje la respuesta a ese mensaje es ejecutar el procedimiento asociado. Este procedimiento recibe el nombre de método.

En adición, las propiedades permitirán almacenar información para dicho objeto. Un método puede también enviar mensajes a otros objetos solicitando una acción o información.

La estructura más interna de un objeto esta oculta para otros usuarios y la única conexión que tiene con el exterior son los mensajes. Los datos que están dentro de un objeto solamente pueden ser manipulados por los métodos asociados al propio objeto.

I.III.II.II Propiedades de los objetos.

Cada clase de objeto tiene predefinido un conjunto de propiedades, como título, nombre, color, etc. Las propiedades de un objeto representan todos los datos que por definición están asociadas con ese objeto.

I.IV Diagramas de flujo.

Como se mencionó anteriormente (Introducción) los diagramas de flujo son de ayuda en la organización de la lógica plasmada en un algoritmo de solución e incluso en el desarrollo de la solución de un problema.

Estos son una representación gráfica del flujo lógico, instrucciones y de datos que se utilizarán en una parte específica del programa o en algunos casos de todo. Esto da a entender que los diagramas se dibujan antes de escribir el programa con la finalidad de asegurar el desarrollo lógico de este.

Algunos expertos los recomiendan y otros no los consideran necesarios, en este trabajo solo algunos de los programas realizados se ejemplificaran por medio de diagramas de flujo.

I.IV.I Razones para el uso de diagramas de flujo.

1. Se usan para expresar y comunicar un algoritmo.
2. Son utilizados en la planeación y comunicación lógica para alguien externo a los programas.
3. Son una herramienta excelente en la redacción o pedagogía, y sirven como un proceso lógico de disciplina.

I.IV.II Símbolos utilizados en la construcción de un diagrama de flujo.

Los símbolos más utilizados son los siguientes:

1. ***Terminal.*** Indica el comienzo o el punto final del diagrama.

Figura I.2 Terminal.

2. ***Proceso.*** Proceso interno de la computadora, o sea, cualquier transferencia de datos u operaciones aritméticas.

Figura I.3 Proceso.

3. ***Entrada/Salida.*** Operación de entrada o salida, como lectura o escritura.

Figura I.4 Entrada/Salida.

4. ***Decisión.*** Verificar si el resultado de una expresión es cierto o falso.

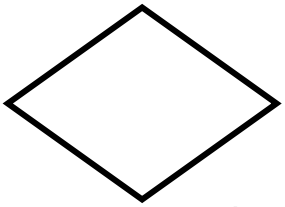


Figura I.5 Decisión.

5. ***Conector.*** Punto de referencia que indica dónde debe continuar el diagrama de flujo. Se utiliza para indicar un cambio en el flujo normal de datos (transferencia o bifurcación).

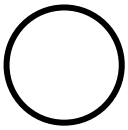


Figura I.6 Conector.

6. ***Sentido del Flujo de Datos.*** Indica cual es la siguiente operación a realizar a partir del símbolo actual.

Figura I.7 Flujo de Datos.

7. *Procedimiento.* Este símbolo sustituye a todo un subprograma cuyo desarrollo se pospone para el final (subrutinas).

Figura I.8 Procedimiento.

8. *Impresión.* Indica cuando se tiene una serie de datos que se van a imprimir (resultados).

Figura I.9 Impresión

I.IV.III Reglas para la construcción de diagramas de flujo.

1. Todo diagrama debe tener un inicio y un fin.
2. Las líneas utilizadas para indicar la dirección del flujo del diagrama deben ser rectas verticales y horizontales.
3. Todas las líneas utilizadas para indicar dirección deben estar conectadas.

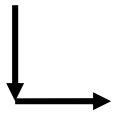


Figura I.10 Dirección.

4. El diagrama debe construirse de arriba hacia abajo y de izquierda a derecha.
5. No puede llegar más de una línea o un símbolo a un bloque.

Correcto ***Incorrecto***

Figura I.11 Líneas.

6. Dentro de cada símbolo se escribe un comentario para indicar la operación o proceso especifico que se ha de ejecutar.
7. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación.
8. La solución presentada en el diagrama puede escribirse posteriormente en cualquier lenguaje de programación.

I.IV.IV Etapas en la construcción de un diagrama de flujo.

Generalmente se trata de solo tres etapas que conforman un diagrama de flujo:

- Etapa 1. Lectura de datos.
- Etapa 2. Proceso de datos.
- Etapa 3. Impresión de resultados.

Sin embargo es importante aclarar que estas etapas se pueden repetir varias veces durante el flujo del diagrama dependiendo del problema que se esté tratando.

I.V Aritmética de las computadoras.

Las computadoras no almacenan los números con precisión infinita sino de forma aproximada empleando un número fijo de *bits* o *bytes.* Prácticamente todas las computadoras permiten al programador elegir entre varias representaciones o "tipos de datos". Los diferentes tipos de datos pueden diferir en el número de bits empleados, pero también en cómo el número representado es almacenado en formato de número entero (integer) o como número real (punto flotante).

De lo anterior es importante mencionar que un Bit es la mínima expresión que puede ser representada por una computadora mientras que un Byte es un conjunto de 8 Bits.

I.V.I Sistemas de números.

I.V.I.I Sistema decimal.

En el sistema decimal se utilizan los diez dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) para representar números. La posición de cada dígito es la que determina el valor del número que se quiere representar.

I.V.I.II Sistema binario.

Este es el sistema que utilizan las computadoras para representar los números, y como el nombre lo indica es un sistema con base 2 y los únicos números que utiliza son los dígitos uno y cero (1 y 0).

I.V.I.III Sistema hexadecimal.

Este sistema es utilizado por algunos tipos de computadoras, la base de este es 16 y utiliza los diez dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) además de las letras A, B, C, D, E y F, que toman los siguientes valores:

A=10	D=13
B=11	E=14
C=12	F=15

I.V.I.IV Ejemplos.

Por ejemplo para un entero:

$$5692=5*10^3+6*10^2+9*10^1+2*10^0$$

Por ejemplo para un real:

$$6921.368=6*10^3+9*10^2+2*10^1+1*10^0+3*10^{-1}+6*10^{-2}+8*10^{-1}$$

Se dice que la base del sistema decimal es 10.
Un número binario a base 10.

$$[10100101]_2=1*2^7+0*2^6+1*2^5+0*2^4+0*2^3+1*2^2+0*2^1+1*2^0=128+32+4+1=[165]_{10}$$

Los números enteros pueden ser representados exactamente por el sistema binario.

Un número hexadecimal a base 10.

$$[39]_{16}=3*16^1+9*16^0=57_{10}$$

$$[EA9D]_{16}=14*16^3+10*16^2+9*16^1+13*16^0=57344+2560+144+13=[60061]_{10}$$

I.V.II Números de punto fijo (Enteros o INTEGER).

Las computadoras almacenan los números con un número finito de bits, por lo tanto, el número de bits limita el valor del número que se puede representar.

3 bits como máximo	$(111)_2$	$=4+2+1=7$	$=2^3-1$
4 bits como máximo	$(1111)_2$	$=8+4+2+1=15$	$=2^4-1$
5 bits como máximo	$(11111)_2$	$=16+8+4+2+1=31$	$=2^5-1$
N bits como máximo			$=2^n-1$

Tabla I.1 # de Bits.

Por ejemplo, una computadora que trabaje con 16 bits, los enteros están comprendidos entre:

$$-(2^{16} - 1) \text{ y } 2^{16} - 1$$

Se tendrá un intervalo de:

[-32768,32767]

Un número representado en formato entero es “exacto” por lo tanto las operaciones aritméticas entre números enteros son también “exactas” siempre y cuando:

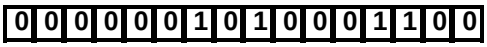
1. La solución no esté fuera del rango del número entero más grande o más pequeño que se puede representar (generalmente con signo). En estos casos se dice que se comete un error de desbordamiento por exceso o por defecto (en inglés: *Overflow* y *Underflow*).
2. La división da lugar a un número entero, despreciando la parte después del punto decimal.

Por estos motivos, la aritmética de punto fijo se emplea muy raramente en cálculos de alto nivel.

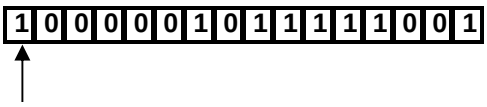
Los números enteros se almacenan como un número binario por medio de una cadena de bits unidos. Si se trabaja con 16 bits se contaría con quince posiciones para colocar el número y una para el signo (sign bit), el signo se representa con 0 para el positivo (+) y 1 para el negativo (-).

Por Ejemplo:

$[652]_{10}=[1010001100]_2$



$[-761]_{10}=[1000001011111001]_2$



De igual forma se trabajaría con 32 bits, se tendrían treinta y una posiciones para el número y una para el signo, si se tratara de 64 bits tendría sesenta y tres posiciones para el número y una para el signo.

I.V.III Números de punto flotante (Reales).

I.V.III.I Notación científica normalizada.

En el sistema decimal, cualquier número real puede expresarse mediante notación científica normalizada. Para expresar un número en notación científica normalizada se multiplica o divide por 10 tantas veces como sea necesario para que todos los dígitos aparezcan a la derecha del punto decimal de modo que el primer dígito después del punto no sea cero.

Por ejemplo:

$$563.2365=0.5632365*10^{3}$$
$$-0.006954=-0.6945*10^{-2}$$

Un número real *x* distinto de cero, se representa en notación científica normalizada en la forma:

$$x = \pm r * 10^{n}$$

En donde *r* es un número tal que $1/10 \leq r < 1$ y n es un entero (positivo, negativo o cero).

Del mismo modo se puede utilizar la notación científica en el sistema *binario*. En este caso:

$$x = \pm q * 2^{m}$$

En donde m es un número entero, q se denomina *mantisa* y el entero m *exponente*.

En una computadora *q* y *m* están representados como números en base 2. Puesto que la mantisa *q* está normalizada, en la representación binaria empleada se cumplirá que:

$$\frac{1}{2} \leq |q| < 1$$

I.V.III.II Representación de números con punto flotante.

En una computadora los números en punto flotante se representan de la manera descrita anteriormente, pero con ciertas restricciones sobre el número de dígitos de *q* y *m* impuestas por la cantidad de bits disponibles y que se van a emplear para almacenar un número.

Existen dos formatos definidos como precisión sencilla y doble precisión que trabajan con 32 bits y 64 bits respectivamente.

De igual forma que con un número entero el signo se representa por un cero cuando es positivo (+) y con uno si el signo es negativo (-).

La manera en que se distribuyen cada uno de los bits es la siguiente:

32 Bits	
Signo del número real x:	1 bit
Signo del exponente m:	1 bit
Exponente (entero |m|):	7 bits
Mantisa (número real |q|):	23 bits

Tabla I.2 Distribución Para 32 Bits.

64 Bits	
Signo del número real x:	1 bit
Signo del exponente m:	1 bit
Exponente (entero |m|):	10 bits
Mantisa (número real |q|):	52 bits

Tabla I.3 Distribución Para 64 Bits.

En la mayoría de los cálculos en punto flotante las mantisas se normalizan, es decir, se toman de forma que el bit más significativo (el primer bit) sea siempre 1. Por lo tanto, la mantisa *q* cumple siempre:

$$\frac{1}{2} \le |q| < 1$$

Como la mantisa siempre se representa normalizada, el primer bit en *q* es siempre 1, por lo que no es necesario almacenarlo proporcionando un bit significativo adicional.
Se dice que un número real expresado como aparece en la ecuación

$$x = \pm q * 2^m$$

y que satisface

$$\frac{1}{2} \le |q| < 1$$

Tiene la forma de punto flotante normalizado. Si además puede representarse exactamente con |*m*| ocupando 7 bits y |*q*| ocupando 24 bits, entonces es un número de computadora de 32 bits.
La restricción de que |*m*| no requiera más de 7 bits significa que:

$$|m| \le (1111111)_2 = 2^7 - 1 = 127$$

Ya que $2^{127} \approx 10^{38}$ con 32 bits se puede manejar números tan pequeños como 10^{-38} y tan grandes como 10^{38}. Este no es un intervalo de valores suficientemente grande, por lo que en muchos casos se deben utilizar programas escritos en doble precisión.

Como *q* debe representarse empleando no más de 24 bits significa que los números de computadora tienen una precisión limitada cercana a las siete cifras decimales, ya que el bit menos significativo de la mantisa representa unidades de:

$$2^{-24} \approx 10^{-7}$$

Por lo tanto, los números expresados mediante más de siete dígitos decimales serán aproximados cuando se almacenen en la memoria de la computadora. Esto genera un tipo de error conocido como error de redondeo, este debe considerarse pues aunque para cálculos pequeños pueden no ser significativos para programas más robustos en los cuales resultados previos se utilicen en iteraciones

posteriores para la solución del problema al que están dirigidos podrían llegar a ser relevantes, además existe otro tipo de error importante conocido como error de truncamiento.

Ambos tipos de error se tratarán mas adelante con mayor profundidad.

I.V.IV Machine epsilon.

El Machine Epsilon es el número decimal más pequeño que sumado a 1, la computadora arroja un valor diferente de uno, es decir, el número no es redondeado.

Representa la exactitud relativa de la aritmética de la computadora, es decir, define la precisión que tiene la computadora y depende de las características de esta, por ejemplo: procesador, memoria RAM, velocidad, etc.

El siguiente código realizado en FORTRAN® 90 permite conocer el Machine Epsilon de una maquina:

```
MACH1=2.0
MACH=1.0
I=0.0
DO WHILE (MACH1>1.0)
      MACH=MACH/2.0
      MACH1=MACH+1.0
      I=I+1.0
      WRITE(20,*)I,'    ',MACH
END DO
```

II. SOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES.

II.I Introducción.

Muchos problemas relacionados con la ingeniería en todas sus ramas, se pueden expresar en términos de sistemas de ecuaciones lineales simultáneas. Comúnmente se conocen dos métodos para resolver estos sistemas, la eliminación de las incógnitas mediante la combinación de ecuaciones, y el uso de determinantes que es lo que se conoce como Regla de Cramer. En este capítulo se abordarán otro tipo de métodos para resolver estos sistemas y que pueden ser implementados en programas de computadora.

II.II Conceptos básicos.

Un sistema de ecuaciones lineales es un conjunto de ecuaciones de la forma:

$$\begin{aligned}
&a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + ... + a_{1,n}x_n = b_1 \\
&a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + ... + a_{2,n}x_n = b_2 \\
&a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 + ... + a_{3,n}x_n = b_3 \quad \text{...II.1} \\
&\vdots \qquad \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots \\
&a_{n,1}x_1 + a_{n,2}x_2 + a_{n,3}x_3 + ... + a_{n,n}x_n = b_n
\end{aligned}$$

Que tiene la siguiente representación matricial:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \cdots a_{1n} \\ a_{21} & a_{22} & a_{23} \cdots a_{2n} \\ a_{31} & a_{32} & a_{33} \cdots a_{3n} \\ \vdots & \vdots & \vdots \quad \vdots \\ a_{n1} & a_{n2} & a_{n3} \cdots a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} \quad \text{...II.2}$$

Y que a su vez se puede representar como:

$$A\bar{x} = \bar{b} \ldots \text{II.3}$$

La solución del sistema de ecuaciones es un conjunto de n valores $x_1, x_2, x_3, \ldots, x_n$ que satisfacen simultáneamente a todas las ecuaciones.

La matriz 2 es una matriz de coeficientes, es decir, los elementos que conforman la matriz son los coeficientes de las incógnitas que forman el sistema de ecuaciones, si las constantes b del sistema de ecuaciones se agregan a la matriz de coeficientes se tendrá la matriz ampliada del sistema y tiene la siguiente forma:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \cdots a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} \cdots a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} \cdots a_{3n} & b_3 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} \cdots a_{nn} & b_n \end{bmatrix} \ldots \text{II.4}$$

II.II.I Método de Cramer.

La expresión general de la solución de un SEL por el método de Cramer es:

$$x_i = \frac{\begin{vmatrix} a_{11} \cdots a_{1,i-1} & b_1 & a_{1,i+1} \cdots a_{1n} \\ a_{21} \cdots a_{2,i-1} & b_2 & a_{2,i+1} \cdots a_{2n} \\ \vdots \quad \vdots & \vdots & \vdots \quad \vdots \\ a_{n1} \cdots a_{n,i-1} & b_n & a_{n,i+1} \cdots a_{nn} \end{vmatrix}}{A} \quad i = 1, \ldots, n \ldots \text{II.5}$$

Para aplicar este método en primer lugar hay que evaluar n+1 determinantes y luego realizar n divisiones.

Para el cálculo de los determinantes, una de las posibles técnicas necesita de $n!n$ multiplicaciones y $n!-1$ sumas. Por consiguiente, el método de Cramer necesita de:

- $(n+1)(n!-1)$ sumas
- $(n+1)(n!n)$ productos
- n divisiones

Cada operación requiere de una cantidad de recursos de la computadora para poder efectuarla, por lo que mientras más operaciones se realicen más complejo será para esta realizar dichas operaciones.

El número de operaciones que tendría que realizar una computadora para resolver un SEL de orden n utilizando el método de Cramer esta dado por la siguiente expresión:

$$T_c = (n+1)^2 n!$$

Los datos reportados en la siguiente tabla muestran el número de operaciones necesarias para resolver un SEL de 5, 10 y 100 incógnitas.

N	*Tc*
5	4319
10	$4*10^8$
100	10^{158}

Tabla II.1 Operaciones Elementales.[1]

Si se dispone de una computadora capaz de realizar 100 millones de operaciones en punto flotante por segundo, el sistema de n=100 necesitaría aproximadamente de 3 x 10^{142} años para ser resuelto, por lo que es evidente que el método de Cramer resulta inadecuado para resolver cualquier SEL.

II.II.II Conceptos de norma, número de condición y estabilidad.

II.II.II.I Norma de una matriz.

Cuando se manejan matrices o vectores es necesario explicar de alguna manera su magnitud en términos cuantitativos. Una medida de esa magnitud es la norma. Esta expresa la exactitud de la solución de un SEL en términos cuantitativos determinando la norma del vector error (la solución verdadera menos la solución aproximada). Las normas son también usadas para estudiar cuantitativamente la convergencia de un método iterativo para resolver SEL.

Para esto existen varias maneras de calcular la norma de una matriz, estas pueden ser:

- Norma de Frobenius.

$$A_f = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} \quad \text{...II.6}$$

- Norma de Magnitud Máxima.

$$A_1 = \underbrace{\max}_{1 \le j \le n} \sum_{i=1}^{n} a_{ij} = \text{Máximo de Suma de Columna...II.7}$$

$$A_1 = \underbrace{\max}_{1 \le j \le n} \sum_{i=1}^{n} a_{ij} = \text{Máximo de Suma de Renglones...II.8}$$

II.II.II.II Número de condición y error en la solución.
El número de condición de una matriz es una medida de confiabilidad de la matriz en el momento de realizar los cálculos. El número de condición esta dado por:

$$C(A) = A \cdot A^{-1} \ldots \text{II.9}$$

El número de condición para la matriz identidad es:

$$C(I) = 1.0$$

Por lo tanto la matriz identidad tiene el número de condición más bajo.

Un problema esta bien condicionado si cambios pequeños en la información de entrada ocasionan cambios pequeños en la salida. De otro modo se dice que está pobremente condicionado.

Un número de condición grande indica que la solución es sensible a pequeños cambios en el vector independiente. En los cálculos prolongados es probable que se realicen muchos redondeos y cada uno de ellos desempeñe el papel de un error de entrada para el resto del cálculo y cada uno tiene un efecto sobre la siguiente salida.
II.II.II.III Estabilidad.
Por esto se deben realizar algoritmos estables en que el efecto acumulativo de tales errores sea limitado, de modo que se generen buenos resultados, útiles para la solución del problema. Cuando el error acumulativo es grande el algoritmo es inestable.

II.II.III Solución numérica.

La eficacia de los algoritmos para la resolución de SEL se evalúa a partir de tres criterios fundamentales.
II.II.III.I Número de operaciones fundamentales.
Ligado al tiempo invertido en la realización de los cálculos por la computadora. Se deben tener en cuenta las operaciones elementales entre números de punto flotante. El número de operaciones es obviamente un excelente indicador del número de recursos invertido por la computadora.
II.II.III.II Necesidades de almacenamiento.
Inciden directamente en las limitaciones de la memoria de las diversas computadoras; los diferentes métodos que aquí se estudiarán requieren almacenar las matrices de diferentes maneras y esto varía considerablemente los requerimientos de memoria.
II.II.III.III Rango de aplicabilidad.
No todos los métodos sirven para cualquier matriz no singular; además, en función del método y de las propiedades de la matriz, la precisión de los resultados puede verse afectada, como ya se mencionó en el punto III.II.II.II pequeños errores de redondeo pueden producir errores desproporcionados en la solución numérica.

II.II.IV Métodos directos y métodos indirectos (Iterativos).

II.II.IV.I Métodos directos.

Son aquellos que permiten obtener la solución después de un número finito de operaciones aritméticas. Este número de operaciones es función del tamaño de la matriz. Algunos de estos métodos directos son:

Métodos Directos
Eliminación Gaussiana
Descomposición LU

Tabla II.2 Métodos Directos.[9]

II.II.IV.II Métodos indirectos.

Un método iterativo es el desarrollo de una solución aproximada para el sistema de ecuaciones. La aproximación es remplazada sistemáticamente hasta que converge a la respuesta correcta. Algunos de estos métodos indirectos son:

Métodos Iterativos
Método de Jacobi
Método de Gauss – Seidel
Método de Relajación Sucesiva (SOR)

Tabla II.3 Métodos Indirectos.[9]

II.II.V Tipos de matrices utilizadas en ingeniería.

Las matrices que son utilizadas en problemas de Ingeniería y en las Ciencias Aplicadas se clasifican en dos grandes categorías:

1. Matrices “llenas” con dimensiones pequeñas. Se entiende por “llenas” que tienen muy pocos elementos iguales cero y de dimensiones pequeñas se considera aquellas asociadas con SEL que tienen un número de ecuaciones de algunos miles como máximo. Aparecen en problemas estadísticos, matemáticos, físicos e ingenieriles.

2. Matrices “vacías” con dimensiones grandes. Se entiende por “vacías” que tienen muy pocos elementos diferentes de cero y están situados con cierta regularidad. En la mayoría de los casos el número de ecuaciones del SEL alcanza por lo menos las decenas miles y pueden llegar en ocasiones a los millones. Estas matrices son comunes en la resolución de ecuaciones diferenciales de problemas de Ingeniería.

En general para el primer grupo se aplican métodos directos para su solución mientras que para el segundo grupo se hace uso de métodos indirectos (iterativos).

II.II.VI Conceptos de matriz diagonal, triangular superior y triangular inferior.

II.II.VI.I Matriz diagonal.

Es una matriz que solo tiene elementos diferentes de cero en su diagonal principal

$$A = D = \begin{bmatrix} d_{11} & 0 & \cdots & \cdots & 0 \\ 0 & d_{22} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & d_{n-1,n-1} & 0 \\ 0 & \cdots & \cdots & 0 & d_{nn} \end{bmatrix} \quad \text{...II.10}$$

Y la solución se obtiene directamente:

$$x_i = \frac{b_i}{d_{ii}} \qquad i = 1,...,n \quad \text{...II.11}$$

Existe solución si todos los elementos de la diagonal son no nulos, existen algunos casos particulares de la matriz diagonal en los que no solo se tienen elementos diferentes de cero en la diagonal principal, si no que también se presentan diagonales con elementos no nulos en otras partes de la matriz, por ejemplo:

- Matriz Tridiagonal.

$$\begin{bmatrix} b_{11} & c_{12} & 0 & 0 & \cdots & 0 & 0 \\ a_{21} & b_{22} & c_{23} & \ddots & & & 0 \\ 0 & a_{32} & b_{33} & c_{34} & 0 & & \vdots \\ 0 & \ddots & a_{43} & b_{44} & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & & & \ddots & \ddots & \ddots & c_{n-1,m} \\ 0 & 0 & \cdots & 0 & 0 & a_{n,m-1} & b_{nm} \end{bmatrix}$$

- Matriz Pentagonal.

$$\begin{bmatrix} b_{11} & c_{12} & 0 & e_{14} & \cdots & 0 & 0 \\ a_{21} & b_{22} & c_{23} & 0 & e_{25} & & 0 \\ 0 & a_{32} & b_{33} & c_{34} & 0 & \ddots & \vdots \\ d_{41} & 0 & a_{43} & b_{44} & \ddots & 0 & d_{n-4,m} \\ \vdots & d_{52} & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & 0 & \ddots & \ddots & c_{n-1,m} \\ 0 & 0 & \cdots & d_{n,m-4} & 0 & a_{n,m-1} & b_{nm} \end{bmatrix}$$

- Matriz Heptagonal.

$$\begin{bmatrix} b_{11} & c_{12} & 0 & e_{14} & 0 & g_{16} & 0 \\ a_{21} & b_{22} & c_{23} & 0 & e_{25} & 0 & g_{n-5,m} \\ 0 & a_{32} & b_{33} & c_{34} & 0 & \ddots & 0 \\ d_{41} & 0 & a_{43} & b_{44} & \ddots & 0 & d_{n-4,m} \\ \vdots & d_{52} & 0 & \ddots & \ddots & \ddots & 0 \\ f_{61} & & \ddots & 0 & \ddots & \ddots & c_{n-1,m} \\ 0 & f_{n,m-5} & \cdots & d_{n,m-4} & 0 & a_{n,m-1} & b_{nm} \end{bmatrix}$$

II.II.VI.II Matriz triangular superior.

Es una matriz que tiene elementos por arriba de la diagonal principal diferentes a cero, y todos los que están por debajo iguales a cero. Suele denominarse a esta matriz como U.

$$A = U = \begin{bmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & u_{n-1,n-1} & u_{n-1,n} \\ 0 & \cdots & \cdots & 0 & u_{nn} \end{bmatrix} \quad \text{...II.11}$$

En este caso la solución de la última ecuación es trivial $x_n = b_n / u_{nn}$. Una vez conocido x_n, la penúltima ecuación (la $n-1$) solo tiene una incógnita que se deduce de forma sencilla. Conocidos ahora x_n y x_{n-1}, se pasa a la ecuación anterior (la $n-2$) y se resuelve para su única incógnita x_{n-2}. Retrocediendo progresivamente se obtiene el algoritmo de sustitución hacia atrás que se escribe de la siguiente forma:

$$x_n = b_n / u_{nn}$$

$$x_i = \frac{\left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right)}{u_{ii}} \quad i = n-1, n-2, ..., 1 \quad \text{...II.12}$$

II.II.VI.III Matriz triangular inferior.

Es una matriz que tiene elementos por abajo de la diagonal principal diferentes a cero, y todos los que están por arriba iguales a cero. Suele denominarse a esta matriz como L.

$$A = L = \begin{bmatrix} l_{11} & 0 & \cdots & \cdots & 0 \\ l_{21} & l_{22} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & l_{n-1,n-1} & 0 \\ l_{n1} & \cdots & \cdots & l_{n,n-1} & l_{nn} \end{bmatrix} \quad \text{...II.13}$$

De manera similar al anterior, existe un algoritmo, este se conoce como sustitución hacia delante:

$$x_1 = b_1 \big/ l_{11}$$

$$x_i = \frac{\left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right)}{l_{ii}} \quad i = 2,...,n \quad \text{...II.14}$$

Los algoritmos de sustitución hacia delante y hacia atrás son de gran importancia para los métodos de Eliminación.

II.III Método de eliminación Gaussiana.

Este método consiste en transformar la matriz de coeficientes A del SEL a una matriz U del tipo Triangular Superior (III.II.VI.II) haciendo uso de transformaciones elementales que permitan realizar esto.

Dichas transformaciones incluyen:

- Intercambio de un renglón por otro.
- Suma o resta entre dos renglones elemento a elemento.
- Multiplicación de los elementos de un renglón por un escalar, este escalar debe ser necesariamente diferente a cero.

Durante la transformación del sistema original A al sistema equivalente U las operaciones se realizan sobre la matriz A y al mismo tiempo sobre él termino independiente b, quedando un sistema equivalente a:

$$\begin{bmatrix} a_{11} & a_{22} & a_{33} & \cdots & a_{1n} \\ 0 & a'_{22} & a'_{23} & \cdots & a'_{2n} \\ \vdots & \ddots & a''_{33} & \cdots & a''_{3n} \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & a_{nn}^{n-1} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b_n^{n-1} \end{bmatrix} \quad \text{...II.15}$$

II.III.I Algoritmo.

El primer renglón se multiplica por a_{21}/a_{11} y se resta al segundo renglón para eliminar el primer elemento de este, del mismo modo se hace con el primer elemento de los demás renglones, restando el primer renglón multiplicado por a_{i1}/a_{11} , es importante recordar que también se debe afectar al vector $\bar{b}$, donde:

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}},\ a_{ij}^{k+1} = a_{ij}^{k} - \left(m_{ik}\right) a_{kj}^{k}, i = k+1, 3, 4, \ldots, n,\ \ j = k+1, 2, 3, \ldots, n+1 \ldots \text{II.16}$$

Esto reduciría el sistema a una matriz diagonal superior U, por lo que se utiliza la sustitución hacia atrás dada por la ecuación (II.12):

$$x_n = b_n / u_{nn}$$

$$x_i = \frac{\left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j\right)}{u_{ii}} \quad i = n-1, n-2, ..., 1 \ldots \text{II.12}$$

II.IV Método de descomposición LU.

Una matriz A puede factorizarse como el producto de una matriz triangular inferior L y otra triangular superior U:

$$A = L * U \ldots \text{II.17}$$

La descomposición de A seria en función de L y U de la siguiente manera:

$$L \equiv \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \text{ y } U \equiv \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \ddots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix}$$

Para realizar esta factorización existen tres esquemas:

1. Esquema ***Doolittle.*** Este esquema considera $l_{i,i} = 1$
2. Esquema ***Crout.*** Este esquema considera $u_{i,i} = 1$
3. Esquema ***Choleski.*** Este esquema considera $l_{i,i} = u_{i,i}$

II.IV.I Método Doolittle.

Dada la condición $l_{i,i}=1$, las matrices L y U que factorizan A tienen la siguiente forma:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{a2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \bullet \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \ddots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix} \quad \text{…II.18}$$

II.IV.I.I Algoritmo para descomponer una matriz A en LU.

De manera general la descomposición *LU* para una matriz de orden *N* es el que sigue:

1. Para evaluar $u_{i,j}$ y $l_{i,j}$ primero se multiplica el primer renglón de *L* por cada columna de *U*, así el primer renglón de *U*, para $j=1,...,N$ se obtiene con:

$$u_{1,j}=a_{1,j}\text{, para } j=1,...,N \quad \text{…II.19}$$

2. Se multiplica el segundo renglón y hasta el renglón *N* de *L* por la primera columna de *U*, así la primera columna de L para $i=2,...,N$ se obtiene con:

$$l_{i,1}=\frac{a_{i,1}}{u_{1,1}}\text{, para } i=2,...,N \quad \text{…II.20}$$

3. Se multiplica el segundo renglón de *L* por la segunda columna y hasta la columna *N* de *U* así el segundo renglón de *U* se obtiene con:

$$u_{2,j}=a_{2,j}-l_{2,1}u_{1,j}\text{, para } j=2,...,N \quad \text{…II.21}$$

4. Se multiplica el tercer renglón de *L* por la segunda columna de *U* así la segunda columna de *L* se obtiene con:

$$l_{i,2}=\frac{\left[a_{i,2}-l_{i,1}u_{1,2}\right]}{u_{2,2}}\text{, para } i=3,...,N \quad \text{…II.22}$$

5. El n – ésimo renglón de U se obtiene con:

$$u_{n,j}=a_{n,j}-\sum_{k=1}^{n-1}l_{n,k}u_{k,j}\text{ , para } j=n,...,N \quad \text{…II.23}$$

6. La n- ésima columna de L se obtiene con:

$$l_{i,n} = \frac{\left[a_{i,n} - \sum_{k=1}^{n-1} l_{i,k} u_{k,n} \right]}{u_{n,n}}, \text{ para } i = n+1,..., N \quad \ldots \text{II.24}$$

No se calculan los elementos $l_{i,i}$ de L puesto que estos son igual a uno.

II.IV.I.III Algoritmo para solucionar un SEL factorizado en LU.
Para resolver un SEL por el método LU expresado ahora por

$$LU\bar{x} = \bar{b} \ldots \text{II.25}$$

Se tiene el siguiente algoritmo

$$U\bar{x} = \bar{z} \ldots \text{II.26}$$

y

$$L\bar{z} = \bar{b} \ldots \text{II.27}$$

Como L es diagonal inferior con sus elementos de la diagonal principal iguales a uno obtener los valores del vector $\bar{z}$ es sencillo y se realiza mediante sustitución hacia delante dada por la ecuación (14):

$$z_1 = b_{11} \Big/ l_{11}$$
$$z_i = \frac{\left(b_i - \sum_{j=1}^{i-1} l_{ij} z_j \right)}{l_{ii}} \quad i = 2,..., n \ldots \text{II.14}$$

en donde los elemento l_{ii} son iguales a uno.

Una vez que se han obtenido los valores del vector $\bar{z}$ del sistema II.27 se puede resolver el sistema II.26 para encontrar los valores del vector $\bar{x}$ que resuelve el SEL utilizando la eliminación hacia atrás II.12

$$x_n = z_n \Big/ u_{nn}$$
$$x_i = \frac{\left(z_i - \sum_{j=i+1}^{n} u_{ij} x_j \right)}{u_{ii}} \quad i = n-1, n-2,...,1 \ldots \text{II.12}$$

II.IV.II Método Crout.

Dada la condición $u_{i,i}=1$, las matrices L y U que factorizan la matriz A tienen la siguiente forma:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{a2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \bullet \begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \ddots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

Por lo que es sencillo ver que el algoritmo que da solución a este tipo de sistemas es similar que el de Doolittle.

II.IV.III Método Choleski.

Dada la condición $l_{i,i}=u_{i,i}$ implica que $A=LL^T$, entonces las matrices L y U que factorizan la matriz A tienen la siguiente forma:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{a2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \bullet \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & l_{22} & \ddots & l_{n2} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & l_{nn} \end{bmatrix}$$

II.V Método de Jacobi.

El método de Jacobi es un método iterativo, y cuando converge se aproxima a la solución en cada iteración partiendo de un valor inicial, considerando la ecuación II.28:

$$A\bar{x}=\bar{b} \ldots \text{II.3}$$

se sustituye la matriz *A* por $A=U+R$ donde *U* es una matriz diagonal superior y *R* es otra matriz que contiene ceros en la diagonal principal y los restantes elementos de A, en sus demás elementos.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{a2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \ddots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 0 & r_{12} & \cdots & r_{1n} \\ r_{21} & 0 & \ddots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ r_{n1} & r_{a2} & \cdots & 0 \end{bmatrix} \text{ en donde } a_{i,j}=r_{i,j}$$

Sustituyendo en (3) se tiene:

$$(D+R)\bar{x} = \bar{b}$$
$$D\bar{x} + R\bar{x} = \bar{b}$$
$$D\bar{x} = \bar{b} - R\bar{x}$$

Premultiplicando por D^{-1}

$$\bar{x} = D^{-1}\bar{b} - D^{-1}R\bar{x}$$

Ecuación que se maneja como fórmula de recurrencia:

$$\bar{x}^{(k+1)} = D^{-1}\bar{b} - D^{-1}R\bar{x}^{(k)},\ k = 0,1,2,... \quad \text{II.29}$$

Esto quiere decir que del sistema II.3 se despeje x_1 de la primera ecuación, x_2 de la segunda, y así sucesivamente.

II.V.I Algoritmo.

Despejar cada una de las incógnitas siguiendo el patrón mencionado anteriormente se tiene:

$$\left.\begin{aligned}
x_1^{(k+1)} &= \frac{1}{a_{1,1}}\left(b_1 - a_{1,2}x_2^{(k)} - a_{1,3}x_3^{(k)} - \cdots - a_{1,n}x_n^{(k)}\right) \\
x_2^{(k+1)} &= \frac{1}{a_{2,2}}\left(b_2 - a_{2,1}x_1^{(k)} - a_{2,3}x_3^{(k)} - \cdots - a_{2,n}x_n^{(k)}\right) \\
x_3^{(k+1)} &= \frac{1}{a_{3,3}}\left(b_3 - a_{3,1}x_1^{(k)} - a_{3,2}x_2^{(k)} - \cdots - a_{3,n}x_n^{(k)}\right) \\
&\vdots \quad\quad \vdots \quad\quad\quad \vdots \quad\quad\quad \vdots \quad\quad\quad \vdots \\
x_n^{(k+1)} &= \frac{1}{a_{n,n}}\left(b_n - a_{n,1}x_1^{(k)} - a_{n,2}x_2^{(k)} - \cdots - a_{n,(n-1)}x_{n-1}^{(k)}\right) \\
k &= 0,1,2,\ldots
\end{aligned}\right\} \ldots \text{II.30}$$

Partiendo de una primera aproximación se tiene:

$$\bar{x}^{(0)} = \left[x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)}\right]^T \ldots \text{II.31}$$

Estos valores se sustituirán en los segundos miembros de las ecuaciones del sistema (14) para obtener la siguiente aproximación:

$$\bar{x}^{(1)} = \left[x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \ldots, x_n^{(1)}\right]^T \ldots \text{II.32}$$

Así la siguiente aproximación se obtiene sustituyendo $\bar{x}^{(1)}$:

$$\bar{x}^{(2)} = \left[x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \ldots, x_n^{(2)}\right]^T \quad \ldots \text{II.33}$$

Como ya se mencionó, este método al ser iterativo necesita de un valor inicial, este se obtiene de sustituir el vector $\bar{x}^{(0)} = [0,0,0,\ldots,0]^T$ en los segundos miembros de las ecuaciones del sistema II.30

$$\bar{x} = \left\{ \frac{b_1}{a_{1,1}}, \frac{b_2}{a_{2,2}}, \frac{b_3}{a_{3,3}}, \cdots, \frac{b_n}{a_{n,n}} \right\}^T \quad \ldots \text{II.34}$$

II.V.II Ejemplo.

Resolver el siguiente SEL por medio del método de Jacobi.

$$\left.\begin{aligned} 6x_1 + 2x_2 + x_3 &= 22 \\ -x_1 + 8x_2 + 2x_3 &= 30 \\ x_1 - x_2 + 6x_3 &= 23 \end{aligned}\right\}$$

Despejando x_1 de la primera ecuación, x_2 de la segunda y x_3 de la tercera, obteniendo:

$$x_1 = \frac{1}{6}\left(22 - 2x_2 - x_3\right)$$

$$x_2 = \frac{1}{8}\left(30 + x_1 + 2x_3\right)$$

$$x_3 = \frac{1}{6}\left(23 - x_1 + x_2\right)$$

Escribiendo este sistema en la forma II.30

$$x_1^{(k+1)} = \frac{1}{6}\left(22 - 2x_2^{(k)} - x_3^{(k)}\right)$$

$$x_2^{(k+1)} = \frac{1}{8}\left(30 - x_1^{(k)} - 2x_3^{(k)}\right)$$

$$x_3^{(k+1)} = \frac{1}{6}\left(23 - x_1^{(k)} - x_2^{(k)}\right)$$

$$k = 0,1,2,\ldots$$

Entonces el vector inicial es $\bar{x}^{(0)} = \begin{bmatrix} \frac{22}{6} & \frac{30}{8} & \frac{23}{6} \end{bmatrix}^T$, sustituyendo este primer vector inicial en las ecuaciones anteriores se tiene:

$$x_1^{(1)} = \frac{1}{6}\left(22 - 2x_2^{(0)} - x_3^{(0)}\right) = \frac{1}{6}\left(22 - 2\left(\frac{30}{8}\right) - \frac{23}{6}\right) = 1.778$$
$$x_2^{(1)} = \frac{1}{8}\left(30 + x_1^{(0)} - 2x_3^{(0)}\right) = \frac{1}{8}\left(30 + \frac{22}{6} - 2\left(\frac{23}{6}\right)\right) = 3.250$$
$$x_3^{(1)} = \frac{1}{6}\left(23 - x_1^{(0)} + x_2^{(0)}\right) = \frac{1}{6}\left(23 - \frac{22}{6} + \frac{30}{8}\right) = 3.847$$

Entonces para $k = 0$; $\bar{x}^{(1)} = \begin{bmatrix} 1.778 & 3.250 & 3.847 \end{bmatrix}^T$, continuando con el mismo procedimiento:

$$x_1^{(2)} = \frac{1}{6}\left(22 - 2x_2^{(1)} - x_3^{(1)}\right) = \frac{1}{6}\left(22 - 2(3.250) - 3.847\right) = 1.942$$
$$x_2^{(2)} = \frac{1}{8}\left(30 + x_1^{(1)} - 2x_3^{(1)}\right) = \frac{1}{8}\left(30 + 1.778 - 2(3.847)\right) = 3.011$$
$$x_3^{(2)} = \frac{1}{6}\left(23 - x_1^{(1)} + x_2^{(1)}\right) = \frac{1}{6}\left(23 - 1.778 + 3.250\right) = 4.079$$

Entonces para $k = 1$; $\bar{x}^{(2)} = \begin{bmatrix} 1.942 & 3.011 & 3.4079 \end{bmatrix}^T$, continuando con el mismo procedimiento se obtienen los siguientes vectores:

$$k = 2;\ \bar{x}^{(3)} = \begin{bmatrix} 1.989 & 2.973 & 4.012 \end{bmatrix}^T$$
$$k = 3;\ \bar{x}^{(4)} = \begin{bmatrix} 2.007 & 2.995 & 3.998 \end{bmatrix}^T$$
$$k = 4;\ \bar{x}^{(5)} = \begin{bmatrix} 2.002 & 3.001 & 3.998 \end{bmatrix}^T$$
$$k = 5;\ \bar{x}^{(6)} = \begin{bmatrix} 2.000 & 3.001 & 4.000 \end{bmatrix}^T$$
$$k = 6;\ \bar{x}^{(7)} = \begin{bmatrix} 2.000 & 3.000 & 4.000 \end{bmatrix}^T$$

Por lo que la solución del sistema es:

$$x_1 = 2.000$$
$$x_2 = 3.000$$
$$x_3 = 4.000$$

El número de iteraciones aumentara o disminuirá dependiendo de la tolerancia fijada.

II.VI Método de Gauss – Seidel.

El método de Gauss – Seidel es el mismo que el de Jacobi, la diferencia radica en que este cuando converge se aproxima más rápido a la solución.

II.VI.I Algoritmo.

La rápida convergencia se debe a que la componente $x_i^{(k+1)}$ una vez que fue calculada se utiliza inmediatamente dentro de la misma iteración, es decir:

$$\left.\begin{aligned}
x_1^{(k+1)} &= \frac{1}{a_{1,1}}\left(b_1 - a_{1,2}x_2^{(k)} - a_{1,3}x_3^{(k)} - \cdots - a_{1,n}x_n^{(k)}\right) \\
x_2^{(k+1)} &= \frac{1}{a_{2,2}}\left(b_2 - a_{2,1}x_1^{(k+1)} - a_{2,3}x_3^{(k)} - \cdots - a_{2,n}x_n^{(k)}\right) \\
x_3^{(k+1)} &= \frac{1}{a_{3,3}}\left(b_3 - a_{3,1}x_1^{(k+1)} - a_{3,2}x_2^{(k+1)} - \cdots - a_{3,n}x_n^{(k)}\right) \\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
x_n^{(k+1)} &= \frac{1}{a_{n,n}}\left(b_n - a_{n,1}x_1^{(k+1)} - a_{n,2}x_2^{(k+1)} - \cdots - a_{n,(n-1)}x_{n-1}^{(k+1)}\right) \\
k &= 0,1,2,\ldots
\end{aligned}\right\} \ldots \text{II.35}$$

II.VI.II Convergencia del Método de Jacobi y Gauss - Seidel.

Ambos métodos tienen como principal desventaja de no siempre converger a la solución del sistema, o en algunas ocasiones lo hacen pero de manera muy lenta.

Para poder aplicar estos métodos los elementos de la diagonal principal de la matriz *A* de coeficientes deben ser diferentes de cero.

Existe una condición necesaria para que estos converjan y consiste en que dichos elementos de la diagonal principal además de ser diferentes de cero sean mayores en valor absoluto que la suma de los demás elementos del renglón correspondiente.

Cuando esta condición se cumple garantiza la convergencia, pero de no cumplirse no se puede afirmar la no convergencia.

II.VII Método de Relajación Sucesiva (SOR).

Es un método usado para acelerar la convergencia del método del Gauss-Seidel para solucionar un sistema lineal de ecuaciones. Un método similar se puede utilizar para cualquier proceso iterativo de lenta convergencia.

II.VII.I Algoritmo.

Se busca escribir la matriz A del sistema como:

$$A = D + L + U \ldots \text{II.36}$$

Donde D, L y U son matrices diagonal, triangular inferior y triangular superior respectivamente.

Entonces la iteración dada para la relajación sucesiva (SOR) se define por:

$$(D + \omega L)\phi^{(k+1)} = (-\omega U + (1-\omega)D)\phi^{(k)} + \omega b \ldots \text{II.37}$$

En donde $\varphi^{(k)}$ representa la k – ésima iteración, y ω es un factor de relajación. Esta iteración reduce a la iteración del Gauss-Seidel para $\omega = 1$.

La elección del factor de la relajación no es fácil, y depende de las características de la matriz de coeficientes. Para las matrices simétricas, positivamente definidas puede demostrarse para $0 < \omega < 2$ se tendrá la convergencia,

Como en el método del Gauss-Seidel, no es necesario solucionar un sistema lineal para obtener la iteración II.37; las nuevas iteraciones se pueden obtener con la fórmula:

$$\phi_i^{(k+1)} = (1-\omega)\phi_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\phi_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\phi_j^{(k)} \right),\ i = 1,2,3,\ldots,n \ldots \text{II.38}$$

II.VIII Subrutina NSPIV.[14,15]

Este método permite resolver un sistema lineal de ecuaciones en forma directa, tomando ventaja de la dispersión de la matriz de coeficientes para eliminar operaciones con elementos iguales a cero.

II.VIII.I Algoritmo.

Considerando el sistema:

$$A\bar{x} = \bar{b} \ldots \text{II.3}$$

La eliminación Gaussiana se efectúa con intercambios de columnas en la matriz A para obtener una factorización de la forma:

$$AQ = LU \ldots \text{II.39}$$

En donde L y U son matrices triangulares inferior y superior respectivamente, Q es una matriz de permutación, esto permite establecer el siguiente sistema:

$$LUQ^T\bar{x} = \bar{b} \ldots \text{II.40}$$

Que a su vez se puede descomponer en:

$$L\bar{y} = \bar{b} \ldots \text{II.41}$$

Y

$$UQ^T \bar{x} = \bar{y} \ldots \text{II.42}$$

Primero se resuelve el sistema II.41 mediante una sustitución hacia adelante para obtener el vector $\bar{y}$, posteriormente se calcula el vector $\bar{x}$ del sistema II.42 mediante una sustitución hacia atrás.

Por lo que se requiere de tres vectores en donde se almacenaran los elementos diferentes de cero de la matriz.

II.VIII.II Subrutina de solución.

La siguiente subrutina esta escrita en FORTRAN® y se puede encontrar detallada en la dirección ***http://www.netlib.org/toms/533*** o en el articulo ***"ALGORITHM 533, NSPIV, A Fortran Subroutine for Sparse Gaussian Elimination UIT Partial Pivoting"*** escrito por ***"Andrew H. Sherman"*** publicado en las paginas de ***"ACM Transactions on Mathematical Software Vol.4, No. 4"***, en diciembre de 1978.

```
SUBROUTINE NSPIV (N,IA,JA,A,B,MAX,R,C,IC,X,ITEMP,RTEMP,IERR)          1
    REAL A(1),B(1),X(1),RTEMP(1)
    INTEGER IA(1),JA(1),R(1),C(1),IC(1),ITEMP(1)
    INTEGER IU,JU,U,Y,P

    Y = 1
    U = Y + N
    P = 1
    IU = P + N + 1
    JU = IU + N + 1
    CALL NSPIV1 (N,IA,JA,A,B,MAX,R,C,IC,X,RTEMP(Y),ITEMP(P),
    RETURN
    END
    SUBROUTINE NSPIV1 (N,IA,JA,A,B,MAX,R,C,IC,X,Y,P,IU,JU,U,IERR)        109
    REAL A(1),B(1),U(1),X(1),Y(1)
    REAL DK,LKI,ONE,XPV,XPVMAX,YK,ZERO
    INTEGER C(1),IA(1),IC(1),IU(1),JA(1),JU(1),P(1),R(1)
    INTEGER CK,PK,PPK,PV,V,VI,VJ,VK

    IF (N .EQ. 0) GO TO 1001
    ONE = 1.0
    ZERO = 0.0

    DO 10 J=1,N
      X(J) = ZERO
10    CONTINUE
    IU(1) = 1
    JUPTR = 0
    DO 170 K=1,N
      P(N+1) = N+1
      VK = R(K)

      JMIN = IA(VK)
      JMAX = IA(VK+1) - 1
      IF (JMIN .GT. JMAX)  GO TO 1002
```

```
      J = JMAX

 20     JAJ = JA(J)
        VJ = IC(JAJ)
        X(VJ) = A(J)

        PPK = N+1
 30     PK = PPK
        PPK = P(PK)
        IF (PPK - VJ)  30,1003,40
 40     P(VJ) = PPK
        P(PK) = VJ
        J = J - 1
        IF (J .GE. JMIN) GO TO 20

      VI = N+1
      YK = B(VK)
 50   VI = P(VI)
      IF (VI .GE. K) GO TO 110

      LKI = - X(VI)
      X(VI) = ZERO

      YK = YK + LKI * Y(VI)
      PPK = VI
      JMIN = IU(VI)
      JMAX = IU(VI+1) - 1
      IF (JMIN .GT. JMAX)  GO TO 50
      DO 100 J=JMIN,JMAX
        JUJ = JU(J)
        VJ = IC(JUJ)

        IF (X(VJ) .NE. ZERO)  GO TO 90

        IF (VJ - PPK) 60,90,70
 60     PPK = VI
 70     PK = PPK
        PPK = P(PK)
        IF (PPK - VJ)  70,90,80
 80     P(VJ) = PPK
        P(PK) = VJ
        PPK = VJ

 90     X(VJ) = X(VJ) + LKI * U(J)
 100    CONTINUE
      GO TO 50

 110  IF (VI .GT. N) GO TO 1004
      XPVMAX = ABS(X(VI))
      MAXC = VI
      NZCNT = 0
      PV = VI
 120    V = PV
        PV = P(PV)
        IF (PV .GT. N) GO TO 130
        NZCNT = NZCNT + 1
        XPV = ABS(X(PV))
        IF (XPV .LE. XPVMAX) GO TO 120
        XPVMAX = XPV
        MAXC = PV
        MAXCL = V
        GO TO 120
 130  IF (XPVMAX .EQ. ZERO) GO TO 1004

      IF (VI .EQ. K) GO TO 140
      IF (VI .EQ. MAXC) GO TO 140
      P(MAXCL) = P(MAXC)
      GO TO 150
 140  VI = P(VI)
```

```
150   DK = ONE / X(MAXC)
      X(MAXC) = X(K)
      I = C(K)
      C(K) = C(MAXC)
      C(MAXC) = I
      CK = C(K)
      IC(CK) = K
      IC(I) = MAXC
      X(K) = ZERO

      Y(K) = YK * DK

      IU(K+1) = IU(K) + NZCNT
      IF (IU(K+1) .GT. MAX+1) GO TO 1005

      IF (VI .GT. N)  GO TO 170
      J = VI
160     JUPTR = JUPTR + 1
        JU(JUPTR) = C(J)
        U(JUPTR) = X(J) * DK
        X(J) = ZERO
        J = P(J)
        IF (J .LE. N) GO TO 160
170   CONTINUE

    K = N
    DO 200 I=1,N
      YK = Y(K)
      JMIN = IU(K)
      JMAX = IU(K+1) - 1
      IF (JMIN .GT. JMAX)  GO TO 190
      DO 180 J=JMIN,JMAX
        JUJ = JU(J)
        JUJ = IC(JUJ)
        YK = YK - U(J) * Y(JUJ)
180     CONTINUE
190   Y(K) = YK
      CK = C(K)
      X(CK) = YK
      K = K-1
200   CONTINUE

    IERR = IU(N+1) - IU(1)
    RETURN

1001 IERR = 0
    RETURN

1002 IERR = -K
    RETURN

1003 IERR = -(N+K)
    RETURN

1004 IERR = -(2*N+K)
    RETURN

1005 IERR = -(3*N+K)
    RETURN
    END
```

III. SOLUCIÓN DE SISTEMAS DE ECUACIONES NO LINEALES.

III.I Introducción.

Un sistema físico, matemático o de otro tipo es no lineal cuando las ecuaciones de movimiento, evolución o comportamiento que regulan su comportamiento son no lineales. En particular, el comportamiento de sistemas no lineales no está sujeto al principio de superposición, como lo es un sistema lineal.

La linealidad de un sistema permite a los investigadores hacer ciertas suposiciones matemáticas y aproximaciones, permitiendo un cálculo más sencillo de los resultados. Desde que los sistemas no lineales no son iguales a la suma de sus partes, usualmente son difíciles (o imposibles) de modelar, y sus comportamientos con respecto a una variable dada (por ejemplo, el tiempo) es extremadamente difícil de predecir.[15]

III.II Conceptos Básicos.

III.II.I Ecuación no lineal.

Una ecuación del tipo:

$$y = x^3$$

$$y = senx$$

$$z = e^{yx}$$

Es conocida como no lineal en x, y, z, … porque no puede escribirse como:

$$ax + by + cz + \cdots = cte.$$

Que representa una ecuación lineal donde a, b, c, … son constantes.

III.II.II Sistema de ecuaciones no lineales.

Un sistema de n ecuaciones con n incógnitas $x_1, x_2, \ldots, x_n$ se llama no lineal si una o mas de las ecuaciones es no lineal. Llevando todos los términos diferentes de cero al miembro izquierdo de todas las ecuaciones, cualquier sistema no lineal de n*n puede escribirse en la forma general:

$$\left.\begin{array}{l} f_1(x_1, x_2, x_3, \ldots, x_n) = 0 \\ f_2(x_1, x_2, x_3, \ldots, x_n) = 0 \\ \vdots \qquad\qquad\qquad \vdots \\ f_n(x_1, x_2, x_3, \ldots, x_n) = 0 \end{array}\right\} \ldots \text{III.1}$$

En forma reducida:

$$\begin{cases} f_1(x) = 0 \\ f_2(x) = 0 \\ f_3(x) = 0 \\ \vdots \\ f_n(x) = 0 \end{cases} \text{, donde } x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \ldots \text{III.2}$$

Un vector $\bar{x} = [\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_n]^T$ que satisface $f(\bar{x}) = 0$ se llama raíz del sistema no lineal.

III.III Solución de una ecuación no lineal.

Como ya se mencionó las soluciones de una ecuación no lineal se conocen como raíces.

La mayoría de este tipo de ecuaciones carecen de solución exacta, por ejemplo, la solución analítica de las ecuaciones polinomiales existe solo hasta el cuarto orden, y no existen soluciones en forma exacta para ordenes mayores, es por eso que para encontrar las raíces se hace uso de métodos computacionales basados en procesos iterativos.

III.III.I Métodos para resolver una ecuación no lineal.

Existe varios métodos numéricos diseñados para encontrar raíces, cada uno de ellos tiene características especificas que reflejan sus capacidades y limitaciones para encontrar las raíces de una ecuación no lineal.

En este trabajo solo se analizará el método de Newton o mejor conocido como Newton – Raphson y el método de la Secante por la similitud existente entre ellos.

III.III.II Método de Newton – Raphson.

Este método encuentra una raíz siempre y cuando se conozca una estimación inicial para la raíz deseada. Utiliza rectas tangentes que se evalúan analíticamente.

Si $f(x)$ es diferenciable en x_k entonces se puede aplicar el método a partir de las siguientes consideraciones, si se desea encontrar un raíz tal que $f(x)=0$, al utilizar el desarrollo de la serie Taylor de $f(x)$ en torno a una estimación x_0, se tiene:

$$f(x)= f(x_0)+ f'(x_0)(x-x_0)+O(h^2)\ldots\text{III.3}$$

Donde $h = x - x_0$.

Si se despeja x de la ecuación III.3 no se obtiene el valor exacto, esto es debido al error de truncamiento asociado a la serie de Taylor, sin embargo la solución se aproxima mas al valor de x que la aproximación obtenida con el valor de x_0, entonces al iterar utilizando el nuevo valor como una nueva estimación se obtiene una mejor aproximación, esto se repite hasta que se cumpla una cierta tolerancia.

III.III.II.I Algoritmo.

El método se presenta de manera gráfica en la siguiente figura:

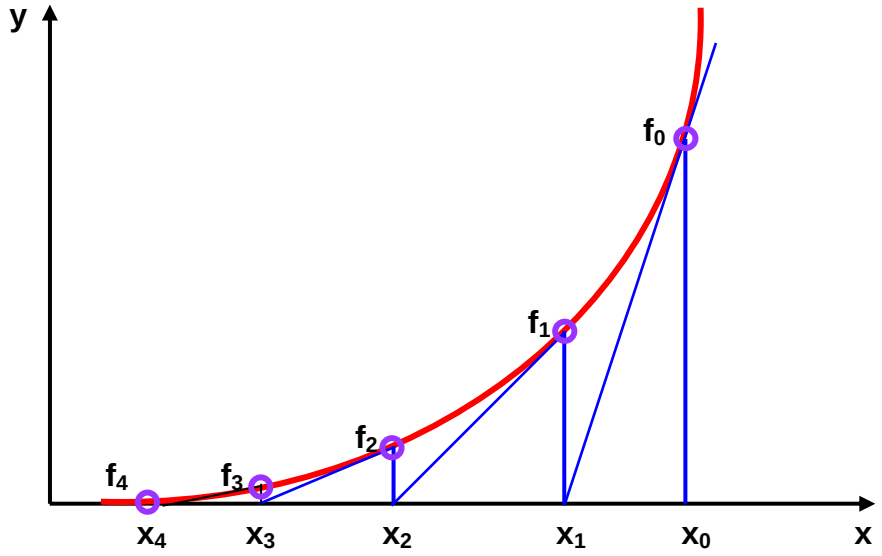


Figura III.1 Método de Newton – Raspón.

El valor de x_0 es una estimación inicial para la raíz deseada. A continuación se obtiene la función lineal que pasa por (x_0, y_0) en forma tangencial. La intersección de la recta tangente con el eje x se denota como x_1 y se considera como una

aproximación de la raíz. Se itera siguiendo el mismo procedimiento utilizando el valor actualizado como estimación para el siguiente ciclo de iteración.

La recta tangente que pasa por $(x_0, f(x_0))$ es:

$$g(x) = f'(x_0)(x - x_0) + f(x_0) \ldots \text{III.4}$$

La raíz de $g(x) = 0$ denotada por x_1 satisface

$$f'(x_0)(x - x_0) + f(x_0) = 0 \ldots \text{III.5}$$

Al resolver la ecuación anterior se obtiene

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \ldots \text{III.6}$$

Escribiendo esta ecuación para las aproximaciones sucesivas a la raíz se tiene:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})} \ldots \text{III.7}$$

El método de Newton – Raphson puede no converger o encontrar una raíz no deseada, esto depende de la estimación inicial.

III.III.II.II Ejemplo.
Encontrar una raiz para el polinomio $x^3 - x^2 - 2x + 1 = 0$ utilizando el método de Newton – Raphson.

Entonces $f(x) = x^3 - x^2 - 2x + 1$ y $f'(x) = 3x^2 - 2x - 2$, considerando $x_0 = 1$, $x_0 = 1.4$ y $x_0 = -2.2$, las iteraciones realizadas para encontrar las tres raíces del polinomio indicado se resumen en las siguientes tablas:

x_0	$f(x)$	$f'(x)$	x_{i-1}	x_i
1.000	-1.00000	-1.00000	-----	0.00000
	1.00000	-2.00000	0.00000	0.50000
	-0.12500	-2.25000	0.50000	0.44444
	0.00137	-2.29630	0.44444	0.44504
	0.00000	-2.29590	0.44504	0.44504
	0.00000	-2.29590	0.44504	0.44504

Tabla III.1 Ejemplo Newton – Raphson $x_0 = 1$.

x_0	$f(x)$	$f'(x)$	x_{i-1}	x_i
1.400	-1.01600	1.08000	-----	2.34074
	3.66453	9.75572	2.34074	1.96511
	0.79672	5.65477	1.96511	1.82422
	0.09438	4.33489	1.82422	1.80245
	0.00211	4.14156	1.80245	1.80194
	0.00000	4.13707	1.80194	1.80194

Tabla III.2 Ejemplo Newton – Raphson $x_0 = 1.4$**.**

x_0	$f(x)$	$f'(x)$	x_{i-1}	x_i
-2.200	-10.08800	16.92000	-----	-1.60378
	-2.48967	8.92392	-1.60378	-1.32479
	-0.43061	5.91482	-1.32479	-1.25199
	-0.02598	5.20644	-1.25199	-1.24700
	-0.00012	5.15905	-1.24700	-1.24698
	0.00000	5.15883	-1.24698	-1.24698

Tabla III.3 Ejemplo Newton – Raphson $x_0 = -2.2$**.**

Entonces las raíces son $x_1 = 0.44504$, $x_2 = 1.80194$ y $x_3 = -1.24698$, el método converge rápidamente si se elige un valor inicial cercano a la raíz deseada.

III.III.III Método de la secante.

Este método es similar al de Newton – Raphson y la principal diferencia es que $f'(x)$ se aproxima utilizando los dos valores de iteraciones consecutivas de $f(x)$, esto permite eliminar la necesidad de evaluar tanto a $f(x)$ como a $f'(x)$ en cada iteración, debido a esto el método de la secante es mas eficiente, en especial cuando evaluar $f(x)$ implica mucho tiempo.

III.III.III.I Algoritmo.

Las aproximaciones sucesivas para la raíz están dadas por:

$$x_n = x_{n-1} - y_{n-1}\frac{x_{n-1} - x_{n-2}}{y_{n-1} - y_{n-2}},\ n = 2,3\ldots \quad \ldots \text{III.8}$$

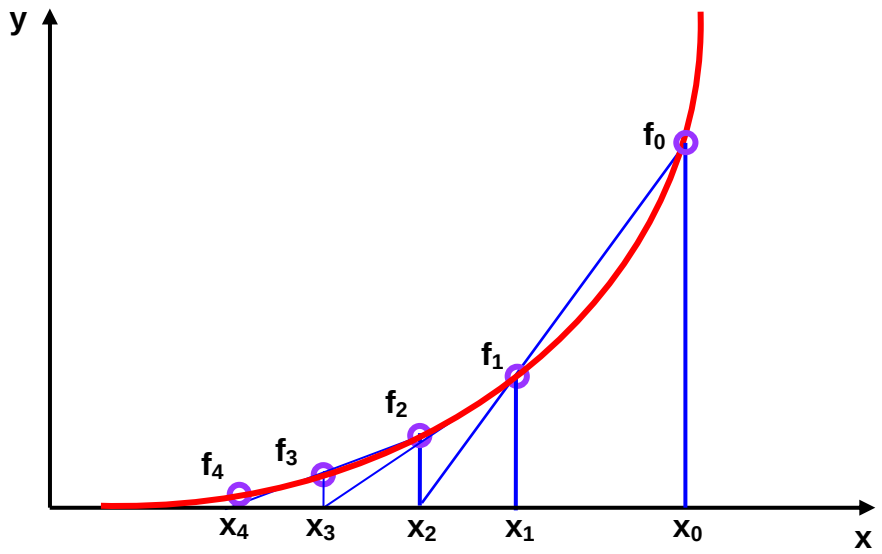


Figura III.2 Método de la Secante.

III.IV Solución de un sistema de ecuaciones no lineales.

En muchas ocasiones aparece la necesidad de resolver sistemas de ecuaciones algebraicas no lineales, Una de las dificultades asociadas con estos sistemas es que no es fácil saber si existen raíces y cuántas son.

*III.VI.I Método de Newton – Raphson para sistemas de n*n.*

Este método tiene algunas similitudes con el empleado para resolver una sola ecuación no lineal, parte de una aproximación inicial y después de un número finito de iteraciones el método converge.

III.IV.I.I Algoritmo.

Sea $\bar{x} = [\bar{x}_1 \quad \bar{x}_2 \quad \dots \quad \bar{x}_n]^T$ una raíz del sistema no lineal de $n*n \quad f(x) = 0$ y cuya iésima ecuación es:

$$f_i(x) = f_i(x_1, x_2, \dots, x_n) = 0, \; i = 1, \dots, n \; \dots \text{III.9}$$

Suponiendo que x_k es una aproximación actual de $\bar{x}$.

Para obtener una mejor aproximación que c denominada x_{k+1} es necesario resolver un sistema lineal que se aproxime al sistema III.9 para x cerca de x_k.

Si $x = x_k + dx$, donde $dx = [dx_1 \quad dx_2 \quad \dots \quad dx_n]^T$, se puede aproximar a la ecuación

exacta $f_i(x_k + dx) = 0$ y usando la diferencial total se tiene:

$$f_i(x_k) + \frac{\partial f_i(x_k)}{\partial x_1} dx_1 + \frac{\partial f_i(x_k)}{\partial x_2} dx_2 + \cdots + \frac{\partial f_i(x_k)}{\partial x_n} dx_n = 0,\ i = 1,\ldots,n \quad \ldots \text{III.10}$$

Este sistema es lineal en $dx_1, dx_2, \ldots, dx_n$, y su forma matricial es:

$$\underbrace{\begin{bmatrix} \frac{\partial f_1(x_k)}{\partial x_1} & \frac{\partial f_1(x_k)}{\partial x_2} & \cdots & \frac{\partial f_1(x_k)}{\partial x_n} \\ \frac{\partial f_2(x_k)}{\partial x_1} & \frac{\partial f_2(x_k)}{\partial x_2} & \cdots & \frac{\partial f_2(x_k)}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n(x_k)}{\partial x_1} & \frac{\partial f_n(x_k)}{\partial x_2} & \cdots & \frac{\partial f_n(x_k)}{\partial x_n} \end{bmatrix}}_{J = f'(x_k)} \underbrace{\begin{bmatrix} dx_1 \\ dx_2 \\ \vdots \\ dx_n \end{bmatrix}}_{dx} = -\underbrace{\begin{bmatrix} f_1(x_k) \\ f_2(x_k) \\ \vdots \\ f_n(x_k) \end{bmatrix}}_{-f(x_k)} \quad \ldots \text{III.11}$$

En donde la matriz $J = f'(x_k)$ es la matriz Jacobiana del sistema no lineal $f(x) = 0$ en x_k. Se puede ver que el renglón i contiene las parciales de $f_1(x), f_2(x), \ldots, f_i(x)$, con $i = 1,\ldots,n$, mientras que la columna j contiene las parciales con respecto a $x_1, x_2, \ldots, x_n$, con $j = 1,2,\ldots,n$.

Por lo tanto se puede obtener x_{k+1} a partir de x_k de la siguiente forma:

$$x_{k+1} = x_k + dx_k \quad \ldots \text{III.12}$$

Donde dx_k es la solución de $f'(x_k)dx = -f(x_k)$.

IV. INTERPOLACIÓN NUMÉRICA

IV.I Introducción.

Interpolación es la construcción de nuevos puntos partiendo del conocimiento de un conjunto discreto de puntos. En ingeniería y otras ciencias es frecuente disponer de un cierto número de datos obtenidos por muestreo o a partir de un experimento y pretender construir una función que los ajuste.

Otro problema estrechamente ligado con el de la interpolación es la aproximación de una función complicada por una más simple. Si se tiene una función cuyo cálculo resulta costoso, entonces, a partir de un cierto número de valores de dicha función se pueden interpolar dichos datos construyendo una función más simple.

En general no se obtienen los mismos valores evaluando la función obtenida que si se evaluara la función original, sin embargo dependiendo de las características del problema y del método de interpolación usado, la eficiencia en la obtención de dicha información aumenta compensando el error cometido.

IV.II Lineal y doble interpolación.

IV.II.I Interpolación lineal para un conjunto de datos conocidos.

Como se mencionó anteriormente, en ocasiones se conocen un cierto número de datos obtenidos experimentalmente, estos pueden llegar a tener un comportamiento lineal, si se desea conocer otro valor no contenido dentro de los datos existentes, este se puede obtener realizando una interpolación. Una tabla de valores cuyo comportamiento es lineal:

X	Y
x_1	y_1
x_2	y_2
x_3	y_3

Tabla IV.1 Tabla de Valores.

Y se desea conocer un valor x que esta entre (x_1, y_1) y (x_2, y_2):

X	Y
x_1	y_1
x	$Y_?$
x_2	y_2

Tabla IV.2 Tabla con Valor Desconocido.

Se puede interpolar haciendo uso de una ecuación de la forma:

$$y = mx + b \ldots \text{IV.1}$$

Que es la ecuación ordenada al origen de una recta.

IV.II.I.I Algoritmo.

La pendiente m de la ecuación IV.1 conocidos dos puntos se define como:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \ldots \text{IV.2}$$

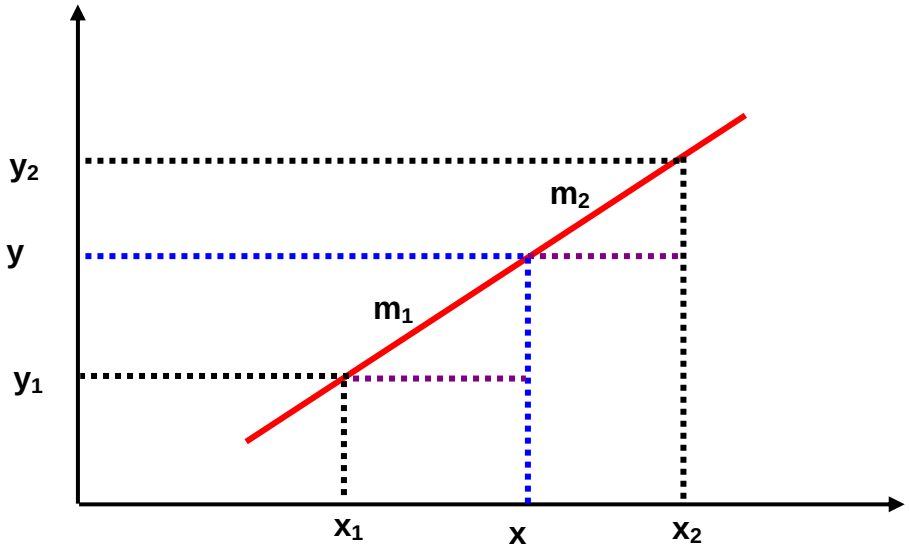


Figura IV.1 Recta de Interpolación a Partir de Dos Puntos.

A partir de la figura se obtienen m_1 y m_2 definidas por los puntos (x_1, y_1), (x, y) y (x_2, y_2):

$$m_1 = \frac{y - y_1}{x - x_1} \ldots \text{IV.3 y } m_2 = \frac{y - y_2}{x - x_2} \ldots \text{IV.4}$$

Igualando estas ecuaciones, pues representan la misma pendiente m de la recta definida por los puntos conocidos:

$$\frac{y-y_1}{x-x_1}=\frac{y-y_2}{x-x_2} \quad \text{...IV.5}$$

Por último haciendo el desarrollo algebraico para despejar y se tiene:

$$y=\frac{y_1(x-x_2)+y_2(x_1-x)}{(x_1-x_2)} \quad \text{...IV.6}$$

Que es la ecuación de interpolación lineal para un conjunto de datos conocidos.

IV.II.I.II Ejemplo.

Con los datos de la siguiente tabla obtener el valor de y para $x=3.4$.

X	Y
1	6
2	5
3	4
4	3
5	2
6	1

Tabla IV.3 Ejemplo Interpolación Lineal.

Para $x=3.4$ se toman los valores dados por los puntos (x_3,y_3) y (x_4,y_4), se sustituyen en la ecuación IV.6.

$$y=\frac{4(3.4-4)+3(3-3.4)}{(3-4)}$$

$$y=3.6$$

IV.II.II Interpolación lineal para una función dada.

La interpolación lineal aplicada a una función es la base para varios modelos numéricos, por ejemplo, al integrar la interpolación lineal se obtiene el método de integración conocido como regla del trapecio.

En este caso la interpolación lineal para una función da como resultado una recta que se ajusta a dos puntos dados.

IV.II.II.I Algoritmo.[2]

Este tipo de interpolación se muestra en la figura:

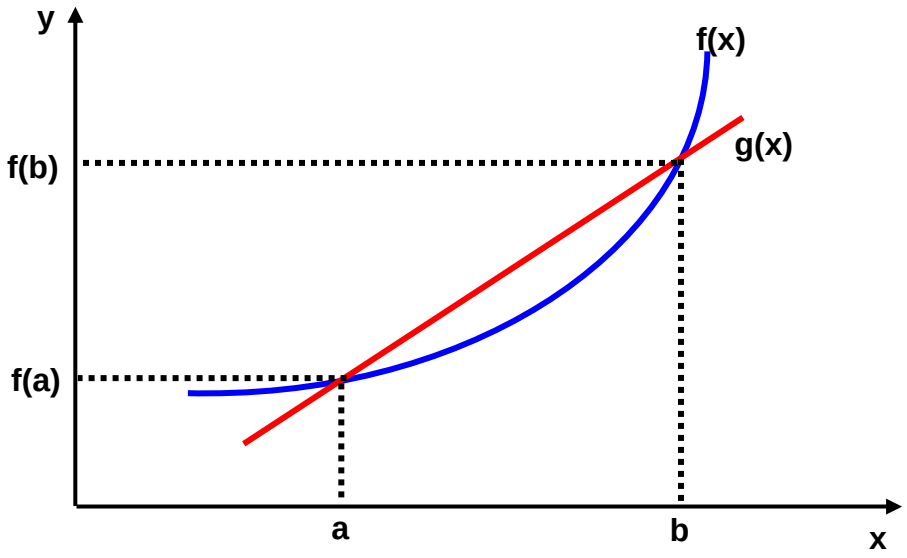


Figura IV.2 Recta de Interpolación Lineal a Partir de Una Función.[2]

Y esta dado por la ecuación:

$$g(x)=\frac{b-x}{b-a}f(a)+\frac{x-a}{b-a}f(b)\ldots\text{IV.7}$$

Donde $f(a)$ y $f(b)$ son valores conocidos de $f(x)$ en $x=a$ y $x=b$ respectivamente.

IV.II.III Doble interpolación.

Este tipo de interpolación se utiliza cuando se tienen datos que dependen de dos variables, es decir de una función $f(x,y)$.

IV.II.III.I Algoritmo.

Se conocen los valores de la función $f(x,y)$ en una malla rectangular de (x_i, y_k). Se denota el valor en el punto (x_i, y_j) como $f_{i,j}=(x_i, y_j)$. La doble interpolación se lleva a cabo en dos etapas, en las que se utiliza una interpolación en dimensión uno.

Suponiendo que es necesario estimar la función en un punto localizado en el rectángulo definido por $x_{i-1}\le x\le x_i$ y $y_{j-1}\le y\le y_j$ como se muestra en la figura.

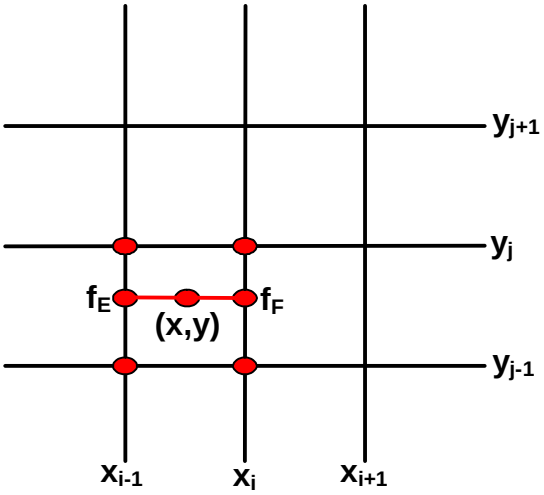


Figura IV.3 Doble Interpolación.

Combinando estos dos pasos en una sola ecuación se tiene:

$$g(x,y)=\frac{\left[(x_i-x)(y_j-y)f_{i-1,j-1}+(x_i-x)(y-y_{j-1})f_{i-1,j}+(x-x_{i-1})(y_j-y)f_{i,j-1}+(x-x_{i-1})(y-y_{j-1})f_{i,j}\right]}{\left[(x_i-x_{i-1})(y_j-y_{j-1})\right]}$$

… IV.8

IV.III Método de Lagrange.

Uno de los métodos más importantes para encontrar una función que pase a través de un número N de datos consiste en ajustar un polinomio, tal y como se muestra en la figura:

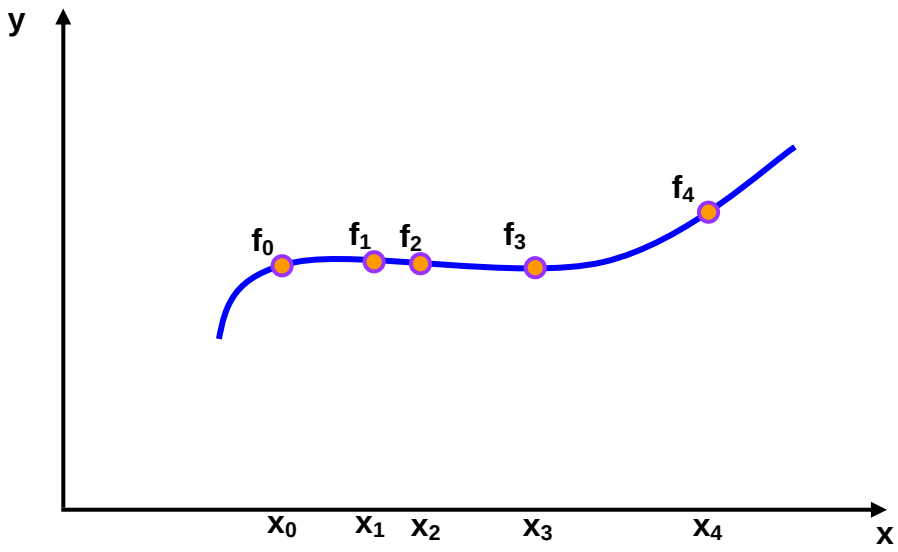


Figura IV.4 Función Ajustada a un Número n de Datos.[2]

IV.III.I Algoritmo.

El producto de factores dados por:

$$V_0(x) = (x - x_1)(x - x_2)\cdots(x - x_N) \quad \text{...IV.9}$$

Que se refiera a los $N+1$ puntos dados.

La función V_0 es un polinomio de orden N de x, y se anula en $x = x_1, x_2, \ldots, x_N$, dividiendo $V_0(x)$ entre $V_0(x_0)$ se tiene:

$$V_0(x) = \frac{(x - x_1)(x - x_2)\cdots(x - x_N)}{(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_N)} \quad \text{...IV.10}$$

La función IV.10 toma para $x = x_0$ el valor de uno, y para $x = x_1, x = x_2, \ldots, x = x_N$ el valor de cero.

Reescribiendo la ecuación IV.10 para i – ésimo elemento:

$$V_i(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_N)}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_N)} \quad \text{...IV.11}$$

La función $V_i(x)$ es un polinomio de orden N y toma el valor de uno en $x = x_i$ y de cero en $x = x_j$ para $j \neq i$. Entonces, multiplicando $V_0(x), V_1(x), \ldots, V_N(x)$ por $f_0, f_1, \ldots, f_N$, respectivamente, y sumando estos resultados se tiene un polinomio cuyo orden es igual a N como máximo e igual a f_i para $i = 0,1,2,\ldots,N$.

Según lo anterior la fórmula de interpolación de Lagrange de orden N es:

$$\begin{aligned} g(x) = {} & \frac{(x - x_1)(x - x_2)\cdots(x - x_N)}{(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_N)} f_0 + \frac{(x - x_0)(x - x_2)\cdots(x - x_N)}{(x_1 - x_0)(x_1 - x_2)\cdots(x_1 - x_N)} f_1 \\ & \vdots \\ & + \frac{(x - x_0)(x - x_1)\cdots(x - x_{N-1})}{(x_N - x_0)(x_N - x_1)\cdots(x_N - x_{N-1})} f_N \end{aligned} \quad \text{...IV.12}$$

Es importante señalar que si se repite algún par de valores de la función el método no funciona adecuadamente, por lo que se debe verificar que estos no se presente o de ser necesario omitirlos en los cálculos.

IV.III.II Ejemplo.[2]

Las densidades de sodio para tres temperaturas están dadas en la siguiente tabla:

i	$T_i\left[^oC\right]$	$\rho_i\left[kg/m^3\right]$
0	94	929
1	205	902
2	371	860

Tabla IV.4 Ejemplo Método de Lagrange.

Determinar la densidad para $T=251\left[^oC\right]$ utilizando la interpolación de Lagrange, escribir la formula de Lagrange que se ajusta a los tres datos.

Como el número de datos es tres el orden de la formula de Lagrange es $N=2$, entonces:

$$g(T)=\frac{(T-205)(T-371)}{(94-205)(94-371)}(929)+\frac{(T-94)(T-371)}{(205-94)(205-371)}(902)+\frac{(T-94)(T-205)}{(371-94)(371-205)}(860)$$

Sustituyendo $T=251\left[^oC\right]$ se obtiene:

$$\rho=890.5\left[kg/m^3\right]$$

IV.IV Método del spline cúbico.

Cuando un número grande de datos tiene que ajustarse a una curva suave, la interpolación de Lagrange no es adecuada.

Para esto se emplea el método del spline cúbico, este ajusta un polinomio cúbico en cada intervalo entre dos puntos consecutivos.

El spline cúbico es una técnica que ha cobrado mucha importancia, inicialmente se conocía como ajuste de datos con *curvígrafo cúbico*, este nombre se tomó de las plantillas de dibujo. Un curvígrafo es una plantilla flexible que se puede sostener por pesos, de manera que pase a través de cada uno de los puntos dados, pero conservando la lisura en cada intervalo de acuerdo con las leyes de la flexión del material. El procedimiento real matemático es una adaptación de esta idea.

IV.IV.I Algoritmo.[12]

Las condiciones para un ajuste con spline cúbico, son que se pase un conjunto de polinomios cúbicos a través de los puntos, utilizando un nuevo polinomio cúbico en cada intervalo. Para corresponder con la idea del curvígrafo, se requiere que tanto la pendiente como la curvatura, sean las mismas para el par de polinomios cúbicos que se unen en cada punto. El polinomio cúbico para el *i*-ésimo intervalo, el cual cae entre los puntos (x_i, y_i) y (x_{i+1}, y_{i+1}) en la forma:

$$y = a_i\left(x - x_i\right)^3 + b_i\left(x - x_i\right)^2 + c_i\left(x - x_i\right) + d_i \ldots \text{IV.13}$$

Puesto que esta se ajusta en los dos puntos extremos del intervalo:

$$y = a_i\left(x_i - x_i\right)^3 + b_i\left(x_i - x_i\right)^2 + c_i\left(x_i - x_i\right) + d_i = d_i \ldots \text{IV.14}$$

$$\begin{aligned} y &= a_i\left(x_{i+1} - x_i\right)^3 + b_i\left(x_{i+1} - x_i\right)^2 + c_i\left(x_{i+1} - x_i\right) + d_i \\ &= a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i \ldots \text{IV.15} \end{aligned}$$

En donde h_i es el Δx en el *i*-ésimo intervalo. Se necesitan las primeras y segundas derivadas para relacionar las pendientes y curvaturas de los polinomios que se unen, de manera que se deriva la ecuación IV.13:

$$y' = 3a_i\left(x - x_i\right)^2 + 2b_i\left(x - x_i\right) + c_i \ldots \text{IV.16}$$

$$y'' = 6a_i\left(x - x_i\right) + 2b_i \ldots \text{IV.17}$$

El procedimiento matemático se simplifica si se escriben las ecuaciones en términos de las segundas derivadas de los polinomios cúbicos interpolantes. Sea S_i la representación de la segunda derivada en el punto $\left(x_i, y_i\right)$ y S_{i+1} en el punto $\left(x_{i+1}, y_{i+1}\right)$. De la ecuación IV.17 se tiene:

$$\begin{aligned} S_i &= 6a_i\left(x_i - x_i\right) + 2b_i = 2b_i \\ S_{i+1} &= 6a_i\left(x_{i+1} - x_i\right) + 2b_i = 6a_i h_i + 2b_i \end{aligned}$$

Reescribiendo:

$$b_i = \frac{S_i}{2} \ldots \text{IV.18}$$

$$a_i = \frac{\left(S_{i+1} - S_i\right)}{6h_i} \ldots \text{IV.19}$$

Ahora sustituyendo a_i, b_i y d_i dadas por las ecuaciones IV.14, IV.18 y IV.19 en la ecuación IV.15:

$$y_{i+1} = \left(\frac{S_{i+1} - S_i}{6h_i}\right)h_i^3 + \frac{S_i}{2}h_i^2 + c_i h_i + y_i$$

Resolviendo para c_i:

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6} \ldots \text{IV.20}$$

Ahora es necesario definir la condición de que las pendientes de los dos polinomios cúbicos, que se unen en (x_i, y_i), sean las mismas. Para la ecuación en el *i*-ésimo intervalo, la ecuación IV.16 se hace, con $x = x_i$:

$$y' = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i = c_i$$

En el intervalo anterior, de x_{i-1} a x_i, la pendiente en su extremo derecho será:

$$y' = 3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1}$$
$$y' = 3a_{i-1}h_{i-1}{}^2 + 2b_{i-1}h_{i-1} + c_{i-1}$$

Igualando a estas y sustituyendo a, b, c, d en términos de S e y, se obtiene:

$$y_i' = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_iS_i + h_iS_{i+1}}{6}$$
$$y_i' = 3\left(\frac{S_i - S_{i-1}}{6h_{i-1}}\right)h_{i-1}^2 + 2\left(\frac{S_{i-1}}{2}\right)h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}} - 2\frac{h_{i-1}S_{i-1} + h_{i-1}S_i}{6}$$

Simplificando:

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_iS_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \ldots \text{IV.21}$$

La ecuación IV.21 se aplica a cada punto interno, de $i = 2$ hasta $i = n-1$; siendo n el número total de puntos. Esto da $n-2$ ecuaciones que relacionan a los n valores de S_i. Se obtienen dos ecuaciones adicionales involucrando a S_1 y S_n cuando se especifican las condiciones pertenecientes a los intervalos extremos de la curva.

Hasta cierto punto estas condiciones en los extremos son arbitrarias. Las tres elecciones alternativas que se suelen utilizar son:

1. $S_1 = 0, S_n = 0$. Esto equivale a suponer que los extremos de los polinomios cúbicos se tornan lineales.

2. $S_1 = S_2, S_n = S_{n-1}$. Esto equivale a suponer que los extremos de los polinomios cúbicos se aproximan a parábolas.

3. S_1 como una extrapolación lineal de S_2 y S_3, y S_n como una extrapolación lineal de S_{n-1} y S_{n-2}. Con esta suposición para un

conjunto de datos que se ajustan a una sola ecuación todos los curvígrafos cúbicos serán la misma cúbica. Las otras condiciones no tienen esta propiedad.

Las relaciones para la condición 3 en los extremos son:

- Extremo izquierdo:

$$\frac{S_2 - S_1}{h_1} = \frac{S_3 - S_2}{h_2},\ S_1 = \frac{(h_1 + h_2)S_2 - h_1 S_3}{h_2} \quad \text{...IV.22ª}$$

- Extremo derecho:

$$\frac{S_n - S_{n-1}}{h_{n-1}} = \frac{S_{n-1} - S_{n-2}}{h_{n-2}},\ S_n = \frac{(h_{n-2} + h_{n-1})S_{n-1} - h_{n-1}S_{n-2}}{h_{n-2}} \quad \text{...IV.22b}$$

Escribiendo las ecuaciones para $S_2, S_3, \ldots, S_{n-1}$ de la ecuación IV.21 en forma matricial se obtiene:

$$\begin{bmatrix} 2(h_1+h_2) & h_2 & & & & \\ h_1 & 2(h_2+h_3) & h_3 & & & \\ & h_2 & 2(h_3+h_4) & h_4 & & \\ & & h_3 & \ddots & \ddots & \\ & & & \ddots & \ddots & h_{n-1} \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ \vdots \\ S_{n-1} \\ S_n \end{bmatrix} = 6 \begin{bmatrix} \frac{Y_3 - Y_2}{h_2} - \frac{Y_2 - Y_1}{h_1} \\ \frac{Y_4 - Y_3}{h_3} - \frac{Y_3 - Y_2}{h_2} \\ \frac{Y_5 - Y_4}{h_4} - \frac{Y_4 - Y_3}{h_3} \\ \vdots \\ \frac{Y_n - Y_{n-1}}{h_{n-1}} - \frac{Y_{n-1} - Y_{n-2}}{h_{n-2}} \end{bmatrix} \quad \text{IV.23}$$

En el arreglo dado por IV.23 solo hay $n-2$ ecuaciones, pero se tienen n incógnitas. Se pueden eliminar dos incógnitas $(S_1 y S_n)$ utilizando las relaciones que correspondan a las condiciones en los extremos.

En cada caso se reduce el vector S a $n-2$ elementos, y la matriz de coeficientes se hace cuadrada, de tamaño $(n-2)*(n-2)$.

Además, la matriz se vuelve tridiagonal y, por tanto, se puede almacenar y resolver con rapidez.

Para cada condición en el extremo, la matriz de coeficientes se convierte en:

- Condición 1 $S_1 = 0, S_n = 0$:

$$\begin{bmatrix} 2(h_1+h_2) & h_2 & & & \\ h_2 & 2(h_2+h_3) & h_3 & & \\ & h_3 & 2(h_3+h_4) & \ddots & \\ & & \ddots & \ddots & h_{n-1} \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \quad \text{...IV.24}$$

- Condición 2 $S_1 = S_2, S_n = S_{n-1}$:

$$\begin{bmatrix} (3h_1+2h_2) & h_2 & & & \\ h_2 & 2(h_2+h_3) & h_3 & & \\ & h_3 & 2(h_3+h_4) & \ddots & \\ & & \ddots & \ddots & h_{n-1} \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \quad \text{...IV.25}$$

- Condición 3 S_1 y S_n son extrapolaciones lineales:

$$\begin{bmatrix} \frac{(h_1+h_2)(h_1+2h_2)}{h_2} & \frac{h_2^2-h_1^2}{h_2} & & & & \\ h_2 & 2(h_2+h_3) & h_3 & & & \\ & h_3 & 2(h_3+h_4) & \ddots & & \\ & & \ddots & \ddots & h_{n-1} & \\ & & & \frac{h_{n-2}^2-h_{n-1}^2}{h_{n-2}} & \frac{(h_{n-1}+h_{n-2})(h_{n-1}+2h_{n-2})}{h_{n-2}} \end{bmatrix} \quad \text{...IV.26}$$

Con la condición 3 después de resolver el conjunto de ecuaciones, se calcula S_1 y S_n utilizando las ecuaciones IV.22 a y b. Para cada caso el vector de los términos independientes es el mismo y esta dado por IV.23. Una vez que se han calculado los S_i valores, se obtienen los valores de a_i, b_i, c_i y d_i para cada intervalo si se desea interpolar para nuevos valores:

$$a_i = \frac{S_{i+1} - S_i}{6h_i}$$

$$b_i = \frac{S_i}{2} \qquad \text{...IV.27}$$

$$c_i = \frac{Y_{i+1} - Y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$

$$d_i = Y_i$$

IV.IV.II Ejemplo.[12]

Ajustar los datos de la siguiente tabla por medio del spline cúbico, haciendo uso de las tres condiciones de los extremos. La función que proporciona los valores de dicha tabla es $y = x^3 - 8$.

x	Y
0	-8
1	-7
2	0
3	19
4	56

Tabla IV.5 Ejemplo Spline Cúbico.

Para la condición 1 en el extremo:

$$\begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix}$$

Resolviendo:

$$S_1 = 0.00, S_2 = 6.4285, S_3 = 10.2857, S_4 = 24.4285, S_5 = 0.00$$

Para la condición 2 en el extremo:

$$\begin{bmatrix} 5 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 5 \end{bmatrix} \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix}$$

Resolviendo:

$$S_1 = 4.80, S_2 = 4.80, S_3 = 12.00, S_4 = 19.20, S_5 = 19.20$$

Para la condición 3 en el extremo:

$$\begin{bmatrix} 6 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 6 \end{bmatrix} \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix}$$

Resolviendo:

$$S_1 = 0.00, S_2 = 6.00, S_3 = 12.00, S_4 = 18.00, S_5 = 24.00$$

La ecuación de los datos $y = x^3 - 8$, da valores para las segundas derivadas iguales a los que se obtienen al utilizar la condición 3.

IV.V. Mínimos Cuadrados.

Otro método utilizado para ajustar datos a una curva para así poder interpolar datos no conocidos es el de mínimos cuadrados.

Este se puede utilizar haciendo el ajuste de una línea recta a una serie de datos, este procedimiento es comúnmente conocido como regresión lineal, cuando los datos no se ajustan a una línea recta se puede utilizar una curva no lineal.

IV.V.I Regresión lineal.

Se desea encontrar una función lineal que se ajuste a una serie de datos (x, y) con una desviación mínima, esta función lineal se conoce como recta de regresión.

IV.V.I.I Algoritmo.

La función lineal se expresa como:

$$g(x) = bx + a \ldots \text{IV.28}$$

Donde a y b son constantes por determinar. La desviación de la recta con respecto a cada dato se define como.

$$r_i = y_i - g(x_i) = y_i - (bx_i + a),\ i = 1,2,\ldots,n \ldots \text{IV.29}$$

Donde n es el número total de datos a y b son las constantes a determinar. Así, el cuadrado total de las desviaciones está dado por:

$$R = \sum_{i=1}^{n} (r_i)^2 = \sum_{i=1}^{n} (y_i - a - bx_i)^2 \ \ldots \text{IV.30}$$

Debido a que a y b son parámetros arbitrarios, se determinan de forma que minimicen a R. El mínimo de R se obtiene si las derivadas parciales de R con respecta de a y b se anulan:

$$\frac{\partial R}{\partial a} = -2\sum_{i=1}^{n}\left(y_i - bx_i - a\right)$$
$$\frac{\partial R}{\partial b} = -2\sum_{i=1}^{n}\left(y_i - bx_i - a\right) \quad \ldots\text{IV.30}$$

Dividiendo entre -2, el sistema IV.30 se puede reescribir como:

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \ldots\text{IV.31}$$

En donde:

$$\begin{aligned} A_{1,1} &= n \\ A_{1,2} &= \sum x_i \\ Z_1 &= \sum y_i \\ A_{2,1} &= \sum x_i \\ A_{2,2} &= \sum \left(x_i\right)^2 \\ Z_2 &= \sum x_i y_i \end{aligned} \quad , \; i = 1,2,\ldots,n$$

Este sistema se puede resolver utilizando alguno de los métodos vistos en el capítulo II, o directamente a partir de las siguientes ecuaciones:

$$a = \frac{A_{2,2}Z_1 - A_{1,2}Z_2}{d} \ldots\text{IV.32}$$
$$b = \frac{A_{1,1}Z_2 - A_{2,1}Z_1}{d} \ldots\text{IV.33}$$
$$d = A_{1,1}A_{2,2} - A_{1,2}A_{2,1} \ldots\text{IV.34}$$

IV.V.I.II Ejemplo.[2]

Calcular la línea de regresión para los datos de la siguiente tabla:

i	x	y
1	0.1	0.61
2	0.4	0.92
3	0.5	0.99

4	0.7	1.52
5	0.7	1.47
6	0.9	2.03

Tabla IV.6 Ejemplo Regresión Lineal.

Graficando estos puntos se tiene la siguiente gráfica:

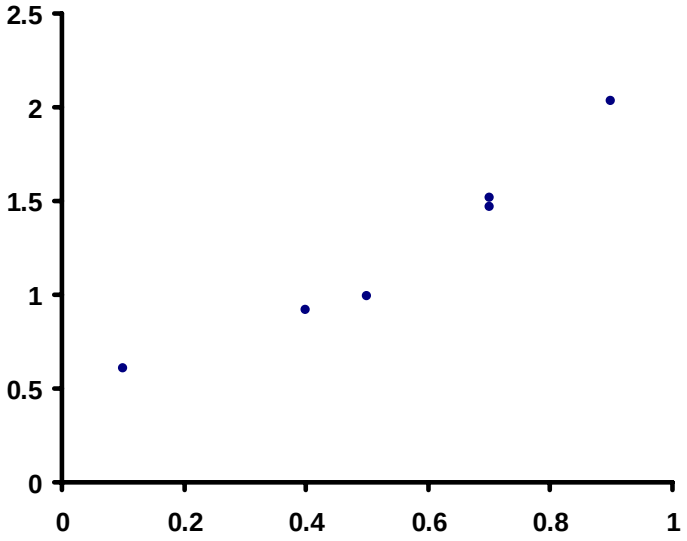


Figura IV.5 Gráfica Ejemplo de Regresión Lineal.

Calculando los valores de A para la matriz de coeficientes y de Z del vector independiente se tiene:

Dato	$A_{1,2}, A_{2,1}$	Z_1	$A_{2,2}$	Z_2
i	x	y	x_i^2	$x_i y_i$
1	0.1	0.61	0.01	0.061
2	0.4	0.92	0.16	0.368
3	0.5	0.99	0.25	0.495
4	0.7	1.52	0.49	1.064
5	0.7	1.47	0.49	1.029
n=6	0.9	2.03	0.81	1.827
Suma:	3.3	7.54	2.21	4.844

Tabla IV.7 Valores A y Z de Regresión Lineal.

Entonces el sistema es:

$$\begin{bmatrix} 6 & 3.3 \\ 3.3 & 2.21 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 7.54 \\ 4.844 \end{bmatrix}$$

Resolviendo este sistema se tienen los valores de a y b con los que se puede escribir la función que representa la recta que se ajusta a estos puntos:

$$g(x) = \underbrace{1.7645}_{b}x + \underbrace{0.2862}_{a}$$

Haciendo uso de esta ecuación se grafica la recta de ajuste junto con los puntos de la tabla:

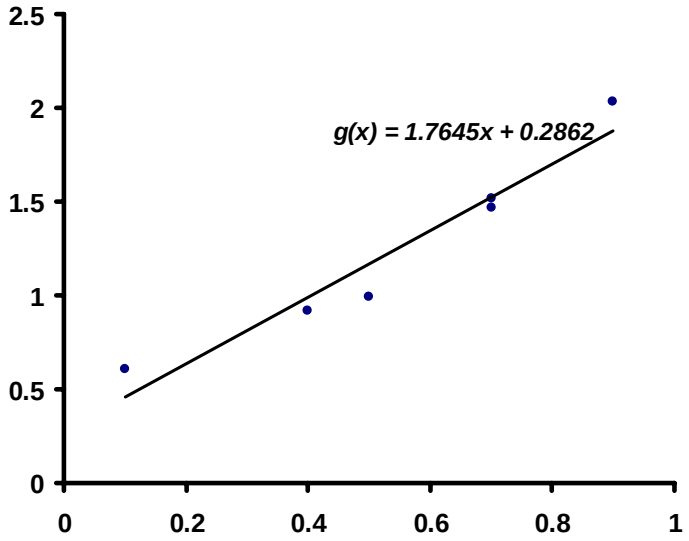


Figura IV.6 Recta de Ajuste Regresión Lineal.

IV.V.II Ajuste de curvas no lineales por mínimos cuadrados.

En muchos casos los datos con los que se cuenta no se ajustan de manera correcta a una función lineal, es por esto que se vuelve necesario ajustar alguna otra función no lineal a dichos datos.

Para esto, el método de mínimos cuadrados se puede extender para ajustar un polinomio de cualquier orden a los datos.

IV.V.II.I Algoritmo.

Considerando un polinomio $g(x)$ de orden N :

$$g(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N$$

La desviación de la curva de los puntos dados es:

$$r_i = y_i - g(x_i),\ i = 1,2,\ldots,L \quad \ldots\text{IV.35}$$

Donde L es el número de datos, el total de los cuadrados de la desviación es el siguiente:

$$R = \sum_{i=1}^{n} (r_i)^2 \quad \text{...IV.36}$$

Entonces, igualando a cero las derivadas parciales de R con respecto a los coeficientes del polinomio buscando minimizar a R:

$$\frac{\partial R}{\partial a_n} = 0\,,\ n = 0,1,2,\ldots,N \quad \text{...IV.37}$$

Esto equivale a:

$$\sum_{n=0}^{N}\left[\sum_{i=1}^{L} x_i^{n+k}\right] a_n = \sum_{i=1}^{L} x_i^k y_i\,,\ k = 0,1,2,\ldots,N \quad \text{...IV.38}$$

Que en forma matricial se escribe:

$$\begin{bmatrix} L & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^N \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{N+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum x_i^n & \sum x_i^{N+1} & \sum x_i^{N+2} & \cdots & \sum x_i^{2N} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^N y_i \end{bmatrix} \quad \text{...IV.39}$$

Este sistema se puede resolver utilizando alguno de los métodos vistos en el capítulo II.

El grado N del polinomio no es fijo indistintamente se puede utilizar un polinomio de grado dos o uno de grado seis, sin embargo, de esto dependerá el correcto ajuste y una buena interpolación de los datos deseados.

Para esto cabe señalar que desde el punto de vista experimental rara vez se necesitan polinomios más complejos que uno de cuarto grado, y cuando se necesitan, el problema con frecuencia es manejado ajustando una serie de polinomios a un subconjunto de datos.[12]

IV.V.II.II Ejemplo.[2]

Ajustar un polinomio cuadrático para los datos de la siguiente tabla:

i	X	y
1	0.1	0.61
2	0.4	0.92
3	0.5	0.99

4	0.7	1.52
5	0.7	1.47
6	0.9	2.03

Tabla IV.8 Ejemplo Mínimos Cuadrados.

Graficando estos puntos se tiene:

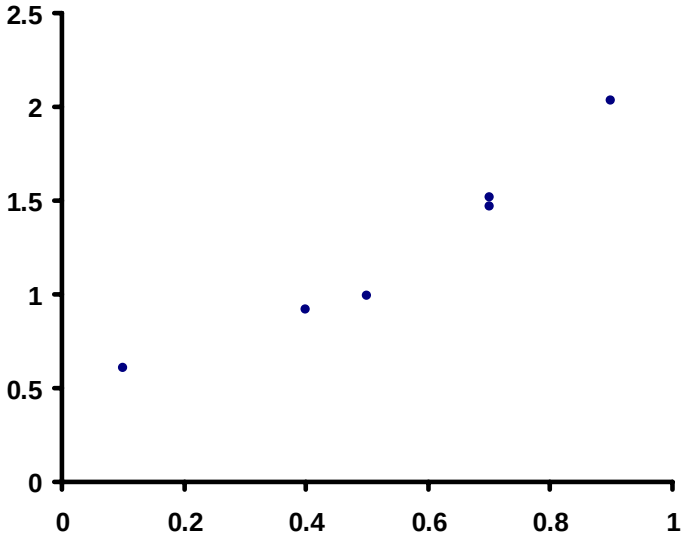


Figura IV.7 Gráfica Ejemplo Mínimos Cuadrados.

Calculando los elementos de la matriz de coeficientes así como los del vector independiente:

Dato i	x	y	x^2	x^3	x^4	$x_i y_i$	$x_i^2 y_i$
1	0.1	0.61	0.01	0.001	0.0001	0.061	0.0061
2	0.4	0.92	0.16	0.064	0.0256	0.368	0.1472
3	0.5	0.99	0.25	0.125	0.0625	0.495	0.2475
4	0.7	1.52	0.49	0.343	0.2401	1.064	0.7448
5	0.7	1.47	0.49	0.343	0.2401	1.029	0.7203
6	0.9	2.03	0.81	0.729	0.6561	1.827	1.6443
Suma :	3.3	7.54	2.21	1.605	1.2245	4.844	3.5102

Tabla IV.9 Elementos Matriz y Vector para Mínimos Cuadrados.

Entonces el sistema es:

$$\begin{bmatrix} 6.0000 & 3.3000 & 2.2100 \\ 3.3000 & 2.2100 & 1.6050 \\ 2.2100 & 1.6050 & 1.2245 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 7.5400 \\ 4.8440 \\ 3.5102 \end{bmatrix}$$

Resolviendo este sistema se encuentran los coeficientes de las potencias:

Potencia n	*Coeficiente* a_n
0	0.587114
1	0.059102
2	1.729537

Tabla IV.10 Coeficientes de Potencias.

Por lo que el polinomio cuadrático determinado es:

$$g(x) = 0.587114 + 0.059102x + 1.729537x^2$$

Haciendo uso de esta ecuación se grafica el polinomio de ajuste junto con los puntos de la tabla:

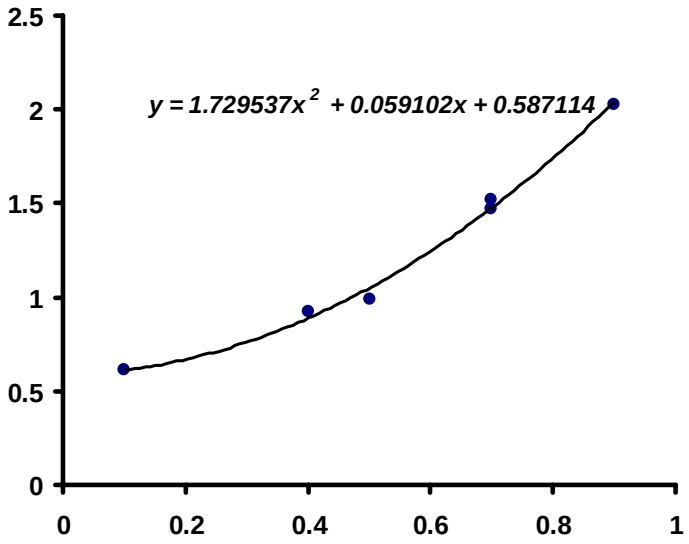


Figura IV.8 Polinomio de Ajuste Mínimos Cuadrados.

V. DERIVACIÓN E INTEGRACIÓN NUMÉRICA.

V.I Introducción.

Con frecuencia es necesario derivar o integrar una función, sin embargo puede ocurrir que la función en términos de x no se conozca y solo se tenga una tabulación de valores que definan la función. En el caso de la integración de una función, aun cuando se conoce la función, no es fácil programar a las computadoras para obtener dicha integral de manera analítica. Para esto se utilizan métodos numéricos de derivación e integración.

V.II Derivación numérica.

Una gran cantidad de fenómenos físicos y de problemas de ingeniería están descritos mediante ecuaciones diferenciales ordinarias o parciales, de ahí la importancia de la derivación numérica.

V.II.I Derivadas de datos.

En general se pueden obtener aproximaciones numéricas de la derivada de una función en un punto derivando alguno de los polinomios interpolantes, por ejemplo el de Lagrange, esto cuando solo se conoce una tabla de datos de dicha función.

V.II.I.I Algoritmo.

Suponiendo que $x_0, x_1, \ldots, x_n$ son $n+1$ puntos en un intervalo I, y que $f(x) \in C^{n+1}(I)$, en donde $C^{n+1}(I)$ representa un conjunto de constantes dentro del intervalo I. Si $P(x) = \sum_{k=0}^{N} f(x_k) L_k(x)$ y considerando el polinomio interpolador de Lagrange se tiene:

$$f(x) = P(x) + \frac{(x - x_0)\ldots(x - x_N)}{(n+1)!} f^{N+1}(\varepsilon(x)) \ldots \text{V.1}$$

Para alguna $\varepsilon(x) \in I$. Derivando esta expresión y evaluando en algún punto x_j (de los $N+1$ puntos x_i) se obtienen:

$$f'(x_j) = \sum_{k=0}^{n} f(x_k) L_k'(x) + \frac{f^{n+1}(\varepsilon(x_j))}{(n+1)!} \prod_{k=0;k\neq j}^{n} (x_j - x_k) \ldots \text{V.2}$$

Esta ecuación se conoce como *fórmula de los* $n+1$ *puntos* para aproximar $f'(x_j)$, esta resulta ser muy compleja por lo que se definen fórmulas para tres y cinco puntos. Suponiendo que los x_i son espaciados con un $\Delta x = cte$, se tiene que $x_1 = x_0 + \Delta x, x_2 = x_0 + 2\Delta x, \ldots, x_n = x_0 + n\Delta x$ con $\Delta x \neq 0$. Con tres datos $x_j = x_0, x_1, x_2$ igualmente espaciados y utilizando la expresión V.2 se obtiene la *fórmula para tres puntos*:

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2} f(x_0) + 2 f(x_1) - \frac{1}{2} f(x_2)\right] \ldots \text{V.3}$$

$$f'(x_1) = \frac{1}{h}\left[-\frac{1}{2} f(x_0) + \frac{1}{2} f(x_2)\right] \ldots \text{V.4}$$

$$f'(x_2) = \frac{1}{h}\left[\frac{1}{2} f(x_0) - 2 f(x_1) + \frac{3}{2} f(x_2)\right] \ldots \text{V.5}$$

Ahora con cinco datos $x_j = x_0, x_1, x_2, x_3, x_4$ igualmente espaciados y utilizando la expresión V.2 se obtiene la *fórmula para cinco puntos*:

$$f'(x_2) = \frac{1}{12h}\left[f(x_0) - 8 f(x_1) + 8 f(x_3) - f(x_4)\right] \ldots \text{V.6}$$

V.II.I.II Ejemplo.

A partir de la siguiente tabla calcular la derivada $f'(0.05)$:

x	$f(x) = e^x$
0.00	1.000000000
0.01	1.010050167
0.02	1.020201340
0.03	1.030454534
0.04	1.040810774
0.05	1.051271096
0.06	1.061836547
0.07	1.072508181
0.08	1.083287068
0.09	1.094174284

Tabla V.1 Ejemplo Derivada de Datos.

Para calcular $f'(0.05)$ utilizando la ecuación V.6 se tiene:

$$x_0 = 0.03, x_1 = 0.04, x_2 = 0.05, x_3 = 0.06, x_4 = 0.07$$

Como $\Delta x = 0.01$ se tiene:

$$f'(0.05) \approx \frac{1}{12 * 0.01}[f(0.03) - 8f(0.04) + 8f(0.06) - f(0.07)] = 1.0512710960256$$

V.II.III Método con serie de Taylor.

La aproximación por diferencias son importantes en la solución de ecuaciones diferenciales ordinarias y parciales, pues se utiliza para evaluar las derivadas de una función por medio de los valores dados en algunos puntos.

V.II.III.I Algoritmo.

Para una derivada de orden n, el número mínimo de datos que se necesita para aproximar la derivada por medio de diferencias finitas es $n+1$.
La notación que se utilizará en el desarrollo de la aproximación por diferencias será:

$$\begin{aligned} f_{i+1} &= f(x_{i+1}) \\ f_i &= f(x_i) \\ f_{i-1} &= f(x_{i-1}) \\ x_{i+1} &= x_i + \Delta x \\ x_{i-1} &= x_i - \Delta x \\ f_i' &= f'(x_i) \end{aligned}$$

Los valores de f en todos los puntos distintos de x_i se desarrollan en una serie de Taylor. El desarrollo de Taylor de f_{i+1} alrededor de x_i es:

$$f_{i+1} = f_i + \frac{\Delta x}{1!} f_i' + \frac{\Delta x^2}{2!} f_i'' + \frac{\Delta x^3}{3!} f_i''' + \cdots + \frac{\Delta x^n}{n!} f_i^n \ldots \text{V.7}$$

Entonces para la primera derivada despejando f_i' se tiene:

$$f_i' = \frac{f_{i+1} - f_i}{\Delta x} - \frac{1}{2!}\Delta x f_i'' - \frac{1}{3!}\Delta x^2 f_i''' - \cdots - \frac{1}{n!}\Delta x^{n-1} f_i^n \ldots \text{V.8}$$

Si después del primer término se trunca la serie se tiene la ecuación que permite obtener la aproximación por diferencias hacia adelante, comúnmente conocidas como diferencias progresivas:

$$f_i^{'} = \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x) \ldots \text{V.9}$$

Donde $O(\Delta x) = -\frac{1}{2}\Delta x f_i^{''}$ y representa el error debido al truncamiento de la serie, este es aproximadamente proporcional al intervalo Δx. Gráficamente la aproximación progresiva se muestra en la figura V.1.

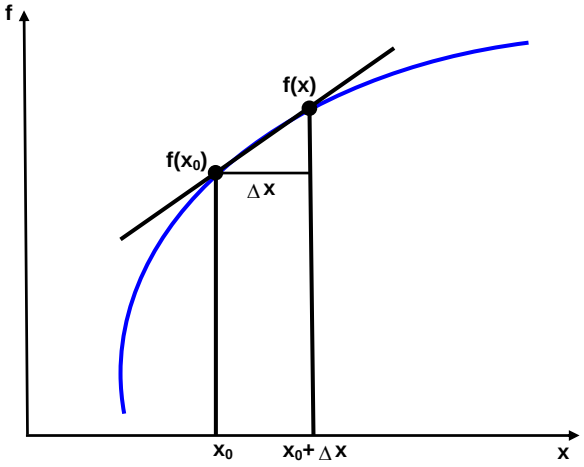


Figura V.1 Aproximación Progresiva.

El desarrollo de Taylor de f_{i-1} alrededor de x_i es:

$$f_{i-1} = f_i - \frac{\Delta x}{1!} f_i^{'} + \frac{\Delta x^2}{2!} f_i^{''} - \frac{\Delta x^3}{3!} f_i^{'''} + \cdots + \frac{\Delta x^n}{n!} f_i^{n} \ldots \text{V.10}$$

Despejando $f_i^{'}$ y truncando la serie hasta el primer término se tiene la expresión para aproximar la derivada por medio de diferencias hacia atrás o regresivas:

$$f_i^{'} = \frac{f_i - f_{i-1}}{\Delta x} + O(\Delta x) \ldots \text{V.11}$$

Donde $O(\Delta x) = \frac{1}{2}\Delta x f_i^{''}$ y representa el error debido al truncamiento de la serie. Gráficamente la aproximación regresiva se muestra en la figura V.2.

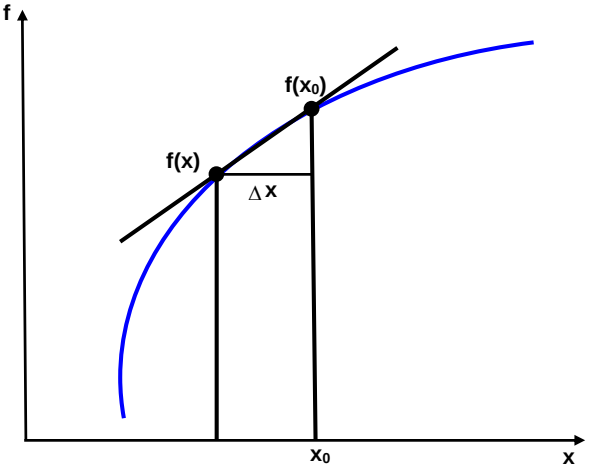


Figura V.2 Aproximación Regresiva.

Utilizando los desarrollos de Taylor para f_{i+1} y f_{i-1} alrededor de x_i se obtiene la tercer expresión para aproximar una derivada por medio de diferencias centrales, restando el desarrollo de f_{i-1} al desarrollo de f_{i+1} se tiene:

$$f_{i+1} - f_{i-1} = 2\Delta x f_i^{'} + \frac{\Delta x^3}{3!} f_i^{'''} + \cdots + \frac{\Delta x^n}{n!} f_i^{n} \ldots \text{V.12}$$

Despejando $f_i^{'}$ y truncando la serie hasta el primer término se tiene la expresión para aproximar la derivada por medio de diferencias centrales:

$$f_i^{'} = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O\left(\Delta x^2\right) \ldots \text{V.13}$$

Donde $O\left(\Delta x^2\right) = -\frac{1}{6}\Delta x^2 f_i^{'''}$ y representa el error debido al truncamiento de la serie, debido a la cancelación del término $f^{''}$ el error de la aproximación por diferencias centrales es proporcional a Δx^2 en vez de Δx. Al decrecer Δx, el error decrece más rápido que en las otras dos aproximaciones, por lo que las diferencias centrales representan la mejor aproximación. Gráficamente la aproximación central se muestra en la figura V.3.

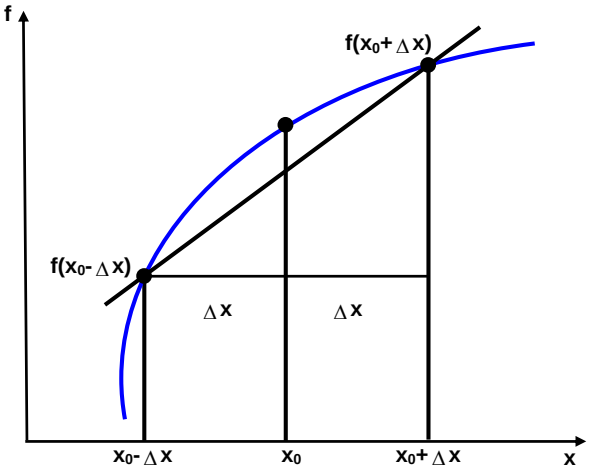


Figura V.3 Aproximación Central.

V.II.IV Derivada con spline cúbico.[12]

Además de su uso como método de interpolación, el spline cúbico puede utilizarse para encontrar derivadas e integrales.

V.II.IV.I Algoritmo.

Como el spline cúbico se aproxima a $f(x)$ como un polinomio cúbico, se puede escribir para el intervalo $x_i \le x \le x_{i+1}$:

$$f(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \ldots \text{V..14}$$

Donde los coeficientes se calculan según las ecuaciones IV.14, IV.18, IV.19 y IV.20 del capítulo IV.

$$a_i = \frac{(S_{i+1} - S_i)}{6(x_{i+1} - x_i)} \ldots \text{V.15}$$

$$b_i = \frac{S_i}{2} \ldots \text{V.16}$$

$$c_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} - \frac{2(x_{i+1} - x_i)S_i + (x_{i+1} - x_i)S_{i+1}}{6} \ldots \text{V.17}$$

$$d_i = f(x_i) \ldots \text{V.18}$$

Por lo que la aproximación de las primeras y segundas derivadas es directa; se estiman estos como los valores de las derivadas del polinomio cúbico:

$$f(x)' \overset{\bullet}{=} 3a_i(x-x_i)^2 + 2b_i(x-x_i) + c_i \ldots \text{V.19}$$

$$f(x)'' \overset{\bullet}{=} 6a_i(x-x_i) + 2b_i \ldots \text{V.20}$$

En los $n+1$ puntos x_i donde se conoce la función y el spline cúbico coincide con $f(x)$, estas fórmulas son particularmente sencillas:

$$f(x)' \overset{\bullet}{=} c_i \ldots \text{V.21}$$

$$f(x)'' \overset{\bullet}{=} 2b_i \ldots \text{V.21}$$

El spline cúbico no es útil para aproximar derivadas de un orden más elevado que el segundo, pues seria necesario un grado mas elevado de la función dada por el spline.

V.II.IV.II Ejemplo.[12]

Calcular la derivada de $f(x) = sen\pi x$ sobre el intervalo $0 \le x \le 1$ utilizando un spline cúbico que se ajusta en $x = 0, 0.25, 0.5, 0.75$ y 1.0:

i	x	$f(x)$
1	0.00	0.0000
2	0.25	0.7071
3	0.50	1.0000
4	0.75	0.7071
5	1.00	0.0000

Tabla V.2 Ejemplo Derivada con Spline Cúbico.

Utilizando la condición 1 de los extremos $S_1 = 0$ y $S_5 = 0$, en la siguiente tabla se muestran los resultados obtenidos por medio del spline cúbico:

i	x	S_i	a_i	b_i	c_i	d_i
1	0.00	0.0000	-4.8960	0.0000	3.1340	0.0000
2	0.25	-7.3440	-2.0288	-3.6720	2.2164	0.7071
3	0.50	-10.3872	2.0288	-5.1936	0.0000	1.0000
4	0.75	-7.3440	4.8960	-3.6720	-2.2164	0.7071

Tabla V.3 Resultados Derivada con Spline Cúbico.

Con estos datos se pueden calcular las derivadas, obteniendo:

x	$f'(x)$	*Exacta*	*Error*	$f''(x)$	*Exacta*	*Error*
0.25	2.2164	2.2214	0.005	-7.344	-6.979	0.365
0.50	0.0000	0.0000	0.000	-10.387	-9.870	0.517
0.75	-2.2164	-2.2214	0.005	-7.344	-6.979	0.365

Tabla V.4 Derivadas y Error.

V.III Integración numérica.

Los métodos de integración numérica se utilizan para integrar funciones, ya sea de forma analítica o por medio de una tabla dada. Estos métodos se obtienen al integrar los polinomios de interpolación, como resultado de esto las distintas fórmulas de interpolación darán diferentes métodos de integración.

V.III.I Fórmulas de Newton – Cotes.

Se basan en las fórmulas de interpolación con puntos de separación uniforme y se deducen al integrar las fórmulas de interpolación de Newton así como la de interpolación de Lagrange. Y se subdividen en: tipo cerrado y tipo abierto.

V.III.I.I Regla del trapecio.

Este método se obtiene de integrar la fórmula de interpolación lineal vista en la sección IV.II.II.

V.III.I.I.I Algoritmo.

Dicha integral que da como:

$$I = \int_a^b f(x)dx = \frac{b-a}{2}\left[f(a) + f(b)\right] + E \quad \ldots \text{V.22}$$

Donde E representa el error. En la figura se puede ver la regla del trapecio de forma gráfica:

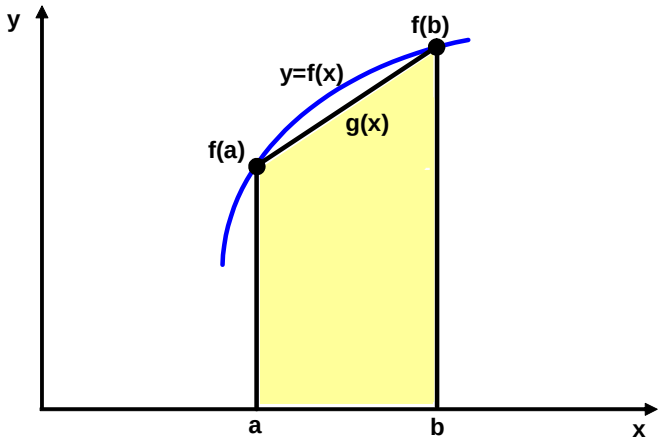


Figura V.4 Regla del Trapecio.

La recta $g(x)$ es la recta de interpolación, mientras que el área sombreada por debajo de esta es el resultado de la integral obtenida con la regla del trapecio, como el área por debajo de la curva $f(x)$ representa la integral exacta el error esta dado por el área entre $g(x)$ y $f(x)$.

La regla se puede extender a varios intervalos y se puede aplicar N veces al caso de N intervalos con una separación uniforme h para así obtener la regla extendida del trapecio:

$$I = \int_a^b f(x) = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{N-1} f(a+jh) + f(b) \right] + E \quad \text{...V.23}$$

Donde $h = {(b-a)}/{N}$. La ecuación anterior se puede escribir en la siguiente forma:

$$I = \frac{h}{2}\left(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{N-1} + f_N \right) + E \quad \text{...V.24}$$

En donde $f_0 = f(a)$, $f_1 = f(a+h)$ y $f_i = f(a+ih)$.

V.III.I.I.II Ejemplo.[2]

El cuerpo de revolución mostrado en la figura se obtiene después de girar la curva $y = 1 + \left({x}/{2}\right)^2$ en el intervalo $0 \le x \le 2$ alrededor del eje x. Calcular el volumen utilizando la regla extendida del trapecio con $N = 2$ y 4. Considerar que el valor exacto es 11.7286 u^3.

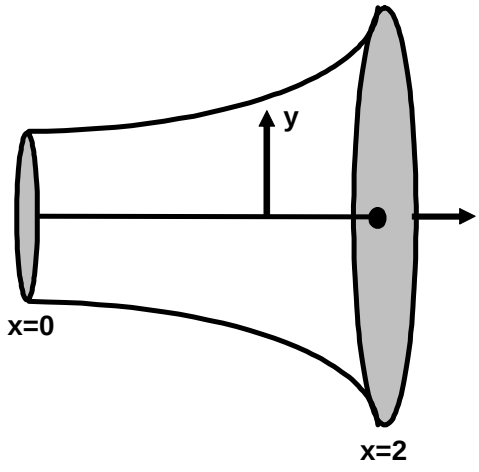


Figura V.5 Ejemplo Regla del Trapecio.

El volumen está dado por $I = \int_0^2 f(x)dx = \int_0^2 \pi\left(1+\left(\frac{x}{2}\right)^2\right)^2 dx$

Para $N = 2$ $h = \frac{(2-0)}{2} = 1$

$$I = \frac{1}{2}[f(0)+2f(1)+f(2)] = 0.5\pi[1+2(1.5625)+4] = 12.7627u^3$$

Para $N = 4$ $h = \frac{(2-0)}{4} = 0.5$

$$I = \frac{0.5}{2}[f(0)+2f(0.5)+2f(1.5)+f(2)] = 11.9895u^3$$

Se observa que al aumentar el valor de N, el cálculo comienza a aproximarse al valor de la integral analítica, con $N = 128$ el resultado obtenido es $11.7288u^3$.

V.III.I.II Regla de Simpson 1/3.

Esta regla es para una cuadrática integrada en dos intervalos Δx de ancho uniforme.

V.III.I.II.I Algoritmo.

La fórmula es:

$$\int_a^b f(x)dx = \frac{h}{3}[f(a)+4f(\bar{x})+f(b)]+E \ldots \text{V.25}$$

Donde $h = (b-a)/2$ y $\bar{x} = (a+b)/2$, el error se anula si $f(x)$ es un polinomio de orden menor o igual que 3.

La regla extendida de Simpson 1/3 se aplica a un dominio dividido en un número par de intervalos n y se escribe:

$$I = \frac{h}{3}\left[f(a)+4\sum_{i=1}^{n-1} f(a+ih)+2\sum_{i=2}^{n-2} f(a+ih)+f(b)\right]+E \ldots \text{V.26}$$

Donde $h = (b-a)/N$; y la primera suma es únicamente sobre las i impares y la segunda es sobre las i pares, esta ecuación se puede escribir también como:

$$I = \frac{h}{3}[f_0+4f_1+2f_2+4f_3+2f_4+\cdots+2f_{n-2}+4f_{n-1}+f_n]+E \ldots \text{V.27}$$

V.III.I.II.II Ejemplo.[2]

Calcular el volumen del ejemplo V.III.I.II.II utilizando la regla extendida de Simpson 1/3 con $N = 2$ y 4. Considerar que el valor exacto es 11.7286 u^3.

El volumen está dado por $I = \int_0^2 f(x)dx = \int_0^2 \pi\left(1+\left(\frac{x}{2}\right)^2\right)^2 dx$

Para $N = 2$ $h = \frac{(2-0)}{2} = 1$

$$I = \frac{1}{3}\left[f(0)+4f(1)+f(2)\right] = \frac{1}{3}\pi\left[1+4\left(1.25^2\right)+2^2\right] = 11.7809u^3$$

Para $N = 4$ $h = \frac{(2-0)}{4} = 0.5$

$$I = \frac{0.5}{3}\left[f(0)+4f(0.5)+2f(1)+4f(1.5)+f(2)\right] = 11.7318u^3$$

Se observa que al aumentar el valor de N, el cálculo comienza a aproximarse al valor de la integral analítica, lo hace con mayor rapidez que la regla del trapecio, con $N = 16$ el resultado obtenido es exacto $11.7286u^3$.

V.III.I.III Regla de Simpson 3/8.

La regla de Simpson 3/8 se obtiene de integrar un polinomio de interpolación cúbico.

V.III.I.III.I Algoritmo.

Para un dominio $[a,b]$ dividido en tres intervalos se tiene:

$$I = \int_a^b f(x)dx = \frac{3}{8}h\left[f_0+3f_1+3f_2+f_3\right]+E \quad \text{...V.28}$$

Donde $h = {}^{(b-a)}\!/_{3}$, $f_i = f(a+ih)$.

V.III.II Cuadratura Gaussiana.

Los métodos de cuadratura Gaussiana son de integración numérica que utilizan puntos obtenidos de las raíces de los polinomios de Legendre (anexo A4). Estas cuadraturas de Gauss no se pueden utilizar para integrar una función dada en forma de tabla con intervalos de separación uniforme debido a que los puntos de Legendre no están separados de esa manera; aun con esta limitante son mas apropiadas para integrar funciones analíticas.

Las cuadraturas de Gauss difieren de gran manera de las fórmulas de Newton – Cotes debido a que los N puntos de la retícula, como ya se mencionó, se obtienen de las raices de los polinomios de Legendre $P_n(x)=0$.

V.III.II.I Algoritmo.[2]

La cuadratura de Gauss que se extiende sobre el intervalo $[-1,1]$ esta dada por:

$$\int_{-1}^{1} f(x)dx = \sum_{k=1}^{n} w_k f(x_k) \ldots \text{V.29}$$

Donde n es el número de puntos de Gauss, los w_i se conocen como pesos y x_i son los puntos de Gauss, algunos de estos valores se muestran en la siguiente tabla:

n	$\pm x_i$	w_i
2	0.577350269	1.000000000
3	0.000000000	0.888888889
	0.774596669	0.555555556
4	0.339981043	0.652145155
	0.861136312	0.347854845
5	0.000000000	0.568888889
	0.538469310	0.478628670
	0.906179846	0.236926885
6	0.238619186	0.467913935
	0.661209387	0.360761573
	0.932469514	0.171324492
8	0.183434642	0.362683783
	0.525532410	0.313706646
	0.796666478	0.222381034
	0.960289857	0.101228536
10	0.148874339	0.295524225
	0.433395394	0.269266719
	0.679409568	0.279086363
	0.865063367	0.149451349
	0.973906528	0.066671344

Tabla V.5 Puntos de Gauss.

Los valores x de los puntos de Gauss son pares debido a los signos $\pm$. Esta fórmula de integración puede aplicarse a cualquier intervalo arbitrario $[a,b]$ siguiendo la siguiente relación:

$$x = \frac{2z - a - b}{b - a} \ldots \text{V.30}$$

Donde z es la coordenada original en $a < z < b$ y x es la coordenada normalizada en $-1 \le x \le 1$. Por lo que la transformación de x en z es:

$$z = \frac{(b-a)x + a + b}{2} \ldots \text{V.31}$$

Entonces la integral se puede escribir:

$$\int_a^b f(z)dz = \int_{-1}^{1} f(z)\left(\frac{dz}{dx}\right)dx = \frac{b-a}{2}\sum_{k=1}^{n} w_k f(z_k) \ldots \text{V.32}$$

Donde $dz/dx = (b-a)/2$. Los valores de z_k se obtienen al sustituir x en la ecuación V.31 por los puntos x_i de los puntos de Gauss de la tabla V.4.

Si $f(x)$ es un polinomio de orden menor a $2n-1$ la cuadratura de Gauss de orden n es exacta.

V.III.II.II Ejemplo.

Integrar la función $f(x) = 6x^2 - 2.6x + 9.7$ en el intervalo $[1,4]$:

$$I = \int_1^4 f(x)dx = \int_1^4 (6x^2 - 2.6x + 9.7)dx$$

Considerando el orden de la cuadratura $n = 2$, se tiene $2(2)-1 = 3$, como $f(x)$ es un polinomio de grado 2 entonces la cuadratura será exacta, de la tabla V.4 se obtienen los valores x_i correspondientes a una cuadratura de orden 2 para calcular los z_i:

$$z_1 = \frac{(4-1)(0.57735)+1+4}{2} = 3.366025$$
$$z_2 = \frac{(4-1)(-0.57735)+1+4}{2} = 1.633975$$

Entonces la cuadratura queda definida como:

$$\int_1^4 f(z)dz = \int_{-1}^{1} f(z)(dz/dx)dx = 1.5[(1)f(3.366025) + (1)f(1.633975)] = 135.6u^3$$

V.III.III Integración con Spline Cúbico.[12]

Para aproximar la integral de $f(x)$ sobre los n intervalos en donde $f(x)$ se ajusta mediante el spline cúbico se hace de manera directa.

V.III.III.I Algoritmo.

La integral se obtiene mediante la siguiente expresión:

$$\int_{x_1}^{x_{n+1}} f(x)dx = \int_{x_1}^{x_{n+1}} P_3(x)dx = \sum_{i=1}^{n} \frac{a_i}{4}(x-x_i)^4 + \frac{b_i}{3}(x-x_i)^3 + \frac{c_i}{2}(x-x_i)^2 + d_i(x-x_i)\Big|_{x_i}^{x_{i+1}} \quad \ldots \text{V.33}$$

Que evaluada en los límites de integración se reduce a:

$$\sum_{i=1}^{n}\left[\frac{a_i}{4}(x_{i+1}-x_i)^4 + \frac{b_i}{3}(x_{i+1}-x_i)^3 + \frac{c_i}{2}(x_{i+1}-x_i)^2 + d_i(x_{i+1}-x_i)\right] \ldots \text{V.34}$$

V.III.III.II Ejemplo.[12]

Continuando con el ejemplo de la sección V.II.III la integral es:

$$\int_0^1 f(x)dx = \frac{(0.25)^4}{4}(0) + \frac{(0.25)3}{3}(-12.5376) + \frac{(0.25)^2}{2}(3.1340) + 0.25(2.4142) = 0.6362$$

El valor exacto es 0.6366 por lo que se tiene un error de 0.0004.

VI. ECUACIONES DIFERENCIALES ORDINARIAS.

VI.I Introducción.

Los problemas que involucran este tipo de ecuaciones se clasifican en problemas con condiciones iniciales y problemas con condiciones en la frontera. Muchos de los problemas con condiciones iniciales dependen del tiempo, es decir, las condiciones que permiten la solución están dadas en función del tiempo inicial. El problema de condiciones iniciales de una ecuación diferencial ordinaria de primer orden se puede escribir de la forma:

$$y'(t) = f(y,t), y(0) = y_0 \ldots \text{VI.1}$$

Donde $f(y,t)$ es una función de y en tanto que t y la segunda ecuación es una condición inicial.

VI.II Métodos de Euler.

Estos métodos son adecuados para una programación rápida debido a su sencillez. Estos métodos se dividen en tres tipos:

- Euler hacia adelante.
- Euler modificado.
- Euler hacia atrás.

VI.II.I Método de Euler hacia adelante.

VI.II.I.I Algoritmo.

El método para la ecuación $y' = f(y,t)$ se obtiene reescribiendo la aproximación por diferencias hacia delante:

$$\frac{y_{n+1} - y_n}{h} \simeq y_n' \ldots \text{VI.2}$$

Como:

$$y_{n+1} = y_n + hf\left(y_n, t_n\right) \ldots \text{VI.3}$$

Donde se usa $y' = f\left(y_n, t_n\right)$. Haciendo uso de la ecuación VI.3 se puede calcular y_n de manera recursiva:

$$\begin{aligned} y_1 &= y_0 + hy_0' = y_0 + hf\left(y_0, t_0\right) \\ y_2 &= y_1 + hf\left(y_1, t_1\right) \\ y_3 &= y_2 + hf\left(y_2, t_2\right) \qquad \ldots \text{VI.4} \\ &\vdots \\ y_n &= y_{n-1} + hf\left(y_{n-1}, t_{n-1}\right) \end{aligned}$$

El método es muy sencillo, sin embargo debe utilizarse de forma cuidadosa para evitar dos tipos de error. Primero se tienen los errores de truncamiento, el segundo tipo son los errores generados por una posible inestabilidad que aparece cuando la constante del tiempo es negativa, a menos que el intervalo h sea suficientemente pequeño.

VI.II.I.II Ejemplo.[2]

Resolver $y' = -20y + 7e^{(-0.5t)}$, $y(0) = 5$, por medio del método de Euler hacia delante con $h = 0.01, 0.001$ y 0.0001 para $0 \le t \le 0.10$.

Evaluar los errores de los tres cálculos comparando los resultados con los obtenidos mediante la solución analítica dada por:

$$y = 5e^{-20t} + \left(\frac{7}{19.5}\right)\left(e^{-0.5t} - e^{-20t}\right).$$

Para $h = 0.01$ los primeros cálculos son los siguientes:

$$\begin{aligned} &t_0 = 0, && y_0 = y(0) = 5 \\ &t_1 = 0.01, && y_1 = y_0 + hy_0' = 5 + (0.01)\left(-20(5) + 7e^{(0)}\right) = 4.07 \\ &t_2 = 0.02, && y_2 = y_1 + hy_1' = 4.07 + (0.01)\left(-20(4.07) + 7e^{(-0.005)}\right) = 3.32565 \\ &\vdots \\ &t_n = nh, && y_n = y_{n-1} + hy_{n-1}' \end{aligned}$$

En la siguiente tabla se incluyen todos los resultados de este tipo de datos:

t	*Analitica*	$h=0.01$	*Error*	$h=0.001$	*Error*	$h=0.0001$	*Error*
0.00	5.00000	5.00000	0.00	5.00000	0.00	5.00000	0.00
0.01	4.15693	4.07000	8.69	4.14924	0.77	4.15617	0.08
0.02	3.46638	3.32565	14.07	3.45379	1.26	3.46513	0.12
0.03	2.90068	2.72982	17.09	2.88524	1.54	2.89915	0.15
0.04	2.43721	2.25282	18.44	2.42037	1.68	2.43554	0.17
0.05	2.05745	1.87087	18.66	2.04023	1.72	2.05574	0.17
0.06	1.74622	1.56497	18.12	1.72932	1.69	1.74454	0.17
0.07	1.49109	1.31990	17.12	1.47496	1.61	1.48949	0.16
0.08	1.28191	1.12352	15.84	1.26683	1.51	1.28041	0.15
0.09	1.11033	0.96607	14.43	1.09646	1.39	1.10895	0.14
0.10	0.96956	0.83977	12.98	0.95696	1.26	0.96831	0.13

Tabla VI.1 Ejemplo Euler Hacia Adelante.

VI.II.II Método de Euler hacia adelante para un conjunto de EDO.

VI.II.II.I Algoritmo.

Considerando un conjunto de ecuaciones diferenciales ordinarias de primer orden:

$$\begin{aligned} y' &= f(y,z,t),\ y(0)=y_0 \\ z' &= g(y,z,t),\ z(0)=z_0 \end{aligned} \quad \text{...VI.5}$$

El método de Euler hacia delante para VI.5 se escribe:

$$\begin{aligned} y_{n+1} &= y_n + hy_n' = y_n + hf(y_n,z_n,t_n) \\ z_{n+1} &= z_n + hz_n' = z_n + hg(y_n,z_n,t_n) \end{aligned} \quad \text{...VI.6}$$

Una ecuación diferencial ordinaria de orden superior se puede descomponer en un sistema simultáneo de ecuaciones diferenciales de primer orden.

VI.II.II.II Ejemplo.[2]

Por medio del método de Euler hacia delante con $h=0.5$, determine los valores de $y(1)$ y $y'(1)$ para:

$$y''(t)-0.05y'(t)+0.15y(t)=0,\ \ y'(0)=0,\ \ y(0)=1$$

Si $y'=z$, entonces la ecuación diferencial ordinaria de segundo orden es:

$$\begin{aligned} y' &= z, && y(0)=1 \\ z' &= 0.05z-0.15y, && z(0)=0 \end{aligned}$$

Por medio del método de Euler hacia adelante se obtiene y y z en $n=1$ y $n=2$:
$t=0.5$:

$$y_0^{'}=z_0=0$$
$$z_0^{'}=0.05z_0-0.15y_0=-0.15$$
$$y_1=y_0+hy_0^{'}=1+(0.5)(0)=1$$
$$z_1=z_0+hz_0^{'}=0+(0.5)(-0.15)=-0.075$$

$t=1$:

$$y_1^{'}=z_1=-0.075$$
$$z_1^{'}=0.05z_1-0.15y_1=(0.05)(-0.075)-(0.15)(1)=-0.15375$$
$$y_2=y_1+hy_1^{'}=1+(0.5)(-0.075)=0.96250$$
$$z_2=z_1+hz_1^{'}=-0.075+(0.5)(-0.15375)=-0.15187$$

Por lo tanto:

$$y(1)=y_2=0.96250$$
$$y^{'}(1)=z(1)=z_2=-0.15187$$

VI.II.III Método de Euler modificado.

El método de Euler modificado tiene dos ventajas sobre el método hacia delante:

1. Es más preciso.
2. Es más estable.

VI.II.III.I Algoritmo.

Se obtiene al aplicar la regla del trapecio para integrar $y^{'}=f(y,t)$:

$$y_{n+1}=y_n+\frac{h}{2}\left[f(y_{n+1},t_{n+1})+f(y_n,t_n)\right]\ldots\text{VI.7}$$

Si f es lineal en y, la ecuación VI.7 se puede resolver de manera sencilla en términos de y_{n+1}.

Si f es una ecuación no lineal de y, la ecuación VI.7 es una función no lineal de y_{n+1}, por lo que:

$$y_{n+1}^{(k)}-y_n=\frac{h}{2}\left[f\left(y_{n+1}^{(k-1)},t_{n+1}\right)+f(y_n,t_n)\right]\ldots\text{VI.8}$$

Donde $y_{n+1}^{(k)}$ es la k-ésima aproximación iterativa de y_{n+1}, y $y_{n+1}^{(0)}$ es una estimación inicial de y_{n+1}. Esta iteración se detiene si $\left|y_{n+1}^{(k)} - y_{n+1}^{(k-1)}\right|$ es menor que una cierta tolerancia preestablecida. La estimación inicial es igual a y_n. Entonces el primer paso de iteración es idéntico al método de Euler hacia adelante

VI.II.IV Método de Euler hacia atrás.

VI.II.IV.I Algoritmo.

Este método se basa en aproximación por diferencias hacia atrás, y se escribe como:

$$y_{n+1} = y_n + hf\left(y_{n+1}, t_{n+1}\right) \ldots \text{VI.9}$$

La precisión de este método es exactamente igual a la que tiene el de Euler hacia delante.

VI.III Métodos de Runge – Kutta.

La principal desventaja de los métodos de Euler consiste en que el grado de precisión es bajo. Para mantener una alta precisión se necesita una h pequeña, lo que aumenta el tiempo de cálculo y provoca errores de redondeo.

En los métodos de Runge – Kutta, el orden de precisión aumenta al utilizar puntos intermedios en cada intervalo. Una mayor precisión implica además que los errores decrecen más rápido al reducir h.

Considerando la ecuación diferencial ordinaria:

$$y'(t) = f(y,t), y(0) = y_0 \ldots \text{VI.1}$$

Para calcular y_{n+1} en $t_{n+1} = t_n + h$, dado un valor de y_n, integrando la ecuación VI.1 en el intervalo $[t_n, t_{n+1}]$:

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(y,t)\,dt \ldots \text{VI.10}$$

Así, los métodos de Runge – Kutta se obtienen al aplicar un método de integración numérica al lado derecho de la ecuación VI.10.[2]

VI.III.I Método de Runge – Kutta de segundo orden.

VI.III.I.I Algoritmo.

Aplicando la regla del trapecio a la ecuación VI.10:

$$\int_{t_n}^{t_{n+1}} f(y,t)\,dt \simeq \frac{1}{2}h\left[f(y_n,t_n) + f(y_{n+1},t_{n+1})\right] \ldots \text{VI.11}$$

En esta ecuación, y_{n+1} es una incógnita, por lo que se aproxima el segundo término mediante $f\left(\overline{y}_{n+1},t_{n+1}\right)$ donde $\overline{y}_{n+1}$ es la primera estimación de y_{n+1} obtenida mediante el método de Euler hacia delante.

Esto se puede resumir como:

$$\begin{aligned} k_1 &= hf(y_n,t_n) \\ k_2 &= hf(y_n+k_1,t_{n+1}) \quad \ldots \text{VI.12} \\ y_{n+1} &= y_n + \frac{1}{2}\left[k_1+k_2\right] \end{aligned}$$

Ahora, el método de Runge – Kutta de segundo orden para una ecuación diferencial ordinaria de orden superior se escribe como:

$$\begin{aligned} k_1 &= hf(y_n,z_n,t_n) = hz_n \\ l_1 &= hg(y_n,z_n,t_n) = h\left(-az_n - by_n + q_n\right) \\ k_2 &= hf(y_n+k_1,z_n+l_1,t_{n+1}) = h\left(z_n+1\right) \\ l_2 &= hg(y_n+k_1,z_n+l_1,t_{n+1}) = h\left(-a\left(z_n+l_1\right) - b\left(y_n+k_1\right) + q_{n+1}\right) \ldots \text{VI.13} \\ y_{n+1} &= y_n + \frac{1}{2}\left(k_1+k_2\right) \\ z_{n+1} &= z_n + \frac{1}{2}\left(l_1+l_2\right) \end{aligned}$$

VI.III.I.II Ejemplo.[2]

El circuito que se muestra en la figura tiene una autoinductancia de $L = 50H$, una resistencia de $R = 20\left[ohms\right]$ y una fuente de voltaje de $V = 10\left[volts\right]$. Si el interruptor se cierra en el instante $t = 0$, la corriente $I(t)$ satisface la ecuación:

$$L\frac{d}{dt}I(t) + RI(t) = E, \quad I(0) = 0$$

Determinar el valor de la corriente para $0 < t \leq 10$ segundos, mediante el método de Runge – Kutta de segundo orden, con $h = 0.1$

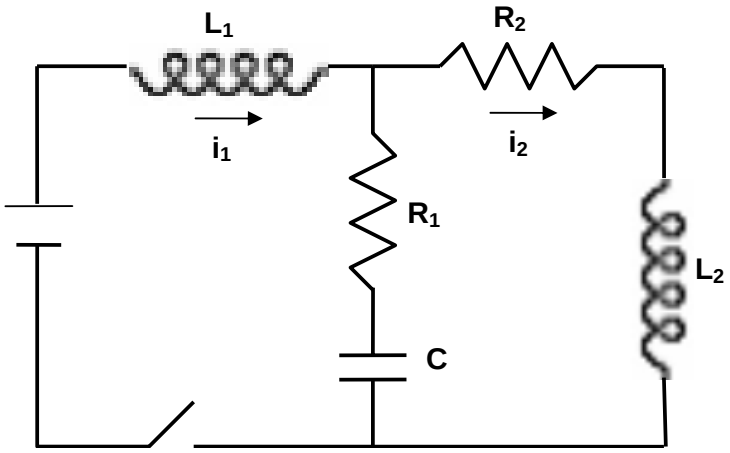


Figura VI.1 Ejemplo Runge – Kutta Segundo Orden.

Reescribiendo la ecuación como:

$$\frac{d}{dt} I = -\frac{R}{L} I + \frac{E}{L} \equiv f\left(I, t\right)$$

Entonces el método de Runge – Kutta de segundo orden se escribe:

$$k_1 = h\left[-\frac{R}{L} I_n + \frac{E}{L}\right]$$

$$k_2 = h\left[-\frac{R}{L}\left(I_n + k_1\right) + \frac{E}{L}\right]$$

$$I_{n+1} = I_n + \frac{1}{2}\left(k_1 + k_2\right)$$

Entonces, para los primeros dos pasos:

$n = 0\left(t = 0.1\right)$:

$$k_1 = 0.1\left[\left(-0.4\right)\left(0\right) + 0.2\right] = 0.02$$

$$k_2 = 0.1\left[\left(-0.4\right)\left(0 + 0.02\right) + 0.2\right] = 0.0192$$

$$I_1 = I_0 + \frac{1}{2}\left(k_1 + k_2\right) = 0 + \frac{1}{2}\left(0.02 + 0.0192\right) = 0.0196$$

$n=1\left(t=0.2\right)$:

$$k_1 = 0.1\left[\left(-0.4\right)\left(0.0196\right)+0.2\right] = 0.019216$$
$$k_2 = 0.1\left[\left(-0.4\right)\left(0.0196+0.019216\right)+0.2\right] = 0.018447$$
$$I_1 = I_0 + \frac{1}{2}\left(k_1+k_2\right) = 0.0196+\frac{1}{2}\left(0.019216+0.018447\right) = 0.038431$$

En la siguiente tabla se resume el cálculo para $t=1\left[seg\right]$:

$t\left[seg\right]$	k_1	k_2	I_i
0.0	0.0000	0.0000	0.0000
0.1	0.0200	0.0192	0.0196
0.2	0.0192	0.0184	0.0384
0.3	0.0185	0.0177	0.0565
0.4	0.0177	0.0170	0.0739
0.5	0.0170	0.0164	0.0906
0.6	0.0164	0.0157	0.1067
0.7	0.0157	0.0151	0.1221
0.8	0.0151	0.0145	0.1369
0.9	0.0145	0.0139	0.1511
1.0	0.0140	0.0134	0.1648

Tabla VI.2 Resultados Runge – Kutta Segundo Orden $1\left[seg\right]$.

Realizando el mismo procedimiento para $1\left[seg\right] \le t \le 10\left[seg\right]$ se tiene:

$t\left[seg\right]$	k_1	k_2	I_i
1.0	0.0140	0.0134	0.1648
2.0	0.0094	0.0090	0.2753
3.0	0.0063	0.0060	0.3494
4.0	0.0042	0.0040	0.3990
5.0	0.0028	0.0027	0.4323
6.0	0.0019	0.0018	0.4546
7.0	0.0013	0.0012	0.4696
8.0	0.0008	0.0008	0.4796
9.0	0.0006	0.0005	0.4863
10.0	0.0004	0.0004	0.4908
∞	0.0000	0.0000	0.5000

Tabla VI.3 Resultados Runge – Kutta Segundo Orden $1\left[seg\right] \le t \le 10\left[seg\right]$.

VI.III.II Método de Runge – Kutta de tercer orden.

Este método es de mayor precisión que el anterior y resulta de una integración numérica de orden superior para el segundo término de la ecuación:

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(y,t)\,dt \quad \text{...VI.10}$$

VI.III.II.I Algoritmo.

Con la regla de 1/3 de Simpson la ecuación VI.10 se escribe:

$$y_{n+1} = y_n + \frac{h}{6}\left[f(y_n,t_n) + 4f\left(\bar{y}_{n+\frac{1}{2}}, t_{n+\frac{1}{2}}\right) + f\left(\bar{y}_{n+1}, t_{n+1}\right)\right] \quad \text{...VI.14}$$

Donde $\bar{y}_{n+1}$ y $\bar{y}_{n+\frac{1}{2}}$ son estimaciones, puesto que no se conocen y_{n+1} y $y_{n+\frac{1}{2}}$, la estimación $\bar{y}_{n+\frac{1}{2}}$ se obtiene mediante el método de Euler hacia delante:

$$\bar{y}_{n+\frac{1}{2}} = y_n + \frac{h}{2} f(y_n,t_n) \quad \text{...VI.15}$$

La estimación $\bar{y}_{n+1}$ es:

$$\bar{y}_{n+1} = y_n + hf(y_n,t_n) \text{ o } \bar{y}_{n+1} = y_n + hf\left(\bar{y}_{n+\frac{1}{2}}, t_{n+\frac{1}{2}}\right)$$

O como una combinación lineal de ambas:

$$\bar{y}_{n+1} = y_n + h\left[\theta f(y_n,t_n) + (1-\theta) f\left(\bar{y}_{n+\frac{1}{2}}, t_{n+\frac{1}{2}}\right)\right] \quad \text{...VI.16}$$

Donde θ es un parámetro que se debe determinar de manera que la precisión del método se maximice. Con la ecuación VI.16 se tiene:

$$\begin{aligned} k_1 &= hf(y_n,t_n) \\ k_2 &= hf\left(y_n + \frac{1}{2}k_1, t_n + \frac{h}{2}\right) \\ k_3 &= hf\left(y_n + \theta k_1 + (1-\theta)k_2, t_n + h\right) \\ y_{n+1} &= y_n + \frac{1}{6}\left[k_1 + 4k_2 + k_3\right] \end{aligned} \quad \text{...VI.17}$$

En resumen, el método se escribe como:

$$\begin{aligned} k_1 &= hf\left(y_n, t_n\right) \\ k_2 &= hf\left(y_n + \frac{1}{2}k_1, t_n + \frac{h}{2}\right) \\ k_3 &= hf\left(y_n - k_1 + 2k_2, t_n + h\right) \\ y_{n+1} &= y_n + \frac{1}{6}\left[k_1 + 4k_2 + k_3\right] \end{aligned} \quad \text{…VI.18}$$

VI.III.III Método de Runge – Kutta de cuarto orden.

VI.III.III.I Algoritmo.

Existen dos versiones para este método, la primera se basa en la regla de 1/3 de Simpson y se escribe:

$$\begin{aligned} k_1 &= hf\left(y_n, t_n\right) \\ k_2 &= hf\left(y_n + \frac{k_1}{2}, t_n + \frac{h}{2}\right) \\ k_3 &= hf\left(y_n + \frac{k_2}{2}, t_n + \frac{h}{2}\right) \\ k_4 &= hf\left(y_n + k_3, t_n + h\right) \\ y_{n+1} &= y_n + \frac{1}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right] \end{aligned} \quad \text{…VI.19}$$

El segundo se deriva de la regla 3/8 de Simpson y se escribe:

$$\begin{aligned} k_1 &= hf\left(y_n, t_n\right) \\ k_2 &= hf\left(y_n + \frac{k_1}{3}, t_n + \frac{h}{3}\right) \\ k_3 &= hf\left(y_n + \frac{k_1}{3} + \frac{k_2}{3}, t_n + \frac{2h}{3}\right) \\ k_4 &= hf\left(y_n + k_1 - k_2 + k_3, t_n + h\right) \\ y_{n+1} &= y_n + \frac{1}{8}\left[k_1 + 3k_2 + 3k_3 + k_4\right] \end{aligned} \quad \text{…VI.20}$$

VII. ECUACIONES DIFERENCIALES PARCIALES.

VII.I Introducción.

Una gran cantidad de problemas de ingeniería tiene solución por medio de ecuaciones diferenciales parciales. Un caso típico es el que se tiene en la conducción de calor sobre algún cuerpo.

La clasificación de las ecuaciones diferenciales parciales de segundo orden en elípticas, parabólicas o hiperbólicas, depende de los coeficientes de los términos de la segunda derivada. Cualquier ecuación de segundo orden se puede escribir como:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right) = 0$$

Dependiendo del valor de $b^2 - 4ac$, se clasifica a las ecuaciones como.

Elípticas, si $B^2 - 4AC < 0$;
Parabólicas, si $B^2 - 4AC = 0$;
Hiperbólicas, si $B^2 - 4AC > 0$.

En el caso de la Ingeniería Petrolera este tipo de ecuaciones se utilizan en muchas áreas para diversos problemas, uno de los más importantes es el de la Simulación Numérica de Yacimientos.

En la simulación numérica, el flujo de fluidos a través de medios porosos se representa por medio de la ecuación de Darcy mientras que la ecuación de continuidad se encarga de la conservación de masa.

De esta manera combinando ambas ecuaciones y considerando las propiedades de los fluidos resulta en la ecuación conocida como ecuación de difusividad, esta tiene la capacidad de describir el comportamiento del yacimiento.

Más adelante se hará el análisis de la ecuación de difusividad, pero cabe mencionar, que se trata de una ecuación en derivadas parciales de segundo orden y cuya forma general es no lineal, lo cual provoca que su solución se realice de forma numérica.

VII.II Ecuaciones parabólicas.

Este tipo de ecuaciones se utilizan para dar solución a problemas que involucran el tiempo. Uno de estos problemas es el flujo calorífico de estado inestable y esta representado por una ecuación diferencial parcial parabólica.

VII.II.I Ecuaciones de conducción de calor.

El caso mas simple para este problema es aquel en el que se considera flujo de calor en una sola dirección.

Considerando una varilla de sección transversal uniforme y aislada alrededor de su perímetro, de tal manera que el calor solo fluye longitudinalmente.

Considerando una parte diferencial de la varilla de longitud dx con un área de sección transversal A.

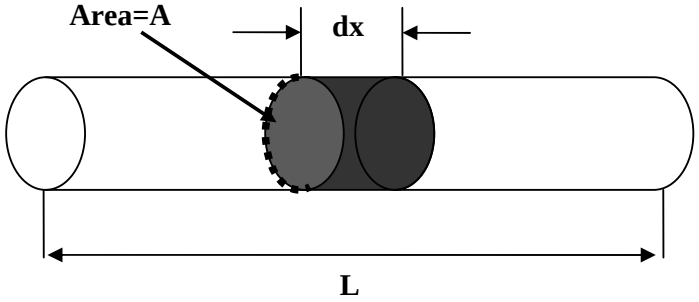


Figura VII.1 Varilla Conducción de Calor.

Sea u la temperatura en cualquier punto en la varilla, cuya distancia al extremo izquierdo es x. El calor esta fluyendo de izquierda a derecha bajo la influencia del gradiente de temperatura $\partial u / \partial x$. Debe realizarse un balance de la rapidez de flujo calorífico que entra y sale del elemento utilizando k para la conductividad térmica, la cual se supone constante:

La rapidez de flujo del calor que entra en la varilla se expresa:

$$-kA\frac{\partial u}{\partial x} \quad \ldots \text{VII.1}$$

La rapidez de flujo del calor que sale en la varilla se expresa:

$$-kA\left(\frac{\partial u}{\partial x}+\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)dx\right)\ldots\text{VII.2}$$

La diferencia entre la rapidez de flujo que entra y que sale es la rapidez en la cual el calor está siendo almacenado en el elemento. Si c es la capacidad calorífica con unidades $cal/g\cdot{}^{\circ}C$ y ρ es la densidad en g/cm^3 y t representa el tiempo en segundos:

$$-kA\frac{\partial u}{\partial x}-\left(kA\left(\frac{\partial u}{\partial x}+\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)dx\right)\right)=c\rho\left(Adx\right)\frac{\partial u}{\partial t}\ldots\text{VII.3}$$

Simplificando esta expresión:

$$k\frac{\partial^2 u}{\partial x^2}=c\rho\frac{\partial u}{\partial t}\ldots\text{VII.4}$$

Este es el modelo matemático más simple para el flujo de estado inestable. En este caso es para el flujo de calor, pero se aplica igualmente a la difusión de material, flujo de electricidad en cables, flujo de fluidos, etc.

Esta ecuación se clasifica como parabólica por que si se compara con la forma estándar de una ecuación con dos variables independientes x y t, se tiene:

$$A\frac{\partial^2 u}{\partial x^2}+B\frac{\partial^2 u}{\partial x\partial t}+C\frac{\partial^2 u}{\partial t^2}+D\left(x,t,u,\frac{\partial u}{\partial x},\frac{\partial u}{\partial t}\right)=0\ldots\text{VII.5}$$

Se encuentra que $B^2-4AC=0$.

En dos o tres dimensiones, se aplica las ecuaciones:

$$k\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=c\rho\frac{\partial u}{\partial t}\ldots\text{VII.6}$$

$$k\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)=c\rho\frac{\partial u}{\partial t}\ldots\text{VII.7}$$

VII.II.II Condiciones iniciales y de frontera.

La solución no sólo debe obedecer a la ecuación diferencial, también debe satisfacer una condición inicial y a un conjunto de condiciones de frontera. Para el

problema de flujo de calor en una dimensión la condición inicial en todos los puntos a lo largo de la varilla es:

$$u(x,t)\big|_{t=0} = u(x,0) = f(x) \ldots \text{VII.8}$$

Las condiciones en la frontera describirán la temperatura en cada extremo de la varilla como funciones del tiempo, por ejemplo:

$$\begin{aligned} u(0,t) &= c_1 \\ u(L,t) &= c_2 \end{aligned} \ldots \text{VII.9}$$

Estas significan que en los extremos de la varilla no existe cambio en la temperatura.

En el caso de la simulación numérica de yacimientos las condiciones de frontera se pueden referir a la pared de los pozos y a la frontera del yacimiento, la condición inicial se refiere al estado en el que se encuentra el yacimiento al comienzo de su explotación.

VII.III Método explícito, implícito y Crank-Nicholson.

A partir de esta sección se abordara el tema desde el punto de vista de la simulación numérica de yacimientos olvidando las ecuaciones de conducción de calor de la sección anterior, sin embargo los métodos tratados son igualmente aplicables a dichas ecuaciones.

El criterio para resolver ecuaciones diferenciales parciales parabólicas por medio de un método numérico consiste en reemplazar las derivadas parciales por aproximaciones en diferencias finitas.

Entonces, olvidando la ecuación VII.4 y considerando la ecuación AVI.9 que es la ecuación de difusión para el flujo de fluidos a través de medios porosos:

$$\frac{\partial^2 p}{\partial x^2} = \frac{\phi \mu c_t}{k} \frac{\partial p}{\partial t} \ldots \text{AVI.9}$$

Para resolver el término $\frac{\partial^2 p}{\partial x^2}$ se utilizan diferencias centrales, esto es (por lo visto en el capítulo V) debido a que las diferencias centrales tienen un precisión mas alta, la expresión que define las diferencias centrales es:

$$f_i^{'} = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O\left(\Delta x^2\right) \ldots \text{V.13}$$

Para el término $\frac{\phi\mu c_t}{k}\frac{\partial p}{\partial t}$ no es muy común utilizar diferencias centrales, el uso de diferencias regresivas o hacia atrás no es recomendable por cuestiones de estabilidad numérica por lo que se acostumbra hacer uso de diferencias progresivas o hacia delante:

$$f_i^{'} = \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x) \ldots \text{V.9}$$

Y considerando que el valor de p^n es un valor conocido para un tiempo t^n en cada punto o celda, y se requiere conocer p^{n+1} al tiempo t^{n+1}.

VII.III.I Método explícito.

En este método se resuelve para una sola incógnita para el nuevo tiempo t^{n+1}, entonces la solución para la ecuación AVI.9 se representa como:

$$\frac{p_{i+1}^n - 2p_i^n + p_{i-1}^n}{\Delta x^2} = \eta \frac{p_i^{n+1} - p_i^n}{\Delta t} \ldots \text{VII.10}$$

Donde $\eta = \frac{\phi\mu c_t}{k}$.

Entonces la única incógnita es p^{n+1} puesto que las otras presiones p^n son conocidas, este método tiene la desventaja de solo ser estable cuando $\frac{\eta\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$.

En resumen se puede decir que en este método es necesario conocer tres valores de presión para así poder determinar uno a un tiempo futuro.

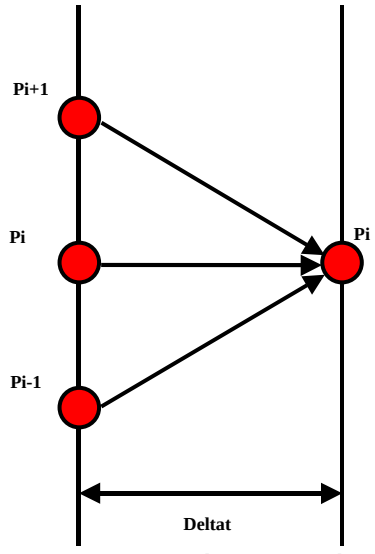


Figura VII.2 Método Explícito.

VII.III.II Método implícito.

En este caso los valores de p en la derivada de espacio se evalúan al tiempo t^{n+1}, entonces la ecuación AVI.9 se resuelve:

$$\frac{p_{i+1}^{n+1} - 2p_i^{n+1} + p_{i-1}^{n+1}}{\Delta x^2} = \left(\frac{\phi \mu c_t}{k}\right) \frac{p_i^{n+1} - p_i^n}{\Delta t} \quad \ldots \text{VII.11}$$

Por lo que se tienen tres incógnitas $\left(p_{i-1}^{n+1}, p_i^{n+1}, p_{i+1}^{n+1}\right)$ para cada intervalo de tiempo, entonces, utilizando la ecuación VII.11 se tiene un sistema de ecuaciones lineales de orden N, este orden esta dado por el número de celdas en las que se divide el dominio, teniendo N incógnitas con N ecuaciones.

Como se puede observar en la siguiente figura, basta con conocer una presión para así poder obtener tres nuevas presiones a un tiempo futuro, pues la solución se realiza de manera simultánea.

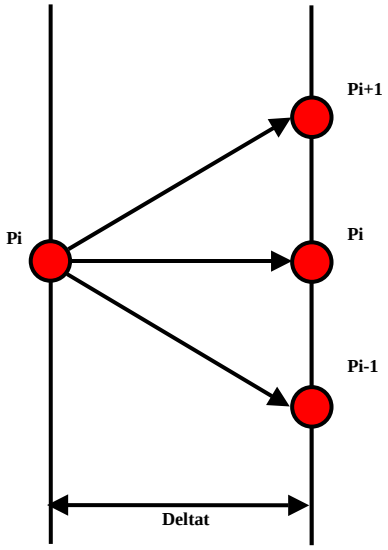


Figura VII.3 Método Implícito.

VII.III.III Método Crank – Nicholson.

Involucra una combinación de los valores de la variable independiente que en este caso es p, para los tiempos n y $n+1$, esto es, la derivada con respecto a distancia (espacial) es promediada para los tiempos n y $n+1$. Entonces para $\frac{\partial^2 p}{\partial x^2}$:

$$\left[\frac{\partial^2 p}{\partial x^2}\right]_i^{n+\frac{1}{2}} = \frac{1}{2}\left\{\left[\frac{\partial^2 p}{\partial x^2}\right]_i^{n+1} + \left[\frac{\partial^2 p}{\partial x^2}\right]_i^{n}\right\} \quad \ldots \text{VII.12}$$

Para el término $\frac{\partial p}{\partial t}$

$$\left.\frac{\partial p}{\partial t}\right|_i^{n+\frac{1}{2}} = \frac{p_i^{n+1} - p_i^n}{\Delta t} \quad \text{...VII.13}$$

Entonces finalmente se tiene:

$$\frac{1}{2}\left[\frac{p_{i+1}^{n+1} - 2p_i^{n+1} + p_{i-1}^{n+1}}{(\Delta x)^2} + \frac{p_{i+1}^{n} - 2p_i^{n} + p_{i-1}^{n}}{(\Delta x)^2}\right] = \left(\frac{\phi \mu c_t}{k}\right)\frac{p_i^{n+1} - p_i^n}{\Delta t} \quad \text{...VII.14}$$

De igual forma que con el método implícito se tiene un sistema de ecuaciones lineales. Aunque esta formulación a diferencia de la obtenida en el método implícito es incondicionalmente estable se usa poco pues requiere de un trabajo mayor.

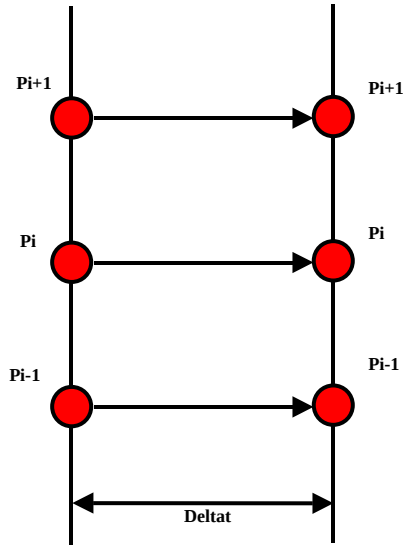


Figura VII.4 Método Crank – Nicholson.

En este caso para cada presión conocida se puede determinar la correspondiente a un tiempo futuro.

En el siguiente capítulo, como parte de las aplicaciones propuestas, se presenta la solución numérica a estos esquemas.

VII.IV Ecuaciones Elípticas.

Este tipo de ecuaciones se aplican a problemas que son independientes del tiempo, a diferencia de las vistas anteriormente.

Si u es una función de dos variables independientes x e y, hay tres derivadas parciales de segundo orden:

$$\frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial x \partial y}, \frac{\partial^2 u}{\partial y^2}$$

Los problemas de la distribución del potencial de estado estable caen dentro de las ecuaciones elípticas.

VII.IV.I Ecuaciones para flujo calorífico de estado estable.

Se deduce la relación para la temperatura *u*, como una función de las dos variables espaciales x e y, para la distribución de la temperatura de equilibrio sobre una placa plana.

De la siguiente figura, considérese el elemento de la placa cuya área superficial es dx, dx y t el grosor. Se supone que el calor fluye sólo en las direcciones x, y, y no en la dirección perpendicular.

Si la placa es muy delgada o si la superficie superior e inferior están bien aisladas, la situación física estará de acuerdo con dicha suposición.

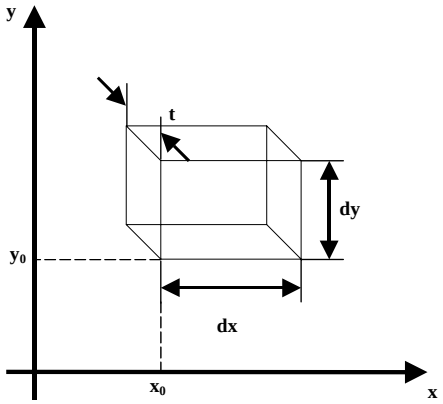


Figura VII.5 Ecuación Flujo Calorífico de Estado Estable.

El calor fluye a una rapidez proporcional al área de la sección transversal, al gradiente de temperatura $\left(\partial u / \partial x \, o \, \partial u / \partial y\right)$, y a la conductividad térmica k, la cual se supondrá constante en todos los puntos. El flujo de calor pasa desde la temperatura más alta a la más baja, significando que se opone a la dirección del gradiente que se incrementa. Se utiliza un signo negativo en la ecuación para tomar esto en consideración. La rapidez de flujo calorífico dentro de un elemento en $x = x_n$ en la dirección de x:

$$k\left(tdy\right)\frac{\partial u}{\partial x} \ldots \text{VII.15}$$

El gradiente en $x_n + dx$ es el gradiente en x_0 más el incremento del gradiente en la distancia dx, gradiente en $x_0 + dx$:

$$\frac{\partial u}{\partial x} + \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)dx \ldots \text{VII.16}$$

Rapidez de flujo saliendo del elemento en $x = x_0 + dx$:

$$-k\left(tdy\right)\left[\frac{\partial u}{\partial x} + \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)dx\right] \ldots \text{VII.17}$$

Rapidez neta de flujo calorífico al elemento en la dirección x:

$$-k\left(tdy\right)\left[\frac{\partial u}{\partial x} - \left(\frac{\partial u}{\partial x} + \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)dx\right)\right] = ktdxdy\frac{\partial^2 u}{\partial x^2} \ldots \text{VII.18}$$

De igual manera, en la dirección y se tiene la rapidez neta de flujo calorífico al elemento en la dirección y:

$$-k\left(tdx\right)\left[\frac{\partial u}{\partial y} - \left(\frac{\partial u}{\partial y} + \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)dy\right)\right] = ktdxdy\frac{\partial^2 u}{\partial y^2} \ldots \text{VII.19}$$

El flujo calorífico total en el volumen elemental por conducción es la suma de estos flujos netos en las direcciones x e y. Si hubiera calor generado dentro del elemento, éste sería adicionado al calor que entra por conducción. La suma debe ser igual a la rapidez de pérdida calorífica de otros mecanismos, o también habrá una acumulación de calor dentro del elemento, haciendo que la temperatura se incremente con el tiempo, en este caso ya no se trataría de un problema elíptico pues se estaría involucrando al tiempo y dejaría de ser estado estable.

Por lo tanto, si hay equilibrio en lo que respecta a la distribución de temperaturas, la rapidez total de flujo calorífico en el elemento más el calor generado debe ser cero:

$$kt\left(dx\right)\left(dy\right)\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + Qt\left(dx\right)\left(dy\right) = 0 \ldots \text{VII.20}$$

En donde Q es la rapidez de generación de calor por unidad de volumen.

Si $Q = 0$, se tiene:

$$kt\left(dx\right)\left(dy\right)\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=0$$

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=\nabla^2 u=0 \quad \ldots \text{VII.21}$$

El operador

$$\nabla^2=\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right) \ldots \text{VII.22}$$

Se conoce como laplaciano, y la ecuación VII.21 es llamada ecuación de Laplace.

VIII. APLICACIONES DE INGENIERÍA PETROLERA.

VIII.I Introducción.

Dentro de la ingeniería petrolera en sus diferentes ramas es necesario realizar una gran cantidad de cálculos que permiten tomar decisiones sobre la forma en la que se explotará un yacimiento por medio de predicciones a futuro, el diseño de la perforación de los pozos que permitirán dicha explotación, conocer el tipo de fluidos producidos para así diseñar las instalaciones de producción y almacenaje ubicadas en la superficie, todos estos cálculos pueden requerir de mucho tiempo para obtener resultados, sin embargo con el uso de una computadora esta inversión de tiempo puede reducirse considerablemente.

Algunos cálculos al ser programados, requieren de una solución numérica como las tratadas en este trabajo, esto implica que se necesita de conocimientos avanzados de programación para realizar un software que permita resolver este tipo de problemas, sin embargo hay otros que no necesitan de este tipo de solución, pues los resultados se obtienen al aplicar los datos de entrada indicados por el usuario a una ecuación que proporciona un valor que después se requerirá en otra, así hasta que se obtenga el resultado final, éstos sólo requieren de conocimientos básicos de programación

VIII.II Equilibrio líquido – vapor.

Conocer la cantidad de líquido y vapor que existe en una mezcla de hidrocarburos permite calcular algunas propiedades importantes, tales como: B_o, B_g, B_t, RGA, viscosidad, que serán usados en la evaluación de reservas.

VIII.II.I Algoritmo.[9]

La suma de la fase líquida más la fase gas de una mezcla debe ser igual a 1, por lo tanto:

$$L + V = 1.0 \ldots \text{VIII.1}$$

Ahora, considerando la fracción molar de cada componente de la mezcla:

$$z_i = x_i L + y_i V\ ,\ i = 1,2,\ldots,n \ldots \text{VIII.2}$$

Donde n es el número de componentes de la mezcla, además se tiene que:

$$\begin{aligned} &\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i = 1 \\ &\sum_{i=1}^{n} \left(y_i - x_i\right) = 1 = f\left(v\right) \end{aligned} \quad \text{VIII.3}$$

La constante de equilibrio k_i definida como:

$$k_i = \frac{y_i}{x_i}\ ,\ i = 1,2,\ldots,n \ldots \text{VIII.4}$$

Combinando las anteriores ecuaciones, se tiene:

$$x_i = \frac{z_i}{1.0 + (k_i - 1.0)v} \ \ldots \text{VIII.5}$$

$$y_i = \frac{k_i z_i}{1.0 + \left(k_i - 1.0\right)v} \ldots \text{VIII.6}$$

Sustituyendo VIII.5 y VIII.6 en VIII.3, se tiene lo siguiente:

$$f\left(v\right) = \sum_{i=1}^{n} \frac{z_i \left(k_i - 1\right)}{1 + \left(k_i - 1\right)v} = 0 \ldots \text{VIII.7}$$

Comúnmente para resolver esta ecuación se suponen valores de v, comenzando en 0.5 que implica 50% de líquido y 50% vapor, y se sustituye en la ecuación VIII.5 para obtener las fracciones de líquido x_i de cada componente, entonces si la suma $\sum_{i=1}^{n} x_i$ es igual a 1, se tendrá la fracción de vapor representada por v y la fracción de líquido estará dada por:

$$l = 1 - v \ldots \text{VIII.8}$$

En caso de que el valor de v supuesto no cumpla con $\sum_{i=1}^{n} x_i = 1$ se debe suponer un nuevo valor hasta tener el que más se aproxime, para calcular las fracciones de

líquido y_i de cada componente se utiliza la ecuación:

$$y_i = k_i * x_i \ldots \text{VIII.9}$$

De igual forma que con las fracciones de líquido, la suma $\sum_{i=1}^{n} y_i$ es igual a 1.

La expresión VIII.7 representa una ecuación no lineal y se puede resolver utilizando el método de Newton – Raphson, para esto es necesario obtener la primera derivada de VIII.7, esta es:

$$f'(v) = -\sum_{i=1}^{n} \frac{z_i (k_i - 1)^2}{\left[1 + (k_i - 1)v\right]^2} \ldots \text{VIII.10}$$

Entonces el algoritmo de Newton - Raphson para resolver esta ecuación queda expresado como:

$$V_{K+1} = V_k - \frac{f(V_K)}{f'(V_K)} \ldots \text{VIII.11}$$

Si $abs(V_{K+1} - V_K)$ es menor que cierta tolerancia, el algoritmo converge, lo que permite tener el valor de v que permite tener el equilibrio líquido - vapor exacto de la mezcla de hidrocarburos.

VIII.III Balance de materia en un proceso de separación gas – aceite.

El proceso de separación del gas y el aceite es realizado por una serie de etapas de separación, una etapa de separación se define como la condición a la cual el aceite y el gas alcanzan el equilibrio, a la presión y temperatura existentes en el separador, en la figura se muestra el esquema de una batería de separación gas – aceite en tres etapas.

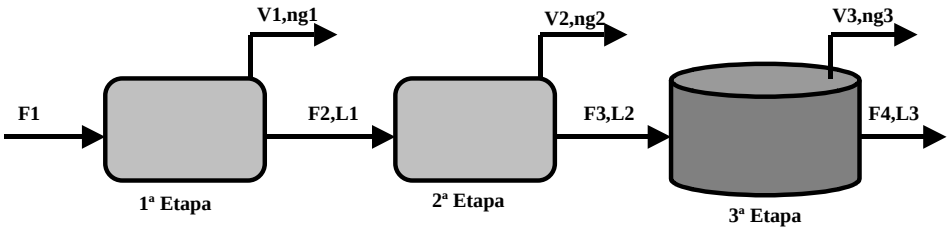


Figura VIII.1 Batería de Separación.

Para determinar las cantidades de gas y líquidos separados se llevan a cabo cálculos de balance de materia, dado que es factible conocer la composición de alimentación al sistema y las condiciones de operación, es posible determinar la constante de equilibrio físico k_i, esta se determina por medio de correlaciones,

gráficas, tabulaciones y ecuaciones de estado, esto lleva al equilibrio líquido – vapor visto anteriormente, obteniendo las fracciones líquido y vapor para cada etapa de separación, de estos balances se pueden obtener factores como: RGA, ºAPI del aceite residual, etc.

VIII.III.I Algoritmo.

Se analiza un sistema en tres etapas como el mostrado en la figura.

Si F_1 son los moles alimentados al sistema, los moles alimentados a la segunda etapa son:

$$F_2 = L_1 = l_1 F_1 \text{...VIII.12}$$

Y a la tercera

$$F_3 = L_2 = l_2 L_1 = l_2 l_1 F_1 \text{...VIII.13}$$

Los moles retenidos en el tanque de almacenamiento son:

$$L_3 = l_3 L_2 = l_1 l_2 l_3 F_1 \ldots \text{VIII.14}$$

En el caso en el que se tenga un número n de etapas se tiene que:

$$L_m = l_m l_{m-1} \ldots l_2 l_1 F_1 = F_1 \prod_{i=1}^{n} l_i \ldots \text{VIII.15}$$

Si $F_1 = 1.0$ los moles en el tanque de almacenamiento por mol de alimentación al sistema son:

$$n_t = \prod_{i=1}^{n} l_i \ldots \text{VIII.16}$$

De manera similar el número de moles de gas separado pueden ser calculados;

$$V_1 = V_1 F_1 \text{...VIII.17}$$

Para la segunda etapa:

$$V_2 = V_2 F_2 = V_2 l_1 F_1 \text{...VIII.18}$$

Para la tercera etapa:

$$V_3 = V_3 F_3 = V_3 l_2 l_1 F_1 \text{...VIII.19}$$

El volumen de aceite en el tanque de almacenamiento puede obtenerse a partir de la densidad y el peso molecular del aceite:

$$V_{ot} = \frac{n_t PM_t}{\rho_{ot}} \text{ ...VIII.20}$$

El volumen de gas separado por etapa es calculado mediante los moles de gas liberado y el volumen molar del gas a condiciones estándar es:

$$VG_i = V_i VM_g \text{ ...VIII.21}$$

La relación gas - aceite definida como volumen de gas a volumen de líquido, a condiciones estándar, puede ser determinada por medio de la expresión:

$$RGA_I = \frac{VG_I}{V_{Ot}} = \frac{VG_I VM_G}{\frac{n_t PM_t}{\rho_{Ot}}} = \frac{VG_I VM_g \rho_{ot}}{n_t PM_t} \text{ ...VIII.22}$$

El peso molecular del aceite en el tanque de almacenamiento es calculado mediante la composición de la fase líquida y el peso molecular de cada componente:

$$PM_t = \sum_I X_I PM_I \text{ ...VIII.23}$$

La relación gas - aceite total es la suma de las relaciones gas - aceite por etapa.

$$RGAT = \sum_{I=1}^{M} RGA_I \text{ ...VIII.24}$$

La densidad del aceite a condiciones del tanque, ρ_{ot} se determina a partir de la composición de la fase líquida en el tanque de almacenamiento y la densidad de cada componente medido a condiciones estándar.

$$\rho_{ot} = \frac{\sum_I X_I \rho_{si} PM_I}{\sum_I X_I PM_I} \text{ ...VIII.25}$$

Esta aplicación se puede programar en cualquiera de los lenguajes tratados en este trabajo.

VIII.III.II Ejemplo.

A partir de la siguiente información calcular el valor de la RGA_{total} para un sistema de separación en tres etapas:

$$P_1 = 654.7\left[\,psia\right], T_1 = 140\left[^{o}F\right]$$
$$P_2 = 100\left[\,psia\right], T_2 = 135\left[^{o}F\right]$$
$$P_3 = 14.7\left[\,psia\right], T_3 = 104\left[^{o}F\right]$$
$$\rho_{ot@c.s.} = 0.8436\left[\,{}^{g}\!/_{cm^3}\right]$$
$$PM_t = 203\left[\,{}^{lb}\!/_{lb-mole}\right]$$

$P_k = 3000\left[\,psia\right]$ se supone para obtener los valores de k_i.

La composición de la mezcla que llega al primer separador (1ª etapa) es:

Componente	z_i
H_2S	0.0157
CO_2	0.0214
N_2	0.0037
C_1	0.4921
C_2	0.1038
C_3	0.0594
$i-C_4$	0.0120
$n-C_4$	0.0283
$i-C_5$	0.0121
$n-C_5$	0.0170
C_6	0.0246
C_{7+}	0.2099

Tabla VIII.1 Ejemplo Balance de Materia.

El primer paso es calcular el equilibrio líquido – vapor para cada una de las etapas de separación, en las siguientes tablas se resumen los cálculos de equilibrio líquido – vapor para cada etapa de separación elaborados manualmente, para elaborar la aplicación se utiliza el método de Newton – Raphson tal y como se explico en la sección VIII.II. Asi para la 1ª etapa se tiene:

1^a *Etapa*							
Compenente	z_i	V_{supuesta}	0.5	0.6	0.58	0.588	y_i
		k_i	x_i	x_i	x_i	x_i	
H_2S	0.0157	1.180	0.0144	0.0142	0.0142	0.0142	0.0168
CO_2	0.0214	4.000	0.0086	0.0076	0.0078	0.0077	0.0310
N_2	0.0037	9.300	0.0007	0.0006	0.0006	0.0006	0.0059
C_1	0.4921	5.200	0.1587	0.1398	0.1432	0.1418	0.7375
C_2	0.1038	1.500	0.0830	0.0798	0.0805	0.0802	0.1203
C_3	0.0594	0.640	0.0724	0.0758	0.0751	0.0754	0.0482
$i-C_4$	0.0120	0.320	0.0182	0.0203	0.0198	0.0200	0.0064
$n-C_4$	0.0283	0.250	0.0453	0.0515	0.0501	0.0506	0.0127
$i-C_5$	0.0121	0.132	0.0214	0.0253	0.0244	0.0247	0.0033
$n-C_5$	0.0170	0.110	0.0306	0.0365	0.0351	0.0357	0.0039
C_6	0.0246	0.050	0.0469	0.0572	0.0548	0.0557	0.0028
C_{7+}	0.2099	0.023	0.4104	0.5072	0.4844	0.4933	0.0113
Suma =	**1.0000**		**0.9106**	**1.0158**	**0.9900**	**0.99997**	**1.0000**

Tabla VIII.2 1ª Etapa de Separación.

Los valores de k_i se obtienen considerando $p = 3000[psi]$. La fracción de líquido obtenida después de la primera etapa de separación es la que se considera como z_i en la segunda etapa:

2^a *Etapa*							
Compenente	z_i	V_{supuesta}	0.3	0.2	0.25	0.2853	y_i
		k_i	x_i	x_i	x_i	x_i	
H_2S	0.0142	8.000	0.0046	0.0059	0.0052	0.0047	0.0379
CO_2	0.0077	20.000	0.0012	0.0016	0.0013	0.0012	0.0241
N_2	0.0006	138.000	0.0000	0.0000	0.0000	0.0000	0.0022
C_1	0.1418	30.000	0.0146	0.0209	0.0172	0.0153	0.4588
C_2	0.0802	7.000	0.0286	0.0365	0.0321	0.0296	0.2071
C_3	0.0754	2.600	0.0509	0.0571	0.0538	0.0517	0.1345
$i-C_4$	0.0200	1.150	0.0191	0.0194	0.0193	0.0192	0.0221
$n-C_4$	0.0506	0.900	0.0522	0.0517	0.0519	0.0521	0.0469

$i-C_5$	0.0247	0.400	0.0301	0.0281	0.0291	0.0298	0.0119
$n-C_5$	0.0357	0.330	0.0446	0.0412	0.0428	0.0441	0.0146
C_6	0.0557	0.120	0.0757	0.0676	0.0715	0.0744	0.0089
C_{7+}	0.4933	0.048	0.6905	0.6093	0.6473	0.6772	0.0325
Suma =	**0.99997**		**1.0122**	**0.9392**	**0.9715**	**0.9994**	**1.0014**

Tabla VIII.3 2ª Etapa de Separación.

Para la 3ª etapa se tiene:

3^a *Etapa*							
Compenente	z_i	V_{supuesta}	0.1	0.15	0.13	0.135	y_i
		k_i	x_i	x_i	x_i	x_i	
H_2S	0.0047	32.86	0.0011	0.0008	0.0009	0.0009	0.0294
CO_2	0.0012	86.00	0.0001	0.0001	0.0001	0.0001	0.0083
N_2	0.0000	560.00	0.0000	0.0000	0.0000	0.0000	0.0001
C_1	0.0153	180.00	0.0008	0.0005	0.0006	0.0006	0.1094
C_2	0.0296	37.00	0.0064	0.0046	0.0052	0.0050	0.1868
C_3	0.0517	12.00	0.0246	0.0195	0.0213	0.0208	0.2498
$i-C_4$	0.0192	4.80	0.0139	0.0122	0.0128	0.0127	0.0608
$n-C_4$	0.0521	3.40	0.0420	0.0383	0.0397	0.0394	0.1338
$i-C_5$	0.0298	1.45	0.0285	0.0279	0.0282	0.0281	0.0408
$n-C_5$	0.0441	1.10	0.0437	0.0434	0.0435	0.0435	0.0479
C_6	0.0744	0.37	0.0794	0.0822	0.0811	0.0813	0.0301
C_{7+}	0.6772	0.13	0.7417	0.7788	0.7636	0.7673	0.0998
Suma =	**0.9994**		**0.9824**	**1.0085**	**0.9970**	**0.9998**	**0.9969**

Tabla VIII.4 3ª Etapa de Separación.

En resumen se tiene:

Fracción	1ª *Etapa*	2ª *Etapa*	3ª *Etapa*
V	0.588	0.2853	0.135
L	0.412	0.7147	0.865

Tabla VIII.5 Fracción Líquido - Vapor.

Realizando los cálculos correspondientes para 1ª etapa:

$$L_1 = 1 - V_1 = 1 - 0.588 = 0.412\left[mol_{aceite}\right]$$
$$F_2 = F_1 L_1 = 1 * 0.412 = 0.412$$
$$ng_1 = F_1 V_1 = 1 * 0.588 = 0.588\left[mol_{gas}\right]$$

Para 2ª etapa:

$$L_2 = 1 - V_2 = 1 - 0.2853 = 0.7147\left[mol_{aceite}\right]$$
$$F_3 = F_2 L_2 = 0.412 * 0.7147 = 0.2945$$
$$ng_2 = F_2 V_2 = 0.412 * 0.2853 = 0.1175\left[mol_{gas}\right]$$

Para 3ª etapa:

$$L_3 = 1 - V_3 = 1 - 0.135 = 0.865\left[mol_{aceite}\right]$$
$$F_4 = F_3 L_3 = 0.2945 * 0.865 = 0.2547$$
$$ng_3 = F_3 V_3 = 0.2945 * 0.135 = 0.0398\left[mol_{gas}\right]$$

De la ecuación VIII.16 las moles totales de aceite son:

$$n_t = L_1 L_2 L_3 = 0.412 * 0.7147 * 0.865 = 0.2547\left[mol_{totalac}\right]$$

De la ecuación VIII.20:

$$V_{ot} = \frac{0.2547 * 203}{52.67} = 0.9817\left[pie^3\right]$$

El volumen de gas liberado en cada etapa:

$$VG_1 = 0.588 * 379 = 222.852\left[pie^3\right]$$
$$VG_2 = 0.1175 * 379 = 44.5325\left[pie^3\right]$$
$$VG_3 = 0.0398 * 379 = 15.0842\left[pie^3\right]$$

Por último:

$$RGA_{total} = 5.615\left(\frac{VG_1 + VG_2 + VG_3}{V_{ot}}\right) = 5.615\left(\frac{282.47}{0.9817}\right) = 1615.63\left[\frac{pie^3}{bl}\right]$$

VIII.IV Determinación de Litología y Porosidad por Combinación de los Registros de Porosidad.

Existen una gran cantidad de registros geofísicos que permiten determinan las propiedades petrofísicas de una formación con el fin de caracterizar a un yacimiento. Cabe mencionar que para la interpretación cualitativa y cuantitativa de los registros geofísicos es necesario apoyarse en los resultados obtenidos en análisis geológicos y petrofísicos de muestras de roca. Además en el estudio y la interpretación suele combinarse la información de varios registros, pues la obtenida de uno solo no es suficiente.

VIII.IV.I Registros de Porosidad.

En la siguiente tabla se muestra un breve resumen de las características de este tipo de registros:

Registro	Parámetro Medido	Aplicaciones Principales	Información Necesaria en la Interpretación	Determinaciones en Laboratorio
Sónico Compensado (BHC)	Tiempo de Tránsito	Determinar Porosidad Identificar Litología Realizar Correlaciones Geológicas Estimar Permeabilidad	Relación por - deltat Petrografía Mineralogía	Análisis Petrofísicos para conocer $\Delta t, \phi, k$ Análisis Petrográficos Análisis Mineralógicos
Densidad Compensado (FDC)	Densidad de las Formaciones	Determinar Densidad de la Formación	Densidad Formación ρ_b Matriz ρ_m Fluidos ρ_f Porosidad	Análisis Petrofísicos para conocer ϕ, ρ_b, ρ_m Análisis de Fluidos para conocer ρ_w, ρ_o, ρ_g Análisis Petrográficos Análisis Mineralógicos
Neutrón Compensado (CNL)	Concentración de Hidrogeno en las Formaciones	Determinar Porosidad Determinar Litología Conocer tipo de fluidos existentes en la formación	Litología Petrografía Mineralogía Porosidad ϕ Tipo de Fluido	Análisis Litológico Análisis Petrográficos Análisis Mineralógicos Análisis Petrofísicos para conocer ϕ
Litodensidad (LDT)	Densidad de las Formaciones	Identificar Litología con Base en la Densidad de la Roca Determinar Porosidad	Mineralogía Densidad Matriz ρ_m Densidad Fluidos ρ_f Porosidad ϕ	Análisis Mineralógico Análisis Petrofísico para conocer ϕ, ρ_m Análisis de Fluidos para conocer ρ_w, ρ_o, ρ_g

Tabla VIII.6 Registros de Porosidad.

VIII.IV.II Determinación de Litología y Porosidad.

A partir de la información obtenida por los registros FDC, CNL y BHC se puede establecer el siguiente sistema de ecuaciones lineales:

$$\rho_b = \overbrace{1.0}^{\rho fluido}\,\phi + \overbrace{2.87}^{\rho dolomia}\,V_{dol} + \overbrace{2.71}^{\rho caliza}V_{cal} + \overbrace{2.65}^{\rho arena}V_{ar} \Rightarrow (FDC)\ldots \text{VIII.26}$$

$$\phi_{CNL} = \overbrace{1.0}^{\rho\, fluido}\,\phi + \overbrace{0.02}^{\phi dolomia}\,V_{dol} + \overbrace{0.6}^{\phi caliza}\,V_{cal} - \overbrace{0.035}^{\phi arena}V_{ar} \Rightarrow (CNL)\ldots \text{VIII.27}$$

$$\Delta t = \overbrace{189}^{\Delta tfluido}\,\phi + \overbrace{43.5}^{\Delta tdolomia}\,V_{dol} + \overbrace{47.5}^{\Delta tcaliza}V_{cal} + \overbrace{55.5}^{\Delta tarena}V_{ar} \Rightarrow (BHC)\ldots \text{VIII.28}$$

$$1 = \overbrace{\phi}^{\phi abs} + V_{dol} + V_{cal} + V_{ar} \Rightarrow (BalanceMateria)\ldots \text{VIII.29}$$

Con valores de ρ_b, ϕ_{CNL} y Δt obtenidos de los registros correspondientes, se resuelve el sistema de ecuaciones lineales por alguno de los métodos tratados en este capítulo para así conocer los valores de ϕ, V_{dol}, V_{cal} y V_{ar}.

VIII.V Perfiles de producción, reservas de hidrocarburos, ganancias y número óptimo de pozos.

Cuando se cuenta con poca información acerca del yacimiento se recurre a este tipo de técnicas para pronosticar la producción que se obtendrá, aunque la incertidumbre en los resultados obtenidos es alta sirven para poder tomar mejores decisiones en cuanto a los métodos de explotación que se van a aplicar. De igual forma se puede conocer la cantidad de pozos necesarios para explotar los yacimientos así como la ganancia que se obtendrá con dichos pozos.

Sin embargo cabe señalar que los métodos aquí presentados solo se aplican cuando las condiciones de explotación en cada uno de los pozos no se modifican, es decir, en el momento en que se realicen trabajos en los pozos orientados a mejorar su permeabilidad, disparar un nuevo intervalo de explotación, implementar algún sistema artificial de producción, etc., las predicciones realizadas ya no tendrán validez, y se deberá actualizar la información para obtener una nueva predicción o hacer uso de otros métodos para realizar estas estimaciones. La información necesaria para realizar estos cálculos es la siguiente:

$R_{eo}[bl]$, reservas de crudo.

$R_{eg}\left[pie^3\right]$, reservas de gas.

N, número de pozos.

$o[dl/bl]$, precio de venta del crudo.

$g\left[dl/mpie^3\right]$, precio de venta del gas.

$c[dl/bl]$ o $\left[dl/pie^3\right]$, costo de operación y mantenimiento.

$q_0[bl/d]$, ritmo de producción de crudo inicial.

$q_g\left[pie^3/d\right]$, ritmo de producción de gas inicial.
$d\%$, declinación nominal.
$i\%$, tasa de interés.
$C\left[dl\right]$, costo del pozo.
$D\left[dl\right]$, costo de las obras asociadas al pozo.
n, número de meses.
$k=7$ (base semanal), $k=30.41667$ (base mensual), $k=365$ (base anual).

VIII.V.I Cálculo del perfil de producción de crudo de un pozo.

Esto permite obtener el gasto que producirá un pozo a lo largo del tiempo, para esto es necesario conocer el gasto inicial q_0 y una declinación nominal d en %, el procedimiento es el siguiente:

$$\begin{aligned} q_1 &= q_0\left(1-d\right) \\ q_2 &= q_1\left(1-d\right) \\ &\vdots \\ q_n &= q_{n-1}\left(1-d\right) \end{aligned} \quad \ldots\text{VIII.30}$$

Otra manera es la siguiente:

$$\begin{aligned} q_1 &= q_0 e^{-b} \\ q_2 &= q_1 e^{-b} \\ &\vdots \\ q_n &= q_{n-1} e^{-b} \end{aligned} \quad \ldots\text{VIII.31}$$

En donde:

$$b=-\ln\left(1-d\right) \ldots\text{VIII.32}$$

VIII.V.II Cálculo del volumen mensual de crudo producido.

Una vez que se obtiene el ritmo de producción mensual es posible calcular el volumen que se produciría durante un mes considerando dichos gastos, para esto se hace uso de la siguiente ecuación:

$$\begin{aligned} V_1 &= \frac{\left(30.41667\right)q_0}{b}d \\ V_2 &= \frac{\left(30.41667\right)q_1}{b}d \\ V_3 &= \frac{\left(30.41667\right)q_2}{b}d \\ &\vdots \\ V_n &= \frac{\left(30.41667\right)q_{n-1}}{b}d \end{aligned} \quad \ldots\text{VIII.33}$$

El valor de b esta dado por la ecuación VIII.32.

VIII.V.III Cálculo del volumen de crudo entre t=0 y t=n.

En este caso la siguiente expresión permite obtener el volumen para un tiempo de n meses:

$$V = \frac{(30.41667)q_0}{b}\left[1-e^{-bn}\right] \ldots \text{VIII.34}$$

VIII.V.IV Cálculo del perfil de producción de gas de un pozo, volumen mensual de gas producido y volumen de gas entre t=0 y t=n.

Las ecuaciones empleadas para la predicción en el caso del gas, son las mismas que se emplearon anteriormente, la única diferencia radica en que el gasto de aceite (q_0) es reemplazado por el gasto de gas (q_g) en $\left[pie^3/d\right]$.

VIII.V.V Reserva de hidrocarburos.

Para el cálculo de las reservas es necesario realizar todos los cálculos anteriores para el crudo y gas, de esta forma se podrán sumar ambas reservas para obtener el valor total de estas.

Las expresiones utilizadas son:

$$R_{aceite} = \frac{(30.41667)q_o}{b_o}\left[barriles\right] \ldots \text{VIII.35}$$

$$b_o = -\ln(1-d_o) \ldots \text{VIII.32}$$

$$R_{gas} = \frac{(30.41667)q_g}{b_g}\left[pies^3\right] \ldots \text{VIII.36}$$

$$b_g = -\ln(1-d_g) \ldots \text{VIII.32}$$

Las ecuaciones VIII.35 y VIII.36 no pueden sumarse entre sí, pues una se refiere a crudo y otra gas, por lo que es necesario llevar el volumen de gas a su equivalente en barriles de petróleo crudo equivalente $\left[bpce\right]$.

$$R_{TOTAL} = \frac{(30.41667)q_o}{b_o} + \frac{(30.41667)q_g}{(5000)b_g}\left[bpce\right] \ldots \text{VIII.37}$$

VIII.V.VI Ganancia con un pozo o con N pozos de crudo.

Para calcular la ganancia de explotar un pozo es necesario conocer información como el gasto, la declinación, el precio por barril, los costos del pozo y obras, así como tasas de interés.

Se puede calcular la ganancia que se obtendrá de un solo pozo de aceite por medio de la ecuación:

$$G = \frac{(o-c)(kq_o)}{(b+i)}\left[1-e^{-(b+i)n}\right]-(C+D) \ldots \text{VIII.38}$$

O para valores grandes de n o $(b+i)n$:

$$G = \frac{(o-c)(kq_o)}{(b+i)}-(C+D) \ldots \text{VIII.39}$$

En el caso de que se tenga un número N de pozos la ganancia se calcula mediante las siguientes expresiones:

$$G = \frac{(o-c)(kq_o)}{(b+i)}N-(C+D)N \ldots \text{VIII.40}$$

En donde:

$$b = \frac{(kq_o)N}{R_{eo}} \ldots \text{VIII.41}$$

VIII.V.VII Número óptimo de pozos de crudo.

Dependiendo de las características del yacimiento, la producción esperada, el costo de operación y mantenimiento, el precio de venta del crudo, etc., se puede calcular el número óptimo de pozos para explotar dicho yacimiento mediante la siguiente expresión:

$$N = \frac{R_{eo}}{365q_o}\left[\sqrt{\frac{(o-c)(365q_o)i}{C+D}}-i\right] \ldots \text{VIII.42}$$

VIII.V.VIII Ganancia con N Pozos de Gas.

De igual forma que con el crudo se puede calcular la ganancia para un conjunto N de pozos de gas mediante la siguiente ecuación:

$$G = \frac{\left[\frac{g-c}{1000}\right](365q_o)}{b+i}N-(C+D)N \ldots \text{VIII.43}$$

$$b = \frac{\left(kq_g\right)N}{R_{eg}} \text{ ...VIII.41}$$

VIII.V.IX Número óptimo de pozos de gas.

Considerando las mismas variables que en el caso de los pozos de crudo, el número óptimo de pozos de gas esta dado por:

$$N = \frac{R_{eg}}{365q_g}\left[\sqrt{\frac{0.001\left(g-c\right)\left(365q_g i\right)}{C+D}} - i\right]$$

VIII.V.X Ejemplo.

Considerando los siguientes datos:

Datos	
$q_o\left[bl/d\right]$	3,750.00
d% mensual	2.00
$q_g\left[pie^3/d\right]$	7,750,000.00
d% mensual	3.00
$o\left[dl/bl\right]$	25.00
$g\left[dl/mpie^3\right]$	3.00
$c\left[dl/bl\right]$	4.25
$C\left[dl\right]$	2,500,000.00
$D\left[dl\right]$	450,000.00
i% anual	12.00
n	240.00
k	30.42

Tabla VIII.7 Ejemplo Perfiles de Producción.

Obtener el perfil de producción, las reservas, ganancias y número óptimo de pozos.

Aplicando la ecuación VIII.32 para obtener b_o y b_g:

$$b_o = -\ln(1-0.02)$$
$$b_o = 0.02$$
$$b_g = 0.03$$

Con la ecuación VIII.30 se calculan los gastos para los meses deseados:

$$q_{o1} = 3,750\left(1-0.02\right) = 3,675$$
$$q_{o2} = 3,675\left(1-0.02\right) = 3,602$$
$$q_{o3} = 3,602\left(1-0.02\right) = 3,530$$
$$\vdots$$

$$q_{g1} = 7,750,000\left(1-0.03\right) = 7,517,500$$
$$q_{g2} = 7,517,500\left(1-0.03\right) = 7,291,975$$
$$q_{g3} = 7,291,975\left(1-0.02\right) = 7,073,216$$
$$\vdots$$

De igual forma haciendo uso de la ecuación VIII.31:

$$q_{o1} = 3,750e^{-0.02} = 3,675$$
$$q_{o2} = 3,675e^{-0.02} = 3,602$$
$$q_{o3} = 3,602e^{-0.02} = 3,530$$
$$\vdots$$

Ambas ecuaciones proporcionan aproximadamente los mismos resultados, las diferencias están solamente en decimales que se pueden despreciar, en la siguiente tabla se resumen algunos de los resultados obtenidos:

Mes	$Gasto\left(q_o\left[bl/d\right]\right)$	$Gasto\left(q_g\left[pie^3/d\right]\right)$
0	3,750	7,750,000
1	3,675	7,517,500
2	3,602	7,291,975
3	3,529	7,073,216
4	3,459	6,861,019
5	3,390	6,655,189
6	3,322	6,455,533
7	3,255	6,261,867
8	3,190	6,074,011
9	3,127	5,891,791
10	3,064	5,715,037
⋮	⋮	⋮
230	36	7,027

231	35	6,817
232	35	6,612
233	34	6,414
234	33	6,221
235	33	6,035
236	32	5,854
237	31	5,678
238	31	5,508
239	30	5,343
240	29	5,182

Tabla VIII.8 Perfil de Producción.

Graficando esta información se puede ver el perfil de producción de este pozo durante 240 meses y considerando una declinación constante:

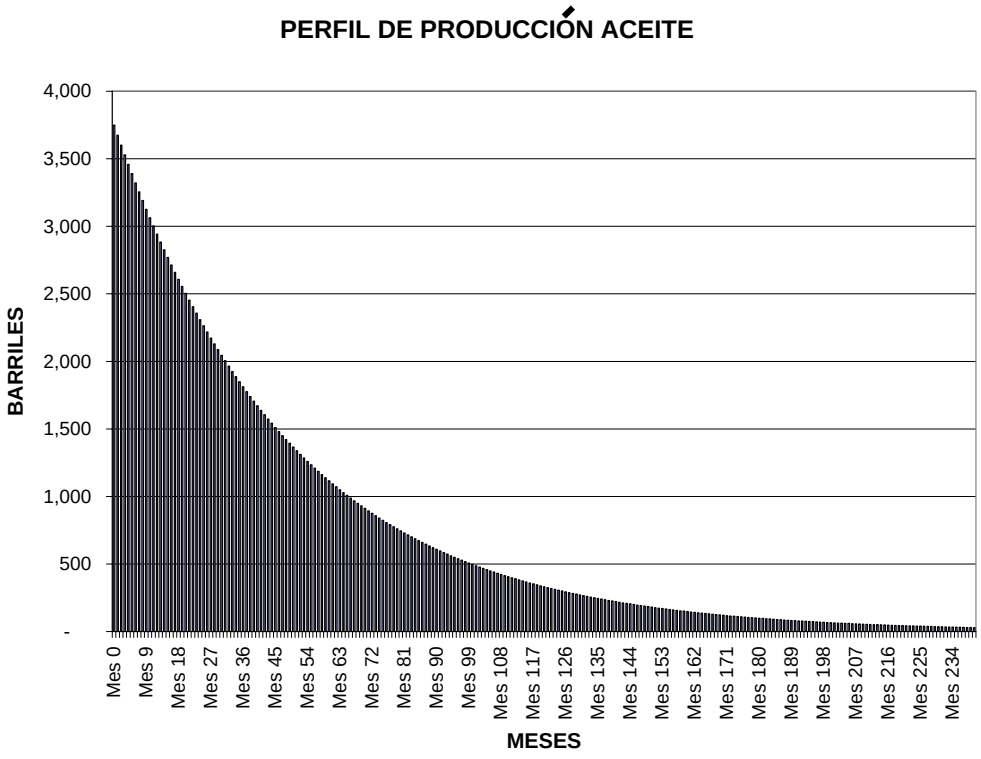


Figura VIII.2 Perfil de Producción de Aceite.

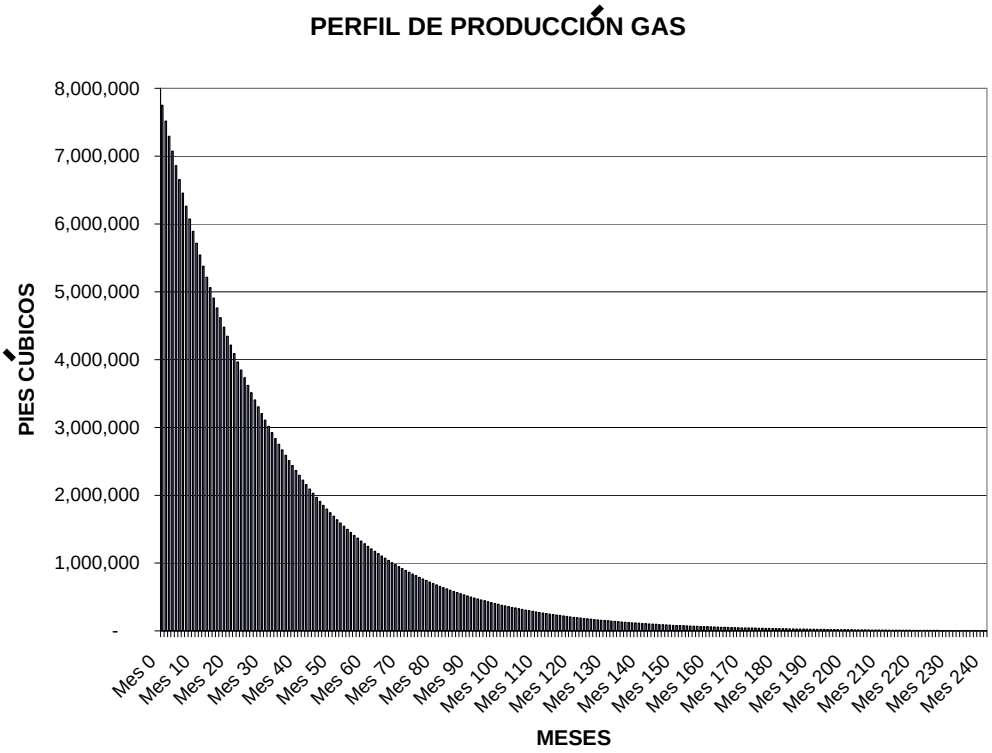


Figura VIII.3 Perfil de Producción de Gas.

Los resultados para los otros puntos se resumen en la siguiente tabla:

Resultados	
$R_{eo}\left[bl\right]$	5,645,902.34
$R_{eg}\left[\ pies^3\ \right]$	7,739,176,819.30
$R_{eg}\left[bpce\right]$	1,547,835.36
$R_{total}\left[bpce\right]$	7,193,737.70
N_{pozos}	4
$G_{1\,pozo}\ \ mensual$	301,211,508.14
$G_{Npozos}\ \ mensual$	122,550,793.11

Tabla VIII.10 Resultados Perfil de Producción.

Esta aplicación es sencilla de programar, pues como se puede apreciar no se requiere de ningún tipo de solución numérica, y se puede hacer uso de cualquiera de los lenguajes de programación tratados en este trabajo, sin embargo se recomienda el uso de de VISUAL BASIC 6.0® o de MATLAB 7.0® pues brindan capacidades para presentar las gráficas utilizadas.

VIII.VI Pruebas de incremento.

Se analizan para conocer las propiedades del yacimiento y las condiciones del pozo, tienen la desventaja de que para realizarlas se tiene que cerrar el pozo, afectando directamente a la producción, adicionalmente existe dificultad para mantener gasto (q) constante antes del cierre.

VIII.VI.I Algoritmo.[13]

A partir de la siguiente ecuación:

$$\Delta P(t) = \frac{q_1 \mu B}{2\pi kh} P_D(t) + \frac{(q_2 - q_1)\mu B}{2\pi kh} P_D(t - t_1) \ldots \text{VIII.30}$$

Considerando $q_1 = q$, $q_2 = 0$, $t - t_1 = \Delta t = t - t_p$, $t = t_p + \Delta t$:

$$\Delta P(t_p + \Delta t) = \frac{q\mu B}{2\pi kh} P_D(t_p + \Delta t) - \frac{q\mu B}{2\pi kh} P_D(\Delta t) = \frac{q\mu B}{2\pi kh}\left[P_D(t_p + \Delta t) - P_D(\Delta t)\right] \ldots \text{VIII.31}$$

Usando la aproximación logarítmica $\log_{10}$ y considerando $\Delta P = P_i - P_{ws}$, donde p_{ws} es la presión durante el cierre.

$$P_{ws} = P_i - 162.6 \frac{q\mu B}{kh} \log \frac{t_p + \Delta t}{\Delta t} \ldots \text{VIII.32}$$

Reescribiendo esta ecuación:

$$P_{ws} = P_i - m \log \frac{t_p + \Delta t}{\Delta t} \ldots \text{VIII.33}$$

La gráfica de la ecuación anterior es una línea recta con intersección P_i y pendiente $-m$, donde:

$$m = \frac{162.6 qB\mu}{kh} \ldots \text{VIII.34}$$

Despejando la permeabilidad:

$$k = \frac{162.6 qB\mu}{mh} \ldots \text{VIII.35}$$

VIII.VI.II Método de Horner.[13]

La gráfica de $P_{ws} vs\left(tp+\Delta t/\Delta t\right)$ en escala semilogarítmica, se conoce como gráfica de Horner; y al método involucrado se le conoce como método de Horner.

Para efectos numéricos se puede graficar $P_{ws} vs \log\left[\left(tp+\Delta t/\Delta t\right)\right]$ en escala normal.

Una vez que se grafican los puntos, se ajusta una línea recta, para esto se deben despreciar los puntos que representan el efecto de almacenamiento.

En la siguiente figura se muestra la sección de la línea recta. Esta puede ser extrapolada a $\left(tp+\Delta t/\Delta t\right)=1$, $\log\left[\left(tp+\Delta t/\Delta t\right)\right]=0$, esto equivale a un tiempo de cierre infinito, para obtener una estimación de P_i.

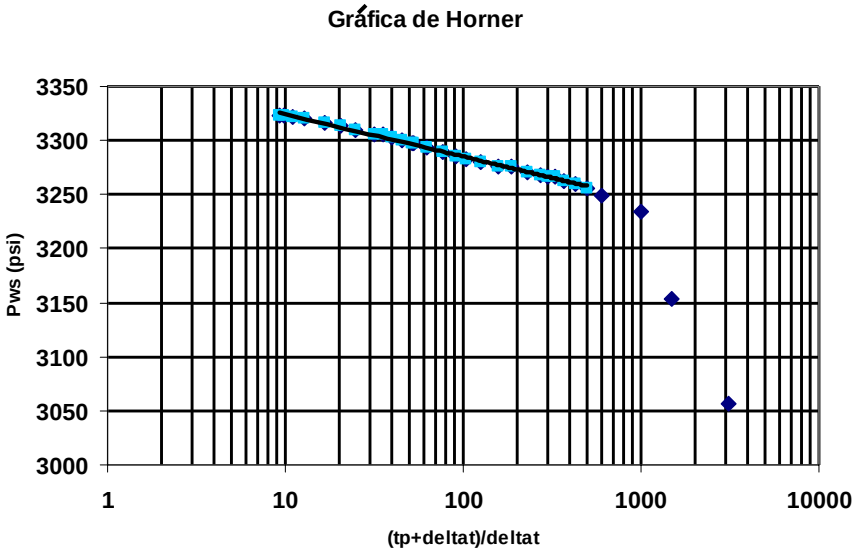


Figura VIII.4 Gráfica de Horner.

El factor de daño se calcula con la siguiente expresión:

$$S=1.1513\left[\frac{P_{1hr}-P_{wf\,(\Delta t=0)}}{m}-\log\frac{k}{\phi\mu c r_w^2}+3.2275\right]\ldots\text{VIII.36}$$

Donde $P_{wf}\ (\Delta t=0)$ es la presión de fondo fluyendo inmediatamente antes del cierre. La P_{1hr} $(P_{ws}\Delta t=1hr)$ debe ser tomada de la línea recta de la gráfica de Horner. Cuando los datos de incremento no caen sobre la línea recta a 1 hr. , ésta deberá de ser extrapolada para 1 hora.

VIII.VI.II.I Ejemplo.[13]

En la tabla se muestran los datos de incremento de presión para un pozo de aceite, con un radio de drene de $2{,}640[pies]$. Antes del cierre estaba produciendo a un gasto estabilizado de $4{,}900[Bl/día]$ por un periodo de 310 horas.

i	Δt	$tp+\Delta t$	$Log((tp+\Delta t)/\Delta t)$	$(tp+\Delta t)/\Delta t$	p_{ws}	$p_{ws}-p_{wf}$
1	0.00	--------	--------	--------	2761	0
2	0.10	310.10	3.4915018	3101.0	3057	296
3	0.21	310.21	3.1694365	1477.2	3153	392
4	0.31	310.31	3.0004341	1001.0	3234	473
5	0.52	310.52	2.7760862	597.2	3249	488
6	0.63	310.63	2.6929028	493.1	3256	495
7	0.73	310.73	2.6290603	425.7	3260	499
8	0.84	310.84	2.5682576	370.0	3263	502
9	0.94	310.94	2.5195487	330.8	3266	505
10	1.05	311.05	2.4716409	296.2	3267	506
11	1.15	311.15	2.4322720	270.6	3268	507
12	1.36	311.36	2.3597239	228.9	3271	510
13	1.68	311.68	2.2683997	185.5	3276	515
14	1.99	311.99	2.1952876	156.8	3276	515
15	2.51	312.51	2.0951902	124.5	3280	519
16	3.04	313.04	2.0127263	103.0	3283	522
17	3.46	313.46	1.9571060	90.6	3286	525
18	4.08	314.08	1.8863801	77.0	3289	528
19	5.03	315.03	1.7967839	62.6	3293	532
20	5.97	315.97	1.7236715	52.9	3297	536
21	6.07	316.07	1.7165946	52.1	3297	536
22	7.01	317.01	1.6553549	45.2	3300	539
23	8.06	318.06	1.5961740	39.5	3303	542
24	9.00	319.00	1.5495482	35.4	3305	544
25	10.05	320.05	1.5030518	31.8	3306	545
26	13.09	323.09	1.3923839	24.7	3310	549
27	16.02	326.02	1.3085817	20.4	3313	552
28	20.00	330.00	1.2174839	16.5	3317	556
29	26.07	336.07	1.1102887	12.9	3320	559
30	31.03	341.03	1.0410108	11.0	3322	561
31	34.98	344.98	0.9939741	9.9	3323	562
32	37.54	347.54	0.9665105	9.3	3323	562

Tabla VIII.11 Ejemplo Método de Horner.

Los datos del yacimiento son:

$$profundidad = 10{,}476[pies] \qquad \mu_o = 0.2[cp]$$
$$r_w = 4.5[pg] \qquad \phi = 0.09$$
$$c_t = 22.6 * 10^{-6}[psia^{-1}] \qquad B_o = 1.55[Bl/Bl]$$
$$q_o = 4{,}900[Bl/día] \qquad TR_{Di} = 6.276[pg]$$
$$h = 482[pies] \qquad \overline{p} = 3{,}342[psi]$$
$$P_{wf(\Delta t=0)} = 2761[psi]$$

Calcular k y s.

Las gráficas que se obtienen son:

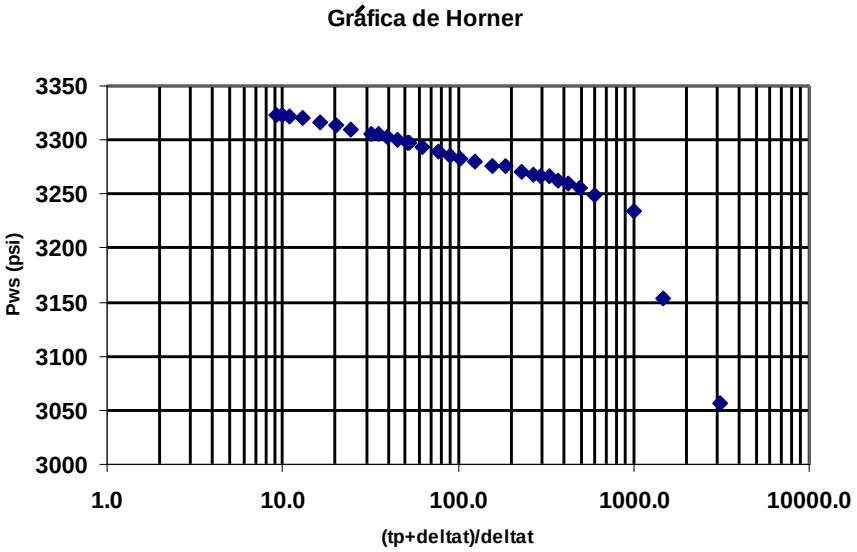


Figura VIII.5 Ejemplo Gráfica de Horner Semilog.

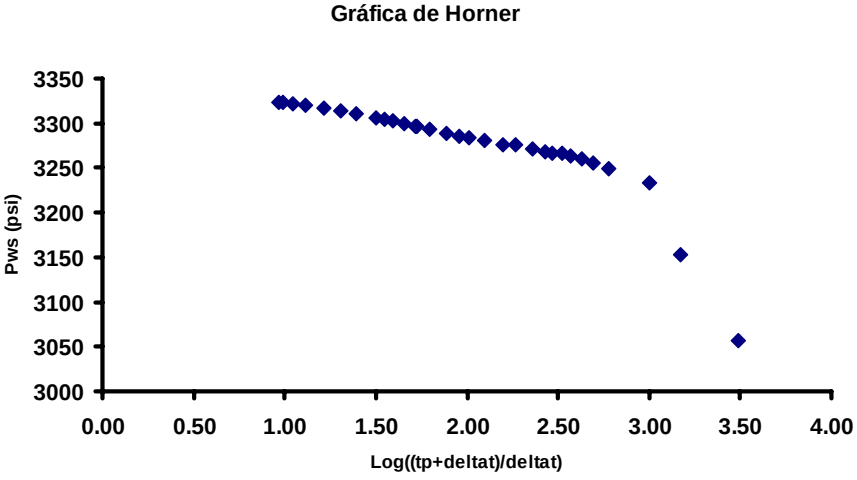


Figura VIII.6 Ejemplo Gráfica de Horner Normal.

Utilizando la gráfica $P_{ws} vs Log\left[\left(tp+\Delta t / \Delta t\right)\right]$ y despreciando los primeros cuatro datos que representan el almacenamiento, además, haciendo uso del método de regresión lineal se puede ajustar la recta necesaria para obtener la pendiente m y así obtener los datos requeridos. En la siguiente tabla se resumen los cálculos referentes a la regresión lineal:

i	$A_{1,2}, A_{2,1}$	Z_1	$A_{2,2}$	Z_2
	$x_i = Log\left((tp+\Delta t)/\Delta t\right)$	$y_i = p_{ws}$	x_i^2	$x_i y_i$
5	2.7760862	3249	7.7067	9019.504174
6	2.6929028	3256	7.2517	8768.091671
7	2.6290603	3260	6.9120	8570.736659
8	2.5682576	3263	6.5959	8380.224595
9	2.5195487	3266	6.3481	8228.846187
10	2.4716409	3267	6.1090	8074.850842
11	2.4322720	3268	5.9159	7948.664782
12	2.3597239	3271	5.5683	7718.65691
13	2.2683997	3276	5.1456	7431.277265
14	2.1952876	3276	4.8193	7191.76217
15	2.0951902	3280	4.3898	6872.223847
16	2.0127263	3283	4.0511	6607.780283
17	1.9571060	3286	3.8303	6431.050416
18	1.8863801	3289	3.5584	6204.304212
19	1.7967839	3293	3.2284	5916.809475
20	1.7236715	3297	2.9710	5682.944998
21	1.7165946	3297	2.9467	5659.612348
22	1.6553549	3300	2.7402	5462.671316
23	1.5961740	3303	2.5478	5272.162764
24	1.5495482	3305	2.4011	5121.256714
25	1.5030518	3306	2.2592	4969.089151
26	1.3923839	3310	1.9387	4608.790609
27	1.3085817	3313	1.7124	4335.331276
28	1.2174839	3317	1.4823	4038.394243
29	1.1102887	3320	1.2327	3686.158533
30	1.0410108	3322	1.0837	3458.237909
31	0.9939741	3323	0.9880	3302.975977
32	0.9665105	3323	0.9341	3211.714486
Suma	***52***	***92119***	***107***	***172174***

Tabla VIII.12 Ejemplo Método de Horner Regresión Lineal.

Haciendo uso de las ecuaciones:

$$a = \frac{A_{2,2}Z_1 - A_{1,2}Z_2}{d} \ldots \text{IV.32}$$

$$b = \frac{A_{1,1}Z_2 - A_{2,1}Z_1}{d} \ldots \text{IV.33}$$

$$d = A_{1,1}A_{2,2} - A_{1,2}A_{2,1} \ldots \text{IV.34}$$

Se obtiene:

$$a = 3{,}364.8$$
$$b = -40.0$$
$$d = 237.19$$

Entonces la recta de ajuste es:

$$P_{ws} = -40\left(Log\left(tp + \Delta t/\Delta t\right)\right) + 3364.8$$

La gráfica que representa este comportamiento es:

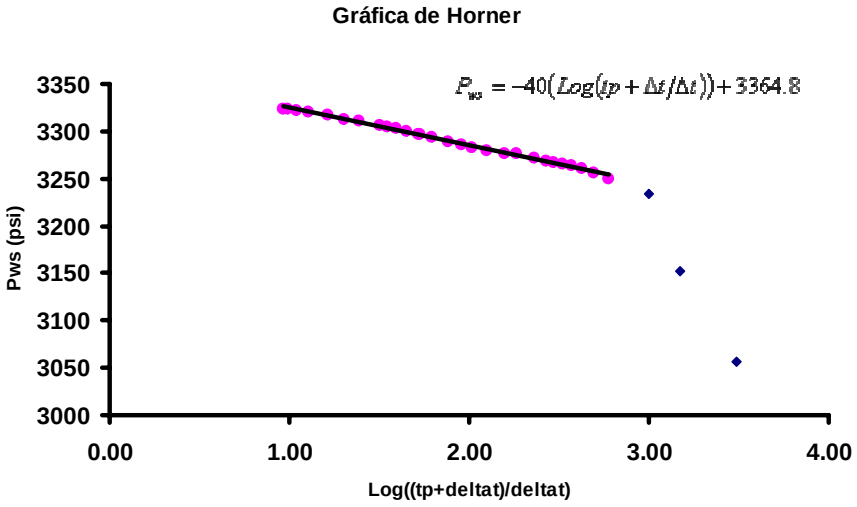


Figura VIII.7 Ejemplo Gráfica de Horner Recta Ajustada.

La pendiente se define como $-m$, entonces en este caso se tiene $-m = -40 \Rightarrow m = 40$, entonces se puede calcular la permeabilidad y el daño:

$$k = \frac{162.6(4900)(1.55)(0.2)}{40(482)} = 12.8[md]$$

$$s = 1.1513\left[\frac{3266 - 2761}{40} - Log\left(\frac{(12.8)(12)^2}{(0.09)(0.20)(22.6*10^{-6})(4.25)^2}\right) + 3.2275\right] = 8.6$$

Para elaborar el programa se puede hacer uso de cualquiera de los lenguajes de programación tratados en este trabajo, sin embargo se recomienda el uso de de VISUAL BASIC 6.0® o de MATLAB 7.0® pues brindan capacidades para presentar las gráficas utilizadas.

VIII.VII Elipse de esfuerzos triaxiales.

La elipse de esfuerzos triaxiales es una herramienta utilizada durante el diseño de las tuberías de producción y revestimiento de un pozo, sin embargo es la culminación de una serie de conceptos referentes a dichas tuberías, en esta sección se abordaran de manera breve y simplificada dichos conceptos para así poder generar la elipse de esfuerzos triaxiales.

VIII.VII.I Tuberías.

Las tuberías se clasifican de acuerdo a propiedades como el proceso de fabricación, el grado del acero, tipo de juntas, rango de longitud y espesor de pared y peso.

El API (American Petroleum Institute) ha desarrollado una clasificación para tuberías utilizadas en la industria petrolera. La tubería de revestimiento se define como un tubo, con un rango de diámetros externos de 4.5 a 20 pulgadas, mientras que una tubería de producción se encuentra entre 1.0 y 4.5 pulgadas.

El API designó un grado de tubería para definir las características de resistencia de la tubería. El código de los grados consiste en una letra seguida de un número, la designación de la letra fue hecha arbitrariamente para signar un valor único para cada grado de tubería de revestimiento dentro de los estándares. El número designa el esfuerzo de cedencia en miles de $\left[psi\right]$, y se define como la fuerza de tensión requerida para producir una elongación total por unidad de longitud de 0.5% de la longitud, excepto para la tubería $P-110$ donde es definido como la fuerza necesaria para producir una elongación de 0.6%.[20, 21]

VIII.VII.II Fuerzas de diseño de las tuberías.

Las más importantes propiedades mecánicas de la tubería de producción y de la tubería de revestimiento son esfuerzo (presión) de ruptura, esfuerzo (presión) de colapso y esfuerzo de tensión.

VIII.VII.II.I Esfuerzo de ruptura.

Cuando la tubería de revestimiento esta expuesta a una presión interna mucho mayor que la externa se dice que la tubería esta expuesta a una presión de ruptura (Burst).

$$P = 0.875\left[\frac{2\sigma_p t}{D}\right] \ldots \text{VIII.37}$$

$P\left[lb/pg^2\right]$; Presión Interna Mínima de Cedencia.

$\sigma_p\left[\frac{lb}{pg^2}\right]$; Mínimo Esfuerzo de Cedencia.

$t[pg]$; Espesor Nominal.

$D[pg]$; Diámetro Externo.

El API recomienda usar esta ecuación con espesores de pared redondeados a la milésima más próxima $0.001[pg]$ y resultados redondeados a la decena más próxima $10\left[lb/pg^2\right]$.[20]

VIII.VII.II.II Esfuerzo de Colapso.

Cuando la tubería de revestimiento esta expuesta a una presión externa mucho mayor que la interna se dice que la tubería esta expuesta a una presión de colapso. El esfuerzo de colapso es primordialmente una función del esfuerzo de cedencia del material, de su relación de espesor D/t y de su esfericidad.

$$P_{\sigma_p} = 2\sigma_p\left[\frac{(D/t)-1}{(D/t)^2}\right] \ldots \text{VIII.38}$$

La relación (D/t) se conoce como rango.

VIII.VII.II.III Esfuerzo de Tensión (Axial).

La fuerza axial de cedencia es determinada por el esfuerzo de cedencia de la tubería:

$$F_y = \frac{\pi}{4}\left(D^2 - d^2\right)\sigma_p \ldots \text{VIII.39}$$

Todas las ecuaciones utilizadas para calcular las diferentes resistencias de las tuberías, consideran la aplicación de esfuerzos axiales. La realidad es diferente, ya que la tubería siempre estará expuesta a una combinación de esfuerzos. El concepto fundamental para el diseño de tuberías establece *"Si cualquier esfuerzo aplicado a la pared de la tubería excede el esfuerzo de cedencia del material, se presentará una condición de falla "*.[21]

VIII.VII.III Esfuerzo Triaxial.

El esfuerzo triaxial no es un esfuerzo verdadero, es un valor teórico de la resultante de esfuerzos en tres dimensiones el cual es comparado con el criterio uniaxial de falla. Si el esfuerzo triaxial excede al esfuerzo de cedencia del material se presentará la falla.

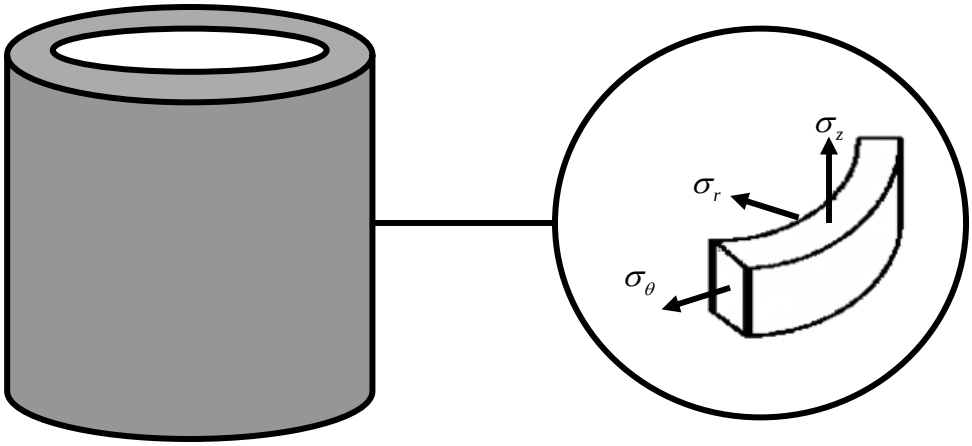


Figura VIII.8 Esfuerzo Triaxial.

El esfuerzo triaxial se obtiene mediante:

$$\sigma_T = \frac{1}{\sqrt{2}}\sqrt{\left[\left(\sigma_z - \sigma_\theta\right)^2 + \left(\sigma_\theta - \sigma_r\right)^2 + \left(\sigma_r - \sigma_z\right)^2\right]} \ldots \text{VIII.40}$$

σ_T Esfuerzo Triaxial.
σ_z Esfuerzo Axial.
σ_θ Esfuerzo Tangencial.
σ_r Esfuerzo Radial.

VIII.VII.III.I Algoritmo.
El esfuerzo triaxial se calcula mediante la siguiente ecuación:

$$\sigma_T = \sqrt{\left(f_1 f_2\right) + f_3} \ldots \text{VIII.41}$$

En donde:

$$f_1 = \left(\frac{r_i}{r}\right)^2 \frac{\sqrt{3}}{2}\left(P_0 - P_i\right) \ldots \text{VIII.42}$$

$$f_2 = \frac{1}{2}\frac{\left(\frac{D}{t}\right)^2}{\left(\frac{D}{t}\right) - 1} \ldots \text{VIII.43}$$

$$f_3 = \sigma_z - \frac{r_i^2 P_i - r_0^2 P_0}{r_0^2 - r_i^2} \ldots \text{VIII.44}$$

P_0 Presión Colapso.
P_i Presión Interna.

Considerando $\sigma_T = \sigma_p$, $r = r_i$, $P_i = 0$ y desarrollando la ecuación VIII.41:

$$\sigma_p^2 = \left(\frac{\sqrt{3}}{2} P_0 f_2 \right)^2 + \left(\sigma_z + W_1 \right)^2 \ldots \text{VIII.45}$$

Despejando σ_z:

$$\sigma_z = -W_1 P_0 + \sqrt{\sigma_p^2 - \left(W_2 P_0 \right)^2} \ldots \text{VIII.46}$$

Considerando P_i:

$$\sigma_z = -W_3 P_i + \sqrt{\sigma_p^2 - \left(W_2 P_i \right)^2} \ldots \text{VIII.47}$$

En donde:

$$W_1 = \frac{r_0^2}{r_0^2 - r_i^2} \ldots \text{VIII.48}$$

$$W_2 = \frac{\sqrt{3}}{2} f_2 \ldots \text{VIII.49}$$

$$W_3 = \frac{r_i^2}{r_0^2 - r_i^2} \ldots \text{VIII.50}$$

Considerando P_i y P_0 iguales a cero se tiene:

$$\sigma_z = -\overbrace{W_3 P_i}^{0} + \sqrt{\sigma_p^2 - \overbrace{\left(W_2 P_i \right)^2}^{0}} \text{ y } \sigma_z = -\overbrace{W_1 P_0}^{0} + \sqrt{\sigma_p^2 - \overbrace{\left(W_2 P_0 \right)^2}^{0}}$$

$$\sigma_z = \pm \sqrt{\sigma_P^2}$$

$$\sigma_z = \pm \sigma_p \ldots \text{VIII.51}$$

La presión al colapso P_0 se obtiene considerando $\sigma_z = 0$, y despejando de la ecuación VIII.46 se tiene:

$$P_0 = \sqrt{\frac{\sigma_P^2}{W_1^2 + W_2^2}} \ldots \text{VIII.52}$$

Las ecuaciones VIII.46 y VIII.47 representan una elipse como la mostrada en la figura y se conoce como elipse de esfuerzos triaxiales en donde P_0 se encuentra en la parte negativa de la gráfica y P_i en la parte positiva, las intersecciones positiva y negativa con el eje de presiones (y) están dadas por $\sqrt{\frac{\sigma_P^2}{W_1^2+W_2^2}}$ .y $-\sqrt{\frac{\sigma_P^2}{W_1^2+W_2^2}}$ respectivamente, y las intersecciones positiva y negativa con el eje de esfuerzos están dadas por σ_p y $-\sigma_p$ que esta indicado por el grado de la tubería.

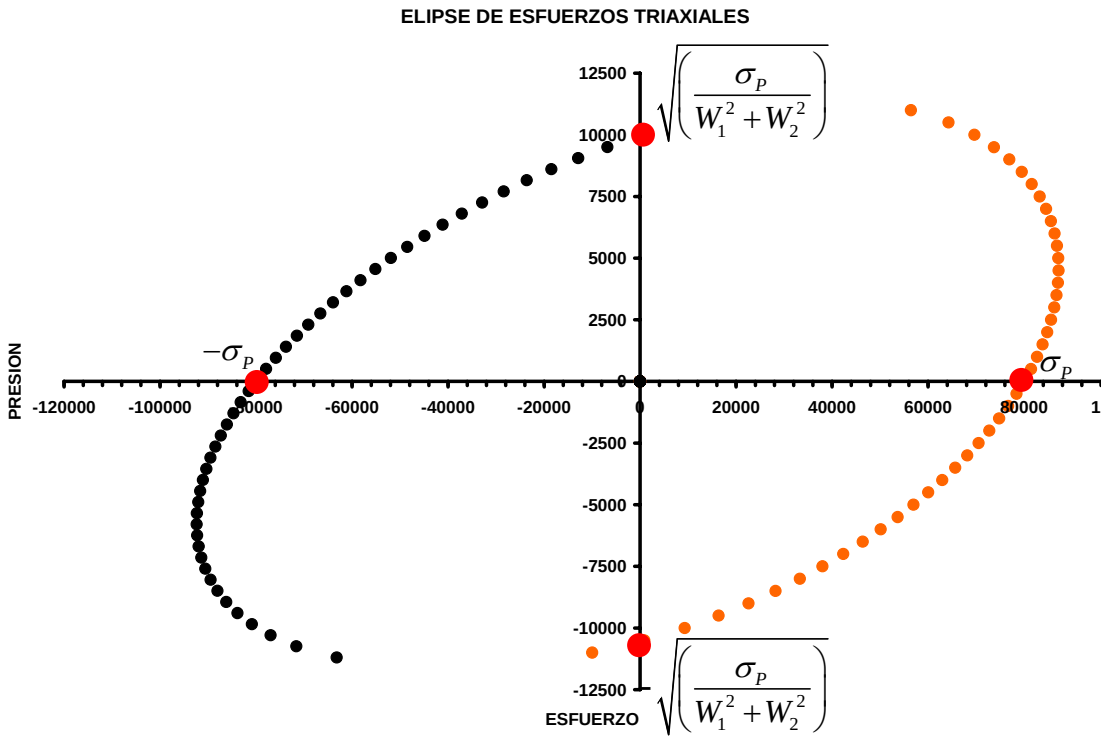


Figura VIII.9 Elipse de Esfuerzos Triaxiales.

Las partes en que no cierra la elipse ocurren cuando el valor de $\frac{\sigma_P^2}{W_1^2+W_2^2}$ es negativo.

VIII.VII.III.II Ejemplo.

A partir de la siguiente información graficar la elipse de esfuerzos triaxiales para una tubería N-80.

$Tubería \;\; N-80$	
$\sigma_P\left[lb/pg^2\right]$	80,000
$D[pg]$	4.5
$d_i[pg]$	3.9
$r_0[pg]$	2.25
$r_i[pg]$	1.95

Tabla VIII.13 Ejemplo Elipse de Esfuerzos Triaxiales.

Calculando el espesor y el rango:

$$t=\frac{(D-d_i)}{2}=\frac{\left(4.5[pg]-3.9[pg]\right)}{2}=0.3[pg]$$

$$\left(\frac{D}{t}\right)=\frac{4.5[pg]}{0.3[pg]}=15$$

Con estos datos se pueden calcular los valores de f_2, W_1, W_2 y W_3:

$$f_2=\frac{1}{2}\frac{15^2}{15-1}=8.035$$

$$W_1=\frac{2.25^2}{2.25^2-1.95^2}=4.017$$

$$W_2=\frac{\sqrt{3}}{2}(8.035)=6.958$$

$$W_3=\frac{1.95^2}{2.25^2-1.95^2}=3.017$$

Con estos valores y las ecuaciones VIII.46 y VIII.47 :se pueden calcular los diferentes puntos que componen la elipse de esfuerzos triaxiales, para estos se debe considerar un valor inicial para P_i, se recomienda comenzar con un valor de $1500\left[lb/pg^2\right]$ y comenzar a disminuir este en algún valor que permita definir la elipse, por ejemplo de $500\left[lb/pg^2\right]$ y los valores de P_0 se calculan mediante la ecuación VIII.52. En la siguiente tabla se resumen los resultados obtenidos para realizar la gráfica solicitada.

PI	Sz(+)=	Po	Sz(-)=	PI	Sz(+)=	Po	Sz(-)=
15000	0.000	9,955.556	-0.056	**-500**	78,415.364	-3,994.444	-91,064.300
14500	0.000	9,505.556	-6,798.320	**-1000**	76,678.884	-4,444.444	-91,636.329
14000	0.000	9,055.556	-12,897.233	**-1500**	74,789.250	-4,894.444	-92,051.941
13500	0.000	8,605.556	-18,466.790	**-2000**	72,744.244	-5,344.444	-92,301.913
13000	0.000	8,155.556	-23,613.040	**-2500**	70,540.666	-5,794.444	-92,375.039
12500	0.000	7,705.556	-28,407.434	**-3000**	68,174.236	-6,244.444	-92,257.619
12000	0.000	7,255.556	-32,900.793	**-3500**	65,639.457	-6,694.444	-91,932.730
11500	0.000	6,805.556	-37,130.742	**-4000**	62,929.422	-7,144.444	-91,379.193
11000	56,434.913	6,355.556	-41,125.997	**-4500**	60,035.552	-7,594.444	-90,570.043
10500	64,255.039	5,905.556	-44,908.998	**-5000**	56,947.245	-8,044.444	-89,470.170
10000	69,638.009	5,455.556	-48,497.607	**-5500**	53,651.389	-8,494.444	-88,032.546
9500	73,716.675	5,005.556	-51,906.244	**-6000**	50,131.680	-8,944.444	-86,191.772
9000	76,932.275	4,555.556	-55,146.675	**-6500**	46,367.657	-9,394.444	-83,852.234
8500	79,512.408	4,105.556	-58,228.575	**-7000**	42,333.279	-9,844.444	-80,863.974
8000	81,592.924	3,655.556	-61,159.930	**-7500**	37,994.770	-10,294.444	-76,965.780
7500	83,262.627	3,205.556	-63,947.345	**-8000**	33,307.210	-10,744.444	-71,615.223
7000	84,583.279	2,755.556	-66,596.266	**-8500**	28,208.837	-11,194.444	-63,171.935
6500	85,599.800	2,305.556	-69,111.150	**-9000**	22,610.846	0.000	0.000
6000	86,345.966	1,855.556	-71,495.597	**-9500**	16,377.390	0.000	0.000
5500	86,847.818	1,405.556	-73,752.448	**-10000**	9,280.866	0.000	0.000
5000	87,125.817	955.556	-75,883.858	**-10500**	880.039	0.000	0.000
4500	87,196.266	505.556	-77,891.351	**-11000**	-9,957.944	0.000	0.000
4000	87,072.279	55.556	-79,775.851	**-11500**	0.000	0.000	0.000
3500	86,764.457	-394.444	-81,537.714	**-12000**	0.000	0.000	0.000
3000	86,281.379	-844.444	-83,176.725	**-12500**	0.000	0.000	0.000
2500	85,629.952	-1,294.444	-84,692.101	**-13000**	0.000	0.000	0.000
2000	84,815.673	-1,744.444	-86,082.470	**-13500**	0.000	0.000	0.000
1500	83,842.822	-2,194.444	-87,345.836	**-14000**	0.000	0.000	0.000
1000	82,714.598	-2,644.444	-88,479.534	**-14500**	0.000	0.000	0.000
500	81,433.222	-3,094.444	-89,480.160	**-15000**	0.000	0.000	0.000
0	80,000.000	-3,544.444	-90,343.478				

Tabla VIII.14 Resultados Elipse de Esfuerzos Triaxiales.

La parte derecha de la gráfica esta dada por los puntos $\left(\sigma_z(+), P_i\right)$ y la parte izquierda por los puntos $\left(\sigma_z(-), P_0\right)$ en este caso los puntos de P_i entre $15000\left[lb/pg^2\right]$ y $11500\left[lb/pg^2\right]$ positivos y negativos se desprecian pues el esfuerzo axial resulta ser un número imaginario razón por la cual no tiene interpretación física, esto es importante considerarlo durante la construcción del

programa capaz de obtener la gráfica, lo mismo ocurre para los valores de P_0 calculados, la elipse obtenida es:

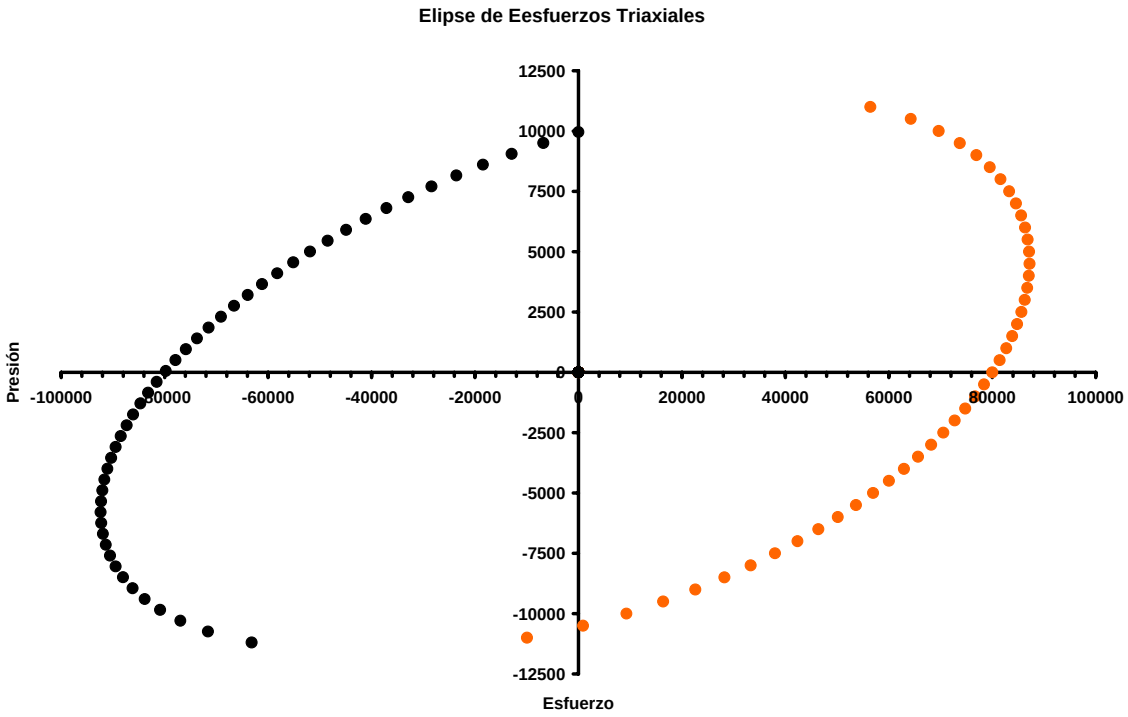


Figura VIII.10 Ejemplo Elipse de Esfuerzos Triaxiales.

Para diseñar el programa de cómputo es recomendable hacer uso de VISUAL BASIC 6.0® o de MATLAB 7.0® pues brindan capacidades gráficas para presentar la elipse.

VIII.VIII Ecuación general para los métodos explícito, implícito y Crank – Nicholson y solución numérica de la ecuación de difusión.

Los métodos explícito, implícito y Crank – Nicholson pueden ser representados por medio de la siguiente ecuación general:

$$\omega\left[\frac{p_{i+1}^{n+1}-2p_i^{n+1}+p_{i-1}^{n+1}}{(\Delta x)^2}\right]+(1-\omega)\left[\frac{p_{i+1}^{n}-2p_i^{n}+p_{i-1}^{n}}{(\Delta x)^2}\right]=\left(\frac{\phi\mu c_t}{k}\right)\frac{p_i^{n+1}-p_i^{n}}{\Delta t} \quad \text{...VIII.53}$$

Esta ecuación es la aproximación discretizada de la ecuación AVI.9 donde ω es un factor de ponderación, si $\omega=0$, se tiene el método explícito, si $\omega=1/2$ se obtiene el método Crank – Nicholson, si $\omega=1$ se tiene el método implícito, tratados en el capítulo VII.

VIII.VIII.I Solución de la Ecuación General.

Desarrollando la ecuación VII.15 se tiene:

$$\left[\frac{\omega\Delta t}{(\Delta x)^2}\right]p_{i-1}^{n+1}-\left[\frac{2\omega\Delta t}{(\Delta x)^2}+\frac{\phi\mu c_t}{k}\right]p_i^{n+1}+\left[\frac{\omega\Delta t}{(\Delta x)^2}\right]p_{i+1}^{n+1}=\left[\frac{(1-\omega)\Delta t}{(\Delta x)^2}\right]p_{i-1}^{n}+\left[\frac{2(1-\omega)\Delta t}{(\Delta x)^2}-\frac{\phi\mu c_t}{k}\right]p_i^{n}-\left[\frac{\omega\Delta t}{(\Delta x)^2}\right]p_{i+1}^{n}$$

…VIII.54

Considerando que:

$$\gamma=\frac{\Delta t}{(\Delta x)^2}$$

$$\alpha=\frac{\phi\mu c_t}{k}$$

…VIII.55

Y sustituyendo en VIII.54:

$$[\omega\gamma]p_{i-1}^{n+1}-[2\omega+\alpha]p_i^{n+1}+[\omega\gamma]p_{i+1}^{n+1}=-[(1-\omega)\gamma]p_{i-1}^{n}+[2(1-\omega)\gamma-\alpha]p_i^{n}-[(1-\omega)\gamma]p_{i+1}^{n}$$

…VII.56

Analizando esta última expresión se puede observar que del lado izquierdo se tienen tres incógnitas de presión con sus respectivos coeficientes, mientras que todo el término del lado derecho es conocido, por lo que se puede simplificar como:

$$a_i p_{i-1}^{n+1}+b_i p_i^{n+1}+c_i p_{i+1}^{n+1}=d_i \ldots \text{VIII.57}$$

En donde:

$$\begin{aligned}a_i &= \omega\gamma\\ b_i &= -(2\omega\gamma+\alpha)\\ c_i &= \omega\gamma\\ d_i &= -[(1-\omega)\gamma]p_{i-1}^{n}+[2(1-\omega)\gamma-\alpha]p_i^{n}-[(1-\omega)\gamma]p_{i+1}^{n}\end{aligned}$$

…VIII.58

Entonces la ecuación VIII.57 se escribe para cada celda i de la malla en la que fue discretizado el dominio, de esta manera se obtiene el sistema de ecuaciones lineales correspondiente al método implícito y al método Crank – Nicholson ($\omega=1$ y $\omega=1/2$ respectivamente), así, el sistema queda representado por:

$$\begin{bmatrix} b_1 & c_1 & & & & & & & \\ a_2 & b_2 & c_2 & & & & & & \\ & a_3 & b_3 & c_3 & & & & & \\ & & a_4 & b_4 & c_4 & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & & & & a_n & b_n \end{bmatrix}^n \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ \vdots \\ \vdots \\ \vdots \\ p_{n-1} \\ p_n \end{bmatrix}^{n+1} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \vdots \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}^n \quad \ldots \text{VIII.59}$$

Se observa que es un sistema tridiagonal que se puede resolver por alguno de los métodos vistos en el capítulo I o por medio del algoritmo de Thomas tratado en el anexo A7.

VIII.VIII.II Ejemplo.

A partir de los siguientes datos y considerando un yacimiento homogeneo, flujo monofásico en una sola dirección simular el comportamiento de 0.1 días hasta 1,000,000 días utilizando nodos distribuidos, y el esquema implícito de solución.

Datos	
$k[mD]$	10.0
$\mu[cp]$	0.5
$c_t\left[lb/pg^2\right]^{-1}$	0.000015
ϕ	23.40%
$L[pie]$	400.0
$\#nodos$	10
$P_{inicial}\left(x,t=0\right)\left[lb/pg^2\right]$	1500.0
$P_{frontera}\left(x=0,t>0\right)\left[lb/pg^2\right]$	0.0
$P_{frontera}\left(x=L,t>0\right)\left[lb/pg^2\right]$	1500.0

Tabla VIII.15 Ejemplo Ecuación General.

Los datos se encuentran en unidades de campo, para obtener resultados correctos es necesario convertirlos a un sistema de unidades consistente, en este caso se convertirán a unidades del sistema internacional, en la siguiente tabla se resumen estos valores:

Datos SI	
$k[m^2]$	9.86E-15
$\mu[Pa \cdot S]$	500.0
$c_t[Pa]^{-1}$	2.17557E-09
ϕ	23.40%
$L[m]$	121.92
$\#nodos$	10
$P_{inicial}(x, t=0)[Pa]$	10342135.5
$P_{frontera}(x=0, t>0)[Pa]$	0.0
$P_{frontera}(x=L, t>0)[Pa]$	10342135.5

Tabla VIII.16 Ejemplo Ecuación General Sistema Internacional.

Para calcular los nodos distribuidos se sigue el procedimiento descrito en la sección AV.II.I.I del anexo AV. El número total de nodos que se requieren en el ejercicio es igual a 10, por lo tanto el espaciamiento entre nodos Δx se calcula con la ecuación AV.I:

$$\Delta x = \frac{L}{IMAX - 1} = \frac{121.92[m]}{10 - 1} = 13.5466[m]$$

Entonces la posición de los nodos se calcula mediante la ecuación AV.2:

$$x_1 = (1-1)*13.5466[m] = 0.0[m] = 0.0[pie]$$

$$x_2 = (2-1)*13.5466[m] = 13.5466[m] = 44.4442[pie]$$

$$\vdots$$

$$x_{10} = (10-1)*13.5466[m] = 121.9194[m] = 400.0[pie]$$

En la siguiente tabla se resume la posición de cada uno de los nodos:

Nodo	*Posición*
1	0.00
2	44.44
3	88.89
4	133.33
5	177.78
6	222.22
7	266.67
8	311.11
9	355.56
10	400.00

Tabla VIII.17 Posición de Nodos.

Entonces para 0.1 días y $\omega = 1.0$ para tener el esquema implícito, y utilizando las ecuaciones VIII.55 y VIII.58 para calcular los vectores $a(i), b(i), c(i), d(i)$ se tiene:

$$\gamma = \frac{\Delta t}{(\Delta x)^2} = \frac{0.1[día] * 86,400[seg/día]}{(13.5466[m])^2} = 47.08\left[\frac{seg}{m^2}\right]$$

$$\alpha = \frac{\phi \mu c_t}{k} = \frac{(0.234)(500[Pa \cdot s])(2.17557 * 10^{-9}[Pa^{-1}])}{9.86 * 10^{-15}[m^2]} = 225,815,587.22\left[\frac{s}{m^2}\right]$$

Vector $a(i)$:

$$a_1 = 0.0$$
$$a_2 = \omega\gamma = 1.0 * 47.08 = 47.08$$
$$\vdots$$
$$a_{10} = \omega\gamma = 1.0 * 47.08 = 47.08$$

Vector $b(i)$:

$$b_1 = -(2\omega\gamma + \alpha) = -[(2 * 1.0 * 47.08) + 225,815,587.22] = -225,815,681.4$$
$$b_2 = -(2\omega\gamma + \alpha) = -[(2 * 1.0 * 47.08) + 225,815,587.22] = -225,815,681.4$$
$$\vdots$$
$$b_{10} = -(2\omega\gamma + \alpha) = -[(2 * 1.0 * 47.08) + 225,815,587.22] = -225,815,681.4$$

Vector $c(i)$:

$$c_1 = \omega\gamma = 1.0 * 47.08 = 47.08$$
$$c_2 = \omega\gamma = 1.0 * 47.08 = 47.08$$
$$\vdots$$
$$c_{10} = 0.0$$

Vector $d(i)$:

$$d_i = -[(1-\omega)\gamma]\, p_{i-1}^n + [2(1-\omega)\gamma - \alpha]\, p_i^n - [(1-\omega)\gamma]\, p_{i+1}^n$$

Como $\omega = 1.0$ la ecuación que sirve para calcular el vector $d(i)$ se reduce a $d_i = [2(1-\omega)\gamma - \alpha]\, p_i^n$, entonces calculando los elementos $d(i)$:

$$d_1 = \left[\left(2\left(1-1.0\right)47.08\right) - 225,815,587.22\right] * 10,342,135.5 = 0.0$$
$$d_2 = \left[\left(2\left(1-1.0\right)47.08\right) - 225,815,587.22\right] * 10,342,135.5 = -2.335415401 * 10^{15}$$
$$d_3 = \left[\left(2\left(1-1.0\right)47.08\right) - 225,815,587.22\right] * 10,342,135.5 = -2.335415401 * 10^{15}$$
$$\vdots$$
$$d_{10} = \left[\left(2\left(1-1.0\right)47.08\right) - 225,815,587.22\right] * 10,342,135.5 = -2.335415401 * 10^{15}$$

En este caso el elemento $d\left(1\right)$ es igual a cero por la condición de frontera:

$$P_{frontera}\left(x = 0, t > 0\right) = 0.0\left[Pa\right]$$

Como ya se explicó anteriormente en el desarrollo del algoritmo, el vector $a\left(i\right)$ representa, la diagonal inmediata inferior a la diagonal principal mientras que el vector $c\left(i\right)$ a la diagonal inmediata superior a la diagonal principal, en este caso, ambas contienen nueve elementos mientras que la principal tiene 9, los elementos $a\left(1\right)$ y $c\left(10\right)$ son igualados a cero para lograr que tengan diez elementos cada uno y así poder utilizar el algoritmo de Thomas correctamente.

Estos resultados se obtuvieron de forma manual haciendo uso de una calculadora CASIO® modelo $fx-991MS$ por lo que no son precisos, en las siguientes tablas se resumen los resultados obtenidos con una computadora por medio de un simulador programado en FORTRAN®, razón por la cual presentan una mayor precisión:

$Pos._{Nodos}$	$Cond._{Inicial\ Para\ t_{acum}}$	*Coeficientes*				$P\left(x,t\right)$
		Δt	0.1	t_{acum}	0.1	
		$a\left(i\right)$	$b\left(i\right)$	$c\left(i\right)$	$d\left(i\right)$	
0	1500.00	0.00	-25815633.43	47.08	0.00	0.000
44.44	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1499.997
88.89	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1500.000
133.33	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1500.000
177.78	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1500.000
222.22	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1500.000
266.67	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1500.000
311.11	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1500.000
355.56	1500.00	47.08	-25815633.43	47.08	-266987792161648.00	1500.000
400	1500.00	47.08	-25815633.43	0.00	-266987792161648.00	1500.000

Tabla VIII.18 Coeficientes y Presión $t_{acum} = 0.1\left[día\right]$.

$Pos._{Nodos}$	$Cond._{Inicial}$ $Para\ t_{acum}$	Coeficientes				$P(x,t)$
		Δt	10.0	t_{acum}	10.1	
		$a(i)$	$b(i)$	$c(i)$	$d(i)$	
0	0.000	0.00	-25825049.69	4755.22	0.00	0.000
44.44	1499.997	4755.22	-25825049.69	4755.22	-266987305242707.00	1499.721
88.89	1500.000	4755.22	-25825049.69	4755.22	-266987792160760.00	1500.000
133.33	1500.000	4755.22	-25825049.69	4755.22	-266987792161648.00	1500.000
177.78	1500.000	4755.22	-25825049.69	4755.22	-266987792161648.00	1500.000
222.22	1500.000	4755.22	-25825049.69	4755.22	-266987792161648.00	1500.000
266.67	1500.000	4755.22	-25825049.69	4755.22	-266987792161648.00	1500.000
311.11	1500.000	4755.22	-25825049.69	4755.22	-266987792161648.00	1500.000
355.56	1500.000	4755.22	-25825049.69	4755.22	-266987792161648.00	1500.000
400	1500.000	4755.22	-25825049.69	0.00	-266987792161648.00	1500.000

Tabla VIII.19 Coeficientes y Presión $t_{acum} = 10.1[día]$.

$Pos._{Nodos}$	$Cond._{Inicial}$ $Para\ t_{acum}$	Coeficientes				$P(x,t)$
		Δt	1,000.0	t_{acum}	1,010.1	
		$a(i)$	$b(i)$	$c(i)$	$d(i)$	
0	0.000	0.00	-26766676.55	475568.65	0.00	0.000
44.44	1499.721	475568.65	-26766676.55	475568.65	-266938153499502.00	1473.537
88.89	1500.000	475568.65	-26766676.55	475568.65	-266987783020698.00	1499.530
133.33	1500.000	475568.65	-26766676.55	475568.65	-266987792159965.00	1499.992
177.78	1500.000	475568.65	-26766676.55	475568.65	-266987792161647.00	1500.000
222.22	1500.000	475568.65	-26766676.55	475568.65	-266987792161648.00	1500.000
266.67	1500.000	475568.65	-26766676.55	475568.65	-266987792161648.00	1500.000
311.11	1500.000	475568.65	-26766676.55	475568.65	-266987792161648.00	1500.000
355.56	1500.000	475568.65	-26766676.55	475568.65	-266987792161648.00	1500.000
400	1500.000	475568.65	-26766676.55	0.00	-266987792161648.00	1500.000

Tabla VIII.20 Coeficientes y Presión $t_{acum} = 1,010.1[día]$.

$Pos._{Nodos}$	$Cond._{Inicial}$ $Para\ t_{acum}$	*Coeficientes*				$P(x,t)$
		Δt	$100,000.0$	t_{acum}	$101,010.1$	
		$a(i)$	$b(i)$	$c(i)$	$d(i)$	
0	0.000	0.00	-120929362.55	47556911.64	0.00	0.000
44.44	1473.537	47556911.64	-120929362.55	47556911.64	-262277600838906.00	944.134
88.89	1499.530	47556911.64	-120929362.55	47556911.64	-266904070041054.00	1229.599
133.33	1499.992	47556911.64	-120929362.55	47556911.64	-266986304183282.00	1368.535
177.78	1500.000	47556911.64	-120929362.55	47556911.64	-266987765716101.00	1436.112
222.22	1500.000	47556911.64	-120929362.55	47556911.64	-266987791691636.00	1469.009
266.67	1500.000	47556911.64	-120929362.55	47556911.64	-266987792153294.00	1485.084
311.11	1500.000	47556911.64	-120929362.55	47556911.64	-266987792161499.00	1493.061
355.56	1500.000	47556911.64	-120929362.55	47556911.64	-266987792161645.00	1497.271
400	1500.000	47556911.64	-120929362.55	0.00	-266987792161648.00	1500.000

Tablas VIII.21 Coeficientes y Presión $t_{acum} = 101,010.1[día]$.

$Pos._{Nodos}$	$Cond._{Inicial}$ $Para\ t_{acum}$	*Coeficientes*				$P(x,t)$
		Δt	$1,000,000.0$	t_{acum}	$1,101,010.1$	
		$a(i)$	$b(i)$	$c(i)$	$d(i)$	
0	0.000	0.00	-9537197962.33	4755691211.54	0.00	0.000
44.44	944.134	4755691211.54	-9537197962.34	4755691211.54	-168048248752507.00	323.066
88.89	1229.599	4755691211.54	-9537197962.34	4755691211.54	-218858565935459.00	481.665
133.33	1368.535	4755691211.54	-9537197962.34	4755691211.54	-243588118939568.00	636.204
177.78	1436.112	4755691211.54	-9537197962.34	4755691211.54	-255616326356523.00	786.768
222.22	1469.009	4755691211.54	-9537197962.34	4755691211.54	-261471722522308.00	933.806
266.67	1485.084	4755691211.54	-9537197962.34	4755691211.54	-264332805237175.00	1077.940
311.11	1493.061	4755691211.54	-9537197962.34	4755691211.54	-265752669071188.00	1219.863
355.56	1497.271	4755691211.54	-9537197962.34	4755691211.54	-266502065300583.00	1360.303
400	1500.000	4755691211.54	-9537197962.34	0.00	-266987792161648.00	1500.000

Tablas VIII.22 Coeficientes y Presión $t_{acum} = 1,101,010.1[día]$.

Para poder realizar un mejor análisis se debe acumular el tiempo, por lo que Δt es el valor que se tiene en la celda correspondiente a t_{acum}. Realizando una gráfica de presión contra longitud se puede apreciar el comportamiento de la presión respecto al tiempo:

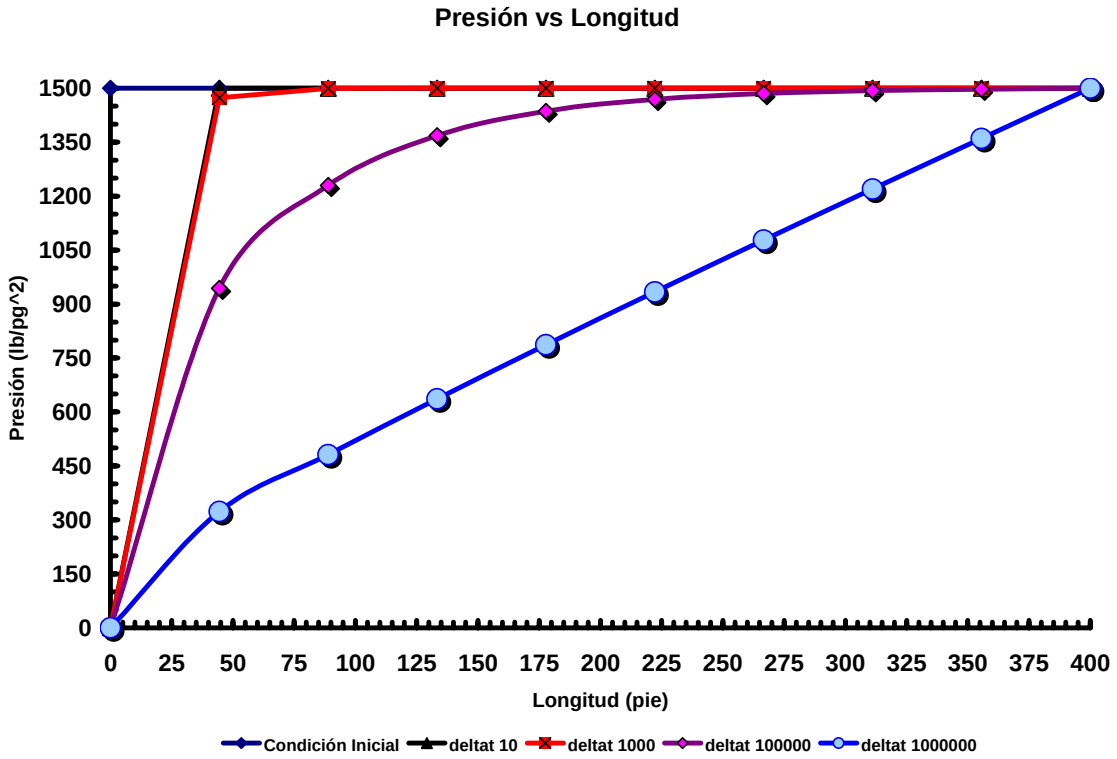


Figura VIII.11 Gráfica Presión vs Longitud.

La condición inicial dice que lo largo del yacimiento se tenia una presión constante de $1500.0\left[lb/pg^2\right]$, las condiciones de frontera dicen que en el momento en que $t \neq 0.0$, la presión en $x = 0.0$ cae a $0.0\left[lb/pg^2\right]$, esto se puede explicar considerando la existencia de un pozo productor, y la presión en $x = L = 400\left[pie\right]$ cae a $1500.0\left[lb/pg^2\right]$, esto se explica considerando la existencia de un pozo inyector que mantenga la presión en ese punto, la línea marcada para un $\Delta t = 1,000,000\left[día\right]$ representa el perfil de presiones que se tendrían en el yacimiento para tiempos muy grandes de explotación.

Para diseñar el programa de cómputo es recomendable hacer uso de VISUAL BASIC 6.0® o MATLAB 7.0® pues brinda capacidades gráficas para presentar la gráfica.

VIII.IX Solución analítica de la ecuación de difusión para flujo radial y yacimiento infinito.

Si se considera flujo radial, la ecuación de difusión se escribe de la siguiente forma:

$$\frac{\partial^2 p}{\partial r^2}+\frac{1}{r}\frac{\partial p}{\partial r}=\frac{\phi\mu c}{k_r}\frac{\partial p}{\partial t} \quad \ldots \text{AVII.19}$$

VIII.IX.I Desarrollo.

Se establecen los términos de las derivadas en función de la transformada de Boltzman. Aplicando regla de la cadena para $\frac{\partial p}{\partial r}$ se tiene:

$$\frac{\partial p}{\partial r}=\frac{\partial p}{\partial y}\frac{\partial y}{\partial r}=\frac{2y}{r}\frac{\partial p}{\partial y} \quad \ldots \text{VIII.60}$$

Aplicando la regla de la cadena para $\frac{\partial^2 p}{\partial r^2}$:

$$\frac{\partial^2 p}{\partial r^2}=\frac{\partial}{\partial r}\left(\frac{\partial p}{\partial r}\right)=\frac{\partial}{\partial y}\left(\frac{\partial p}{\partial r}\right)\frac{\partial y}{\partial r}=\frac{\partial}{\partial y}\left(\frac{\partial p}{\partial y}\frac{\partial y}{\partial r}\right)\frac{\partial y}{\partial r} \quad \ldots \text{VIII.61}$$

Derivando:

$$\frac{\partial^2 p}{\partial r^2}=\frac{\partial p}{\partial y}\frac{\partial}{\partial y}\left(\frac{\partial y}{\partial r}\right)\frac{\partial y}{\partial r}+\frac{\partial y}{\partial r}\frac{\partial}{\partial y}\left(\frac{\partial p}{\partial y}\right)\frac{\partial y}{\partial r}=\frac{\partial p}{\partial y}\frac{\partial}{\partial y}\left(\frac{\partial y}{\partial r}\right)\frac{\partial y}{\partial r}+\frac{\partial^2 p}{\partial y^2}\left(\frac{\partial y}{\partial r}\right)^2$$

$$\frac{\partial^2 p}{\partial r^2}=\frac{\partial y}{\partial r}\left[\frac{\partial p}{\partial y}\frac{\partial}{\partial y}\left(\frac{\partial y}{\partial r}\right)+\frac{\partial^2 p}{\partial y^2}\frac{\partial y}{\partial r}\right] \quad \ldots \text{VIII.62}$$

Como $\frac{\partial y}{\partial r}=\frac{2y}{r}$, entonces:

$$\frac{\partial^2 p}{\partial r^2}=\frac{2y}{r}\left[\frac{\partial}{\partial y}\left(\frac{2y}{r}\right)\frac{\partial p}{\partial y}+\frac{2y}{r}\frac{\partial^2 p}{\partial y^2}\right]=\frac{2y}{r}\left[\left(\frac{r\frac{\partial(2y)}{\partial y}-2y\frac{\partial r}{\partial y}}{r^2}\right)\frac{\partial p}{\partial y}+\frac{2y}{r}\frac{\partial^2 p}{\partial y^2}\right]$$

$$\frac{\partial^2 p}{\partial r^2}=\frac{2y}{r}\left[\left(\frac{2r-2y\frac{\partial r}{\partial y}}{r^2}\right)\frac{\partial p}{\partial y}+\frac{2y}{r}\frac{\partial^2 p}{\partial y^2}\right] \quad \ldots \text{VIII.63}$$

Nuevamente, como $\frac{\partial y}{\partial r}=\frac{2y}{r}$, se tiene que $\frac{\partial r}{\partial y}=\frac{r}{2y}$.

De donde:

$$\frac{\partial^2 p}{\partial r^2}=\frac{2y}{r}\left[\left(\frac{2r-2y\frac{\partial r}{\partial y}}{r^2}\right)\frac{\partial p}{\partial y}+\frac{2y}{r}\frac{\partial^2 p}{\partial y^2}\right]=\frac{2y}{r}\left[\left(\frac{2}{r}-\frac{2y}{r^2}\frac{r}{2y}\right)\frac{\partial p}{\partial y}+\frac{2y}{r}\frac{\partial^2 p}{\partial y^2}\right]$$

$$\frac{\partial^2 p}{\partial r^2}=\frac{2y}{r}\left[\left(\frac{2}{r}-\frac{1}{r}\right)\frac{\partial p}{\partial y}+\frac{2y}{r}\frac{\partial^2 p}{\partial y^2}\right]$$

$$\frac{\partial^2 p}{\partial r^2}=\frac{2y}{r}\left[\frac{1}{r}\frac{\partial p}{\partial y}+\frac{2y}{r}\frac{\partial^2 p}{\partial y^2}\right] \ldots \text{VIII.64}$$

Esto es:

$$\frac{\partial^2 p}{\partial r^2}=\frac{2y}{r^2}\frac{\partial p}{\partial y}+\frac{4y^2}{r^2}\frac{\partial^2 p}{\partial y^2} \ldots \text{VIII.65}$$

Aplicando la regla de la cadena para $\frac{\partial p}{\partial t}$:

$$\frac{\partial p}{\partial t}=\frac{\partial p}{\partial y}\frac{\partial y}{\partial t}=\frac{\partial p}{\partial y}\frac{\partial}{\partial t}\left(\frac{\phi\mu c_t r^2}{4kt}\right) \ldots \text{VIII.66}$$

Derivando:

$$\frac{\partial p}{\partial t}=\frac{\partial p}{\partial y}\left[\frac{4kt\frac{\partial}{\partial t}\left(\phi\mu c_t r^2\right)-\phi\mu c_t r^2\frac{\partial}{\partial t}\left(4kt\right)}{\left(4kt\right)^2}\right] \ldots \text{VIII.67}$$

Pero $\phi\mu c_t r^2$ es constante en el tiempo, por consiguiente su derivada es cero:

$$\frac{\partial p}{\partial t}=\frac{\partial p}{\partial y}\left(-\frac{\phi\mu c_t r^2 4k}{16k^2t^2}\right)=\frac{\partial p}{\partial y}\left(-\frac{\phi\mu c_t r^2}{4kt^2}\right)$$

$$\frac{\partial p}{\partial t}=\frac{\partial p}{\partial y}\left(-\frac{\phi\mu c_t r^2}{4kt}\frac{1}{t}\right) \ldots \text{VIII.68}$$

Como $y = \frac{\phi \mu c_t r^2}{4kt}$, se tiene:

$$\frac{\partial p}{\partial t} = \frac{\partial p}{\partial y}\left(-\frac{y}{t}\right) \ldots \text{VIII.69}$$

Sustituyendo estas representaciones paras las derivadas en la ecuación de difusión:

$$\frac{1}{r}\frac{2y}{r}\frac{\partial p}{\partial y} + \frac{2y}{r^2}\frac{\partial p}{\partial y} + \frac{4y^2}{r^2}\frac{\partial^2 p}{\partial y^2} = \frac{\phi \mu c_t}{k}\left(-\frac{y}{t}\right)\frac{\partial p}{\partial y} \ldots \text{VIII.70}$$

Como $y = \frac{\phi \mu c_t r^2}{4kt}$, se puede simplificar de la siguiente manera:

$$\frac{4y}{r^2}\frac{\partial p}{\partial y} + \frac{4y^2}{r^2}\frac{\partial^2 p}{\partial y^2} = -\frac{4yt}{r^2}\frac{y}{t}\frac{\partial p}{\partial y}$$

$$\frac{4y}{r^2}\left(\frac{\partial p}{\partial y} + y\frac{\partial^2 p}{\partial y^2}\right) = -\frac{4y^2}{r^2}\frac{\partial p}{\partial y} \ldots \text{VIII.71}$$

Al agrupar se tiene:

$$\frac{\partial p}{\partial y} + y\frac{\partial^2 p}{\partial y^2} + y\frac{\partial p}{\partial y} = 0 \ldots \text{VIII.72}$$

La cual se escribe como:

$$(1+y)\frac{\partial p}{\partial y} + y\frac{\partial^2 p}{\partial y^2} = 0 \ldots \text{VIII.73}$$

La ecuación VIII.73 es la ecuación de difusión en términos de la transformada de Boltzman.

VIII.IX.II Algoritmo.

Después del periodo de almacenamiento, la variación de presión que se podría medir en la cara del pozo refleja la caída de la presión a través del yacimiento. A tiempos cortos la respuesta no esta influenciada por las fronteras, lo que lleva a pensar en un yacimiento de extensión infinita, lógicamente a tiempos largos este efecto desaparece.

Este tiempo intermedio, entre tiempos cortos donde la respuesta esta influenciada por el almacenamiento y tiempos grandes donde dominan los efectos de frontera se conoce como periodo de yacimiento infinito.

Aunque existen varios tipos de flujos identificados durante este periodo de yacimiento infinito, uno de los más fácilmente identificados es el flujo radial. El flujo radial para yacimientos comportándose infinitamente es la base de la interpretación de las técnicas de pruebas de presión. Ahora aplicando la condición de frontera interior, se expresan sobre la ecuación VIII.73:

$$\lim_{r\to 0}\left(r\frac{\partial p}{\partial r}\right)=\lim_{y\to 0}\left(r\frac{2y}{r}\frac{\partial p}{\partial y}\right)=\lim_{y\to 0}2\left(y\frac{\partial p}{\partial y}\right)=\frac{q\mu}{2\pi kh}\ldots\text{VIII.75}$$

Para la condición de frontera exterior:

$$\lim_{r\to\infty}P=\lim_{y\to\infty}P=p_i\ldots\text{VIII.76}$$

Si $dp/dy=p'$ entonces la ecuación VIII.73 se rescribe como:

$$(1+y)\,p'+y\frac{\partial p'}{\partial y}=0\ldots\text{VIII.77}$$

Integrando por partes y agrupando:

$$\frac{dp'}{p'}=-\frac{1+y}{y}dy$$

$$\int\frac{dp'}{p'}=-\int\frac{1+y}{y}dy$$

$$\int\frac{dp'}{p'}=-\int\frac{dy}{y}-\int dy$$

Resolviendo:

$$\ln p'=-\ln y-y+C\ldots\text{VIII.78}$$

Aplicando antilogaritmos:

$$p'=e^{-\ln y-y+C}\Rightarrow p'=e^{-\ln y}e^{-y}e^{C}\Rightarrow p'=\frac{1}{e^{\ln y}}e^{-y}C_1\ldots\text{VIII.79}$$

$$p'=\frac{1}{y}e^{-y}C\Rightarrow yp'=e^{-y}C_1\Rightarrow y\frac{\partial p}{\partial y}=C_1e^{-y}\ldots\text{VIII.80}$$

Aplicando la condición de frontera interior:

$$\lim_{y\to 0}\left(y\frac{\partial p}{\partial y}\right)=\lim_{y\to 0}C_1e^{-y}$$

$$C_1=\frac{q\mu}{4\pi kh}\ldots\text{VIII.81}$$

Sustituyendo en VIII.80:

$$y\frac{\partial p}{\partial y}=\frac{q\mu}{4\pi kh}e^{-y} \ldots \text{VIII.82}$$

Separando e integrando, asignado arbitrariamente ∞ al límite inferior se tiene:

$$\int dp=\frac{q\mu}{4\pi kh}\int_{\infty}^{y}\frac{e^{-y}}{y}dy \Rightarrow p=-\frac{q\mu}{4\pi kh}\int_{y}^{\infty}\frac{e^{-y}}{y}dy+C_2 \ldots \text{VIII.83}$$

Aplicando límites y la condición de frontera exterior:

$$\lim_{y\to 0} p=-\frac{q\mu}{4\pi kh}\int_{y}^{\infty}\frac{e^{-y}}{y}dy+C_2 \ldots \text{VIII.84}$$

Si $C_2=P_i$, entonces:

$$P_{(r,t)}=-\frac{q\mu}{4\pi kh}\int_{y}^{\infty}\frac{e^{-y}}{y}dy+Pi \ldots \text{VIII.85}$$

La integral exponencial está definida como $-Ei(-y)=\int_{y}^{\infty}\frac{e^{-y}}{y}dy$, por lo que:

$$P_{(r,t)}=Pi+\frac{q\mu}{4\pi kh}Ei(-y) \ldots \text{VIII.86}$$

En donde $-y=\left(\frac{\phi\mu c_t r^2}{4kt}\right)$, y considerando:

$$\eta=\frac{k}{\phi\mu c} \ldots \text{VIII.87}$$

Se tiene:

$$p_{(r,t)}=p_i-\frac{q\mu B}{4\pi kh}E_i\left[-\frac{r^2}{4\eta t}\right] \ldots \text{VIII.88}$$

Que es la llamada solución fuente lineal.

Esta es la solución analítica para un yacimiento infinito, en donde $p_{(r,t)}$ es la presión en un punto r para un tiempo t, E_i es la integral exponencial cuyo algoritmo se detalla en el anexo VIII de este trabajo.

VIII.IX.III Ejemplo.

En base a los siguientes datos determinar la presión para radios de 0.16, 0.33, 4.92, 9.84, 49.2, 98.4, 492, 984 pies, para 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0,

10000.0 y 100000.0 días, utilizando la solución analítica dada por la ecuación VIII.88.

Datos	
$P_{inicial}\left[lb/pg^2\right]$	5000.0
$k[mD]$	80.0
$h[pie]$	20.0
$\mu[cp]$	0.9
$c_t\left[lb/pg^2\right]^{-1}$	0.000015
B_o	1.4
ϕ	25.5
q_o	100.0

Tabla VIII.23 Ejemplo Solución Analítica.

Los datos se encuentran en unidades de campo, para obtener resultados correctos es necesario convertirlos a un sistema de unidades consistente, en este caso se convertirán a unidades del sistema internacional, y sustituir directamente los valores en la ecuación VIII.88, por ejemplo:

Para $r = 9.84[pie] = 3.0[m]$ y $t = 0.01[día]$ se tiene:

$$p = 34,555,000 - 36,700E_i\left[\frac{3^2}{57,800(0.01)}\right] = 34,516,516[Pa]$$

En la siguiente tabla se muestran los resultados obtenidos:

Radios	*Tiempos*$[día]$							
	0.01	0.1	1.0	10.0	100.0	1000.0	10000.0	100000.0
0.16	4937.40	4925.16	4912.91	4900.67	4888.43	4876.19	4863.94	4851.70
0.33	4944.77	4932.53	4920.28	4908.04	4895.80	4883.56	4871.31	4859.07
4.92	4973.54	4961.32	4949.08	4936.84	4924.60	4912.35	4900.11	4887.87
9.84	4980.85	4968.69	4956.45	4944.21	4931.97	4919.72	4907.48	4895.24
49.2	4996.17	4985.60	4973.54	4961.32	4949.08	4936.84	4924.60	4912.35
98.4	4999.51	4992.38	4980.85	4968.69	4956.45	4944.21	4931.97	4919.72
492	5000.00	4999.98	4996.17	4985.60	4973.54	4961.32	4949.08	4936.84
984	5000.00	5000.00	4999.51	4992.38	4980.85	4968.69	4956.45	4944.21

Tabla VIII.24 Resultados Ejemplo Solución Analítica.

Graficando en semilog los resultados obtenidos se tiene la siguiente gráfica:

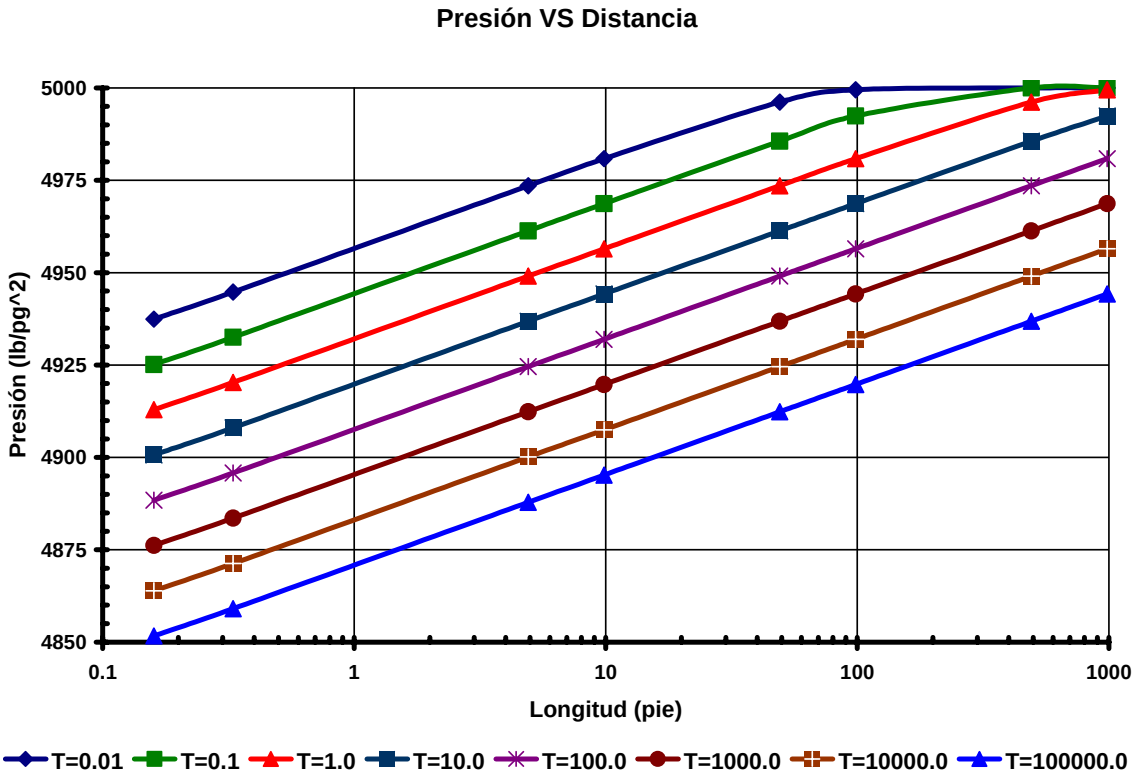


Figura VIII.12 Gráfica Semilog Presión vs Longitud.

Para diseñar el programa de cómputo es recomendable hacer uso de MATLAB 7.0® pues brinda capacidades gráficas para presentar la gráfica en semilog.

Las aplicaciones presentadas en este capítulo son sencillos de programar, sin embargo, si no se tienen las bases de programación pueden llegar a complicarse.

Todas son aplicaciones que los alumnos pueden llegar a utilizar a lo largo de sus estudios, pues la gran mayoría se tomaron de los apuntes de las materias impartidas por los profesores de la carrera.

CONCLUSIONES Y RECOMENDACIONES.

Conclusiones.

La industria petrolera está en constante evolución gracias a los avances científicos y tecnológicos, estos avances no solo se limitan a las herramientas utilizadas en el trabajo de campo, también abarcan los campos de análisis, diseño e interpretación. En ambos casos la intervención de las computadoras se incrementa día a día, a través de programas capaces de resolver diferentes problemas que plantea la industria (y la vida misma), esto obliga al ingeniero petrolero (de hecho ingenieros de cualquier especialidad) a conocer la forma en que una computadora trabaja y es capaz de resolver los problemas que se le plantean.

Para esto se vuelve de gran importancia que el ingeniero petrolero conozca las bases para programar, pues esto le permitirá entender mejor la manera en que la computadora llega a una determinada solución, analizar está y definir si es correcta o no.

Además se ha podido observar el desarrollo de habilidades como el razonamiento, análisis y creatividad en las personas que han aprendido las bases de programación.

Si a lo anterior se añade la gran cantidad de empresas de servicios dentro de la industria petrolera que obtienen una gran parte de sus ganancias por la venta y/o renta de software, es obvio pensar que existe en esta área un amplio campo de trabajo para un ingeniero petrolero, pues si bien no cuenta con los conocimientos avanzados de programación necesarios para elaborar dicho software comercial, si tiene una sólida base de conocimientos petroleros que en conjunto con las bases de programación y trabajando dentro de un equipo multidisciplinario (ingenieros de sistemas, computación, físicos, matemáticos, etc.), le permitirán desarrollar nuevos programas.

Todo esto muestra la importancia de una asignatura que inicia a los futuros ingenieros petroleros en el campo de la programación de computadoras.

Recomendaciones.

Si se considera la situación actual de la industria petrolera dentro del país, en donde la falta de recursos económicos tiene en riesgo la existencia de la misma, el gasto de una gran parte de esos recursos en software comercial resulta ser demasiado costoso, una posible solución es impulsar en los alumnos de la Facultad de Ingeniería, programas de investigación y desarrollo de software económico y de calidad, que permitan ahorrar grandes cantidades de dinero invertido en licencias. Como muestra de lo que se puede lograr basta con revisar el trabajo de tesis ***"Optimación de Redes de Gas"*** elaborado por Iván Santamaría Vite en el año 2001, o el trabajo ***"Automatización de Correlaciones y Modelos Mecanisticos de Flujo Multifásico Vertical Ascendente"*** elaborado por Teodoro Iván Guerrero Sarabia, ambos trabajos incluyen programas de computo complejos que con un poco de trabajo extra podrían ser de gran utilidad en la industria.

En el aspecto educativo se recomienda realizar un revisión general del plan de estudios actual de la carrera de Ingeniería Petrolera, pues en sus modificaciones se eliminó una asignatura antecedente a Programación Avanzada llamada Recursos Computacionales, en esta asignatura se enseñaba a los alumnos los aspectos más sencillos de la programación, esto les permitía que al tomar la siguiente asignatura (Programación Avanzada) tuvieran la capacidad suficiente para desarrollar los algoritmos presentados en este trabajo, actualmente los alumnos entran a la asignatura con conocimientos nulos de programación, esto retrasa el avance en el programa establecido por el temario, pues se invierte tiempo valioso en los aspectos más básicos.

Lo anterior provoca la necesidad, urgente, de volver a incluir por lo menos una materia enfocada a esto, reducir el temario de la asignatura Programación Avanzada o en su defecto recuperar la hora y media reducida en el cambio de plan de estudios, además de ofrecer a los alumnos alternativas de aprendizaje como cursos intersemestrales.

Al alumno se le recomienda practicar constantemente con los ejemplos sencillos planteados en este trabajo antes de comenzar a programar los algoritmos que comprenden el temario de la asignatura, realizar pruebas de escritorio en base a dichos algoritmos y sobre todo tener paciencia, pues con estudio y práctica, es sencillo comenzar a programar.

AI. PROGRAMACIÓN EN FORTRAN®.

AI.I Elementos básicos de programación en FORTRAN®.

AI.I.I Inicio de un programa en VISUAL FORTRAN®.

Para comenzar a escribir el código de un programa en VISUAL FORTRAN® se recomienda crear un espacio de trabajo, aunque no es la única manera de trabajar esta brinda algunas ventajas, pues permite insertar en este espacio varias hojas de código que se pueden utilizar como módulos, estos se compilan automáticamente, a diferencia de otros compiladores que tienen algunos problemas al momento de compilar módulos.

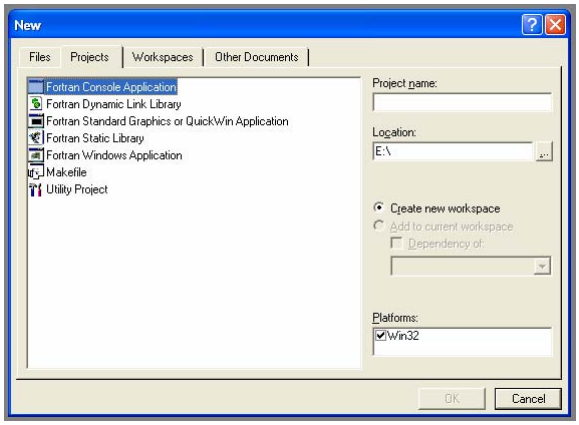


Figura AI.1 Programa Nuevo.

Para crear el espacio de trabajo solo debe dar clic en “Nuevo”, aparecerá un asistente en el que solo se debe indicar el nombre del proyecto y la ubicación en disco que se desea para el espacio de trabajo y seleccionar la opción de “Crear un Nuevo Espacio de Trabajo”. En el nuevo asistente solo es necesario seleccionar la opción de “Proyecto Simple” y automáticamente se generara el espacio de trabajo que permite escribir el código en FORTRAN® 90.

AI.I.II Variables.

Para FORTRAN® (y para cualquier lenguaje) existen diferentes tipos de variables con las que se construye el programa, en este caso se tienen las siguientes:

- **INTEGER:** Variable numérica de tipo entero.
- **REAL:** Variable numérica de tipo real.
- **CHARACTER:** Variable de tipo carácter.
- **LOGICAL:** Variables lógicas.

Las variables pueden asumir como valores una combinación de dígitos, letras, o ambos dependiendo del tipo de variable, para nombrarlas se deben utilizar nombres que contengan hasta 31 caractéres de longitud, aunque esto no es recomendable, pues resulta impractico, además el primer carácter del nombre de la variable no puede ser del tipo numérico.

Por ejemplo para dar nombre a una variable referente a densidad relativa del aceite, *denrelo* seria correcto, pues representa adecuadamente y de forma sencilla el concepto con el que se asocia la variable a lo largo del programa y con las operaciones que la involucran, en cambio *densidad_relativa_aceite* no es práctico, pues aunque presenta claramente el concepto con el que esta asociada es muy largo.

Para declarar una variable se deben escribir las siguientes líneas para cada tipo de variables:

```
INTEGER::I,J,K
REAL(8)::A,B,C
REAL::X,Y
CHARACTER(50)::RUTA
LOGICAL::COMPROBAR
```

Las variables I, J, K son de tipo entero, las variables de A, B, C son de tipo real de doble precisión, esto lo define el número 8 escrito entre paréntesis, en cambio las variables X, Y son de tipo real pero de precisión sencilla, esto se refiere al modo en que la computadora representa los números visto en el Capitulo I, La variable RUTA es de tipo carácter y puede tener una longitud de cincuenta caracteres, esto lo define el número 50 escrito entre paréntesis, la variable COMPROBAR es de tipo lógica.

Mas adelante se abordara el tema de las variables que toman la forma de arreglos vectoriales o matriciales.

AI.I.III Operadores lógicos, aritméticos y de comparación

AI.I.III.I Operadores lógicos.

Los operadores con los que trabaja FORTRAN® son los siguientes:

Operador	*Significado*	*Definición*
A.AND.B	Y	Resultado True si A y B son verdaderas
A.OR.B	O	resultado true si A o B es verdadera
A.EQV.B	EQUIVALENCIA	resultado true si A=B
A.NEQV.B	NO EQUIVALENCIA	
A.NOT.B	NO EQUIVALENCIA	

Tabla AI.1 Operadores Lógicos.

AI.I.III.II Operadores aritméticos.
Los operadores con los que trabaja FORTRAN® son los siguientes:

Operador	*Significado*
+	Suma
-	Resta
*	Multiplicación
/	División
**	Exponenciación

Tabla AI.2 Operadores Aritméticos.

AI.I.III.III Operadores de comparación.
Los operadores con los que trabaja FORTRAN® son los siguientes:

Operador		*Significado*
*Estilo*1	*Estilo*2	
.EQ.	==	igual
.NE.	/=	diferente
.LT.	<	menor que
.LE.	<=	Menor o igual que
.GT.	>	mayor que
.GE.	>=	Mayor o igual que

Tabla AI.3 Operadores Comparación.

AI.I.IV Jerarquía operacional.

La jerarquía que sigue FORTRAN® y la mayoría de los lenguajes para evaluar las operaciones aritméticas que componen el programa, comienza por evaluar el contenido de los paréntesis que forman una expresión desde adentro hacia fuera, en caso de contener exponentes estos se evalúan de derecha a izquierda, las sumas, restas, multiplicaciones y divisiones se evalúan de izquierda a derecha.

AI.II Estructura de un programa en FORTRAN®.

AI.II.I Sección de declaración.

El tipo de variable debe de ser declarada en la sección de declaración al comienzo del programa, después de la instrucción no ejecutable **PROGRAM.** La instrucción **IMPLICIT NONE** deshabilita la definición implícita de las variables. Cuando se emplea **IMPLICIT NONE** se debe de declarar explícitamente todas las variables utilizadas dentro del programa, aunque se puede omitir, es recomendable utilizarla pues esto permite identificar rápidamente posibles fuentes de error durante la compilación del programa e incluso durante la ejecución de este. La instrucción **IMPLICIT NONE** debe aparecer después de la instrucción **PROGRAM** y antes de cualquier instrucción de declaración

AI.II.II Sección de ejecución.

La sección de ejecución consiste de una o más instrucciones ejecutables que describe las acciones que va a realizar el programa.

Para esto se utilizan algunas instrucciones definidas en FORTRAN®, además se pueden escribir operaciones aritméticas que permiten calcular, modificar o comparar los valores asumidos por las diferentes variables.
Algunas instrucciones definidas para FORTRAN® que se utilizan en esta sección son:

- **PRINT ,*** permite escribir en pantalla algún tipo de mensaje.
- **WRITE (*,*)** se utiliza para escribir en pantalla o en archivos externos del programa.
- **READ (*,*)** se utiliza para asignar valores a las variables, estos valores se pueden leer directamente de la información que introduce el usuario en pantalla o de algún archivo externo al programa.

AI.II.III Sección de terminación.

La sección de terminación consiste de las instrucciones **STOP** y **END PROGRAM,** la instrucción **STOP** detiene el programa, mientras que la instrucción **END PROGRAM** indica que no hay más instrucciones a ejecutar.

En algunos programas que utilizan subrutinas estas se colocan después del **END PROGRAM**, esto se debe a que las subrutinas solo pueden ser utilizadas por y en el programa principal, por lo que a lo largo de este en algún punto especifico se mandan a llamar estas subrutinas, por esto las subrutinas pueden ser ubicadas después de la línea que contiene el **END PROGRAM**, además se pueden crear módulos en donde se almacenen las subrutinas que utilice el programa.

AI.II.IV Compilación y ejecución de un programa.

Una vez que se escribió el código, es necesario compilarlo para así poder ejecutar el programa.

Al compilar el código fuente lo que ocurre es un análisis orientado a encontrar errores, es decir, se verifica que todas las líneas contengan instrucciones compatibles con el lenguaje, se detectan variables no declaradas en la sección correspondiente y que sin embargo se encuentran a lo largo del código, etc.

En el caso de VISUAL FORTRAN® y el de la mayoría de los compiladores se especifican las líneas del código en las que se encuentran estos errores.

Cabe aclarar que existen errores que los compiladores no pueden detectar, pues no están asociados a sintaxis del código, si no que pueden deberse a una interpretación errónea que da el programador a la solución del problema, estos errores generalmente ocurren durante la ejecución del programa, y pueden

deberse a operaciones inválidas como una división entre cero, exceder el orden de los arreglos, etc., con la práctica y la experiencia es mas sencillo detectar los errores que indica el compilador o los que se generan por una mala interpretación.

Para compilar el código solo se necesita dar click en el botón  ubicado en la parte superior del espacio de trabajo de VISUAL FORTRAN®, en la parte inferior de la ventana aparece otra conocida como *output* en donde se muestran las líneas que tienen algún error y de que tipo se trata, la última línea muestra la cantidad de errores y de advertencias, la diferencia entre estos puntos es que si existe por lo menos un error el programa no se podrá ejecutar, mientras que no importa cuantas advertencias existan, el programa puede ser ejecutado, sin embargo es muy probable que el programa se desborde, razón por la que se recomienda depurar el código hasta obtener cero errores y cero advertencias.

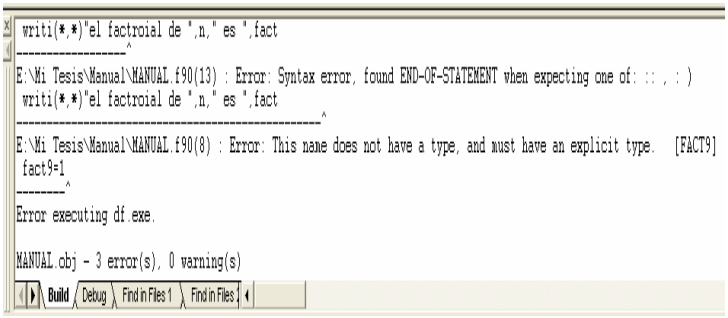


Figura AI.2 Ventana Output.

Si después de compilar el código no se tienen errores ni advertencias se puede ejecutar el programa, para esto basta con hacer click sobre el botón ubicado en la parte superior del espacio de trabajo, esto generara en la carpeta del programa un archivo ejecutable con la extensión exe.

AI.II.V Ejemplo.

El siguiente es un ejemplo sencillo de la estructura de un programa escrito en FORTRAN® 90:

```
PROGRAM PRIMERO
IMPLICIT NONE
        REAL(8)::A,B,C
        WRITE(*,*)'INDICA A'
        READ(*,*)A
        WRITE(*,*)'INDICA B'
        READ(*,*)B
        C=A/B
        WRITE(*,*)C
END PROGRAM
```

Las líneas que contienen la instrucción **WRITE(*,*)** sirven para mostrar en pantalla los mensajes que están entre apóstrofes (‘), también se pueden utilizar comillas

(“), las líneas **READ(*,*)** asignan el valor introducido en pantalla por el usuario a las variables que ahí se indican.

Una vez que se asignaron valores a las variables A y B se asigna el valor a C resultante de la división A entre B mediante **C=A/B** para después mandarlo a imprimir a pantalla. El programa en ejecución es el que se muestra en la imagen.

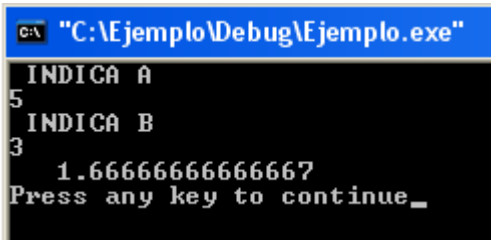


Figura AI.3 Programa en Ejecución.

AI.III Estructuras de control.

Estas estructuras permiten controlar el orden con el que el programa realizará las operaciones indicadas.

AI.III.I Estructuras de decisión.

Por medio de condiciones limitadas mediante comparaciones de una variable con un valor patrón, o entre dos o más variables, se indica al programa qué instrucciones debe o no realizar.

Para esto existe la estructura **IF … THEN … ELSE** cuya estructura en el código escrito en FORTRAN® es:

```
IF (expresiones lógicas, comparativas o ambas) THEN
        instrucciones
        instrucciones
ELSE
        instrucciones
        instrucciones
END IF
```

Si se cumple la condición indicada se realizará el primer bloque de instrucciones y se omitirá el segundo, si no se cumple se omite el primer bloque y se realizán las instrucciones del segundo bloque. Se puede observar que solo se verifica que se cumpla una condición, pero puede darse el caso en que sea necesario verificar mas condiciones para decidir que bloque de instrucciones realizar, cuando se presenta este caso existe la siguiente estructura de decisión:

```
IF (expresión lógica, comparativa o ambas) THEN
        instrucciones
ELSEIF (expresión lógica, comparativa o ambas) THEN
        instrucciones
ELSE
        instrucciones
END IF
```

Otra estructura utilizada en FORTRAN® es **SELECT CASE**, esta estructura permite elegir entre dos o mas opciones, la diferencia principal con el **IF … THEN … ELSE**, es que el **SELECT CASE** solo puede hacer uso de un valor especifico para elegir entre una u otra opción, mientras que con IF el valor que toma la variable para poder decidir puede estar dentro de un rango de valores definidos por el usuario. La estructura es la siguiente:

```
SELECT CASE (expresión)
CASE(case selector 1)
        instrucciones
        instrucciones
CASE(case selector 2)
        instrucciones
        instrucciones
CASE DEFAULT
        instrucciones
        instrucciones
END SELECT
```

El case DEFAULT es opcional, si se utiliza se ejecutará el bloque de instrucciones que le corresponde solo si el valor de expresión esta fuera del rango de todos los case selector.

AI.III.II Estructuras de repetición o ciclos.

Permiten ejecutar un grupo de instrucciones durante un número finito de veces, para esto existen dos tipos de sentencia, los ciclos iterativos y los ciclos condicionados.

La estructura de los ciclos iterativos es la siguiente:

```
DO valor inicial, valor final, incremento
        instrucciones
        instrucciones
END DO
```

La estructura de los ciclos condicionados puede escribirse de dos formas, la primera es:

```
DO WHILE (condición)
        instrucciones
        instrucciones
END DO
```

Mientras la condición se cumpla se repetirán las instrucciones dentro del ciclo, la segunda manera es:

```
DO
        instrucciones
        instrucciones
        IF (expresión lógica, comparativa o ambas) EXIT
END DO
```

Se comienza a ejecutar las instrucciones dentro del ciclo, se realiza una decisión (IF) si se cumple o no la instrucción **EXIT** indica que el ciclo se termina.

AI.III.III Ejemplos de estructuras de decisión y repetición.

Estructuras de decisión.

Ejemplo 1.

```
IF (A>=3) THEN
        C=(A+B)/(A**B)
        ELSEIF (A<=1) THEN
                C=(A+B)**B
        ELSE
                C=A+B
END IF
```

Ejemplo 2.

```
IF (C<=2.5) THEN
        F=D+E
        IF (F>=1.5) THEN
                G=F+C
        ELSE
                G=F-C
        END IF
ELSE
        G=F/C
END IF

WRITE(*,*)C
WRITE(*,*)F
WRITE(*,*)G
```

En este ejemplo se puede observar una estructura **IF (F>=1.5)** anidada, es decir se encuentra dentro de otra estructura **IF** y solo se evaluara si la estructura mas arriba se cumple.

Ejemplo 3.

Un uso común y sencillo del **SELECT CASE** es la creación de un *"menú de opciones"* dentro de un programa, para esto se utiliza un variable que almacenará el número de la opción deseada por el usuario, el siguiente código muestra un pequeño ejemplo de esto:

```
WRITE(*,*)"*****************************************************"
WRITE(*,*)"*                                                   *"
WRITE(*,*)"*                                                   *"
WRITE(*,*)"* ELIJE UNA OPCIÓN:                                 *"
WRITE(*,*)"*                                                   *"
WRITE(*,*)"* 1. SUMA                                           *"
WRITE(*,*)"* 2. RESTA                                          *"
WRITE(*,*)"* 3. MULTIPLICACIÓN                                 *"
WRITE(*,*)"* 4. DIVISIÓN                                       *"
WRITE(*,*)"*                                                   *"
WRITE(*,*)"*                                                   *"
WRITE(*,*)"*****************************************************"
READ(*,*)OPCION

    SELECT CASE (OPCION)
    CASE(1)
        WRITE(*,*)"*****************************************************"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*              S U M A                              *"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*****************************************************"
    CASE(2)
        WRITE(*,*)"*****************************************************"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*              R E S T A                            *"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*****************************************************"
    CASE(3)
        WRITE(*,*)"*****************************************************"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*       M U L T I P L I C A C I O N                 *"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*****************************************************"
    CASE(4)
        WRITE(*,*)"*****************************************************"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*           D I V I S I O N                         *"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*****************************************************"
    CASE DEFAULT
        WRITE(*,*)"*****************************************************"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*    L A  O P C I O N  N O  E X I S T E             *"
        WRITE(*,*)"*                                                   *"
        WRITE(*,*)"*****************************************************"
    END SELECT
```

El usuario deberá decidir entre las cuatro opciones a su disposición, por ejemplo, si elige 1 el programa desplegara el mensaje de suma, si elije 2 el de resta, etc., si elije una opción diferente a las cuatro que se le ofrecen el programa salta a la parte del **CASE DEFAULT** e indica que la opción no existe. Aunque este ejemplo es muy sencillo se pueden construir a partir de esto programas más complejos en los que cada opción realice una gran cantidad de operaciones.

Estructuras de Repetición.

Ejemplo 1.

```
SUMA=0
        DO I=1,N
                SUMA=SUMA+I
        END DO
```

el ciclo realizará la suma de los números enteros de 1 hasta N, donde N puede ser cualquier valor entero positivo.

Ejemplo 2.

```
SUMA=0
        DO I=1,-N,-1
                SUMA=SUMA + I
        END DO
```

el ciclo realizará la suma de los números enteros de 1 hasta -N, donde -N puede ser cualquier valor entero negativo, el -1 indica que el ciclo se moverá hacia atrás.

Ejemplo 3.

```
        FACT=1
        DO I=1,N
                FACT=FACT*I
        END DO
```

El ciclo calcula el factorial de un número N.

AI.IV Arreglos.

En cualquier lenguaje de programación se puede hacer uso de variables que reciben el nombre de arreglos, estos toman la forma de vectores y matrices y se pueden utilizar para almacenar valores provenientes de un sistema de ecuaciones lineales o no lineales, o valores provenientes de tablas de datos necesarios para realizar los cálculos requeridos por un programa.

FORTRAN® no es la excepción y permite el manejo de vectores, matrices e incluso cubos.

AI.IV.I Concepto de vector y matriz en programación.

Al indicar a la computadora la existencia de una variable en forma de vector o matriz lo que se esta haciendo es reservar un espacio de memoria finito en el que se van a almacenar datos, el espacio reservado es la dimensión del vector o matriz, es decir un vector con una dimensión de 5 significa que es capaz de almacenar cinco datos diferentes, una matriz con dimensión de 3*3 puede almacenar nueve datos.

Aunque ese espacio es finito se puede modificar durante la ejecución del programa aumentando o disminuyendo las dimensiones de los arreglos, a esto se le conoce como *"memoria dinámica".*

AI.IV.II Manejo de variables en forma de arreglo.

AI.IV.II.I Declaración de arreglos con dimensión fija.

Los arreglos pueden ser variables de tipo entero y real, con dimensión fija o dinámica y se declaran mediante la siguiente línea de código:

```
REAL(8),DIMENSION(5,5)::A
REAL(8),DIMENSION(4)::B
```

Estas líneas indican que la variable A es de tipo real y doble precisión, la sentencia *DIMENSION(5:5)* indica que se trata de un arreglo de tipo matricial con una dimensión de 5*5, es decir, puede almacenar 25 elementos, la siguiente línea indica que la variable B es de tipo real de doble precisión, la sentencia *DIMENSION(4)* indica que se trata de un arreglo en forma de vector con dimensión de 4 elementos.

AI.IV.II.II Declaración de arreglos con dimensión variable (memoria dinámica).

Si se desea que la dimensión de los arreglos sea variable y se modifique durante la ejecución del programa la declaración se debe realizar de la siguiente forma:

```
REAL(8),ALLOCATABLE,DIMENSION(:,:)::A
REAL(8),ALLOCATABLE,DIMENSION(:)::B
```

Estas líneas indican el tipo de variable y su precisión, adicionalmente la sentencia ALLOCATABLE,DIMENSION(:,:) indica que la variable A es de tipo matricial con dimensión indefinida, esta se asignara durante la ejecución del programa, de igual forma ocurre con la variable B solo que en este caso se trata de un vector.

AI.IV.II.III Memoria dinámica.

Una vez que se han declarado los arreglos de manera dinámica es necesario indicar al programa durante la ejecución la dimensión que estos tomarán, para estos es necesario hacer uso de una o dos variables, según sea el caso, de tipo entero que definan la dimensión de los arreglos, estas variables enteras deberán leerse durante la ejecución del programa.

Si los valores enteros que definen la dimensión de los arreglos ya ha sido introducida, la línea de código que permite dimensionar los arreglos es la siguiente:

```
ALLOCATE(A(N,M))
ALLOCATE(B(N),X(N))
```

Con esto se indica que a la variable A hay que alojar las variables enteras N y M para definir que esta es un arreglo matricial de dimensión N*M mientras a los vectores B y X hay que alojar la variable N para que su dimensión sea de tamaño N.

Si se trata de un programa en el que se puede llegar a utilizar mas de una vez los arreglos con diferentes datos y dimensiones es necesario desalojar las variables que dan la dimensión a los arreglos, esto se realiza mediante la siguiente línea:

```
DEALLOCATE (A)
DEALLOCATE (B,X)
```

Esto significa que las variables A, B, X vuelven a quedar vacías y sin dimensión, por lo que se tendrían que dimensionar una vez más.

AI.IV.II.IV Ejemplo.

```
PROGRAM MATRIZ
IMPLICIT NONE
      INTEGER::I,J                                  !Contadores
      INTEGER::N,M                                  !Dimensión arreglos
      REAL(8),ALLOCATABLE,DIMENSION(:,:)::A,B,C     !Arreglos
      REAL(8),ALLOCATABLE,DIMENSION(:)::X           !Arreglos
            WRITE(*,*)'INDICA EL # DE RENGLONES DE A, B y C, Y EL ORDEN DE X'
            READ(*,*)N
            WRITE(*,*)'INDICA EL # DE COLUMNAS DE A, B y C'
            READ(*,*)M
            ALLOCATE(A(N,M),B(N,M),C(N,M)
            ALLOCATE(X(N))
            ...
            ...
            ...
END PROGRAM
```

AI.IV.III Lectura e impresión de datos.

Para leer los datos que integran un arreglo es necesario las estructuras de repetición vistas en la sección AI.III.II, adicionalmente debido a la complejidad y monotonía que puede llegar a tener escribir directamente en la pantalla del programa los datos que conforman el arreglo se vuelve primordial realizar la lectura de datos desde archivos externos al programa, adicionalmente imprimir

resultados en un archivo externo permite utilizar estos en otros programas como EXCEL® para obtener gráficas, realizar análisis estadísticos, etc.

AI.IV.III.I Lectura en pantalla.
Una vez que la dimensión de los arreglos ha que dado establecida, ya sea de forma fija desde la escritura del código del programa o de manera dinámica en el tiempo de ejecución de este, la lectura de los datos de un arreglo en forma de vector se hace de manera sencilla:

```
DO I=1,N
        READ(*,*)B(I)
END DO
```

Esto significa que se comenzara a leer el valor del elemento B(1), siguiendo con el elemento B(2), el B(3), hasta el N, esto es en caso de ser dinámico, si se trata de un arreglo de dimensión fija la lectura se realizará hasta que se alcance dicha dimensión, en el código solo será necesario escribir en lugar de N el valor fijo de la dimensión del arreglo.
Para leer los datos que formarán un arreglo matricial es un poco mas complicado pues es necesario el uso de un par de ciclos, uno anidado dentro del otro, esto tiene la finalidad de fijar con el primer ciclo los renglones de la matriz y con el segundo moverse sobre los elementos de estos, o en el caso inverso, fijar las columnas y moverse sobre los elementos de estas.

```
DO I=1,N
        DO J=1,M
                READ(*,*)A(I,J)
        END DO
END DO
```

En este caso se comienza a leer los elementos del primer renglón, es decir, cuando I=1 se leen los elementos A(1,J=1), A(1,J=2), ..., A(1,J=M), cuando I=2 se leen los elementos A(2,J=1), A(2,J=2), ..., A(2,J=M), y así sucesivamente hasta que I=N.

AI.IV.III.II Lectura de archivos externos.
Este tipo de lectura tienen una gran importancia pues algunos softwares comerciales la utilizan para recibir la información que necesitan para realizar las tareas para las que fueron diseñados.

Para esto se utilizan archivos de texto en los que se debe escribir la información siguiendo el formato requerido por el programa, se acostumbra escribir estos archivos de texto haciendo uso del programa Bloc de notas® incluido en el sistema operativo WINDOWS®.

Es importante entender que la lectura de archivos externos no es exclusiva de los arreglos vectoriales o matriciales, se puede leer el valor de cualquier variable

desde ellos, sin embargo es para los arreglos en donde se encuentra su mayor beneficio.

En la sección anterior se trató la lectura de los elementos de arreglos desde la pantalla de la computadora, para realizar esta lectura basta con que el usuario introduzca el valor y presione la tecla enter tantas veces como sea necesario, en este caso se recomienda que el archivo externo tenga las siguientes características:

Si se trata de la lectura de los elementos de un vector estos deberán escribirse en el archivo en forma de columna.

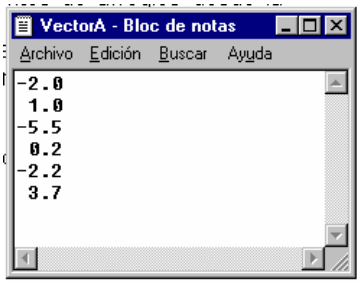


Figura AI.4 Archivos de Datos para Vector.

Si se trata de la lectura de los elementos de una matriz estos deberán estar escritos en forma de renglón – columna.

Figura AI.5 Archivos de Datos para Matriz.

Antes de tratar de leer del archivo externo es necesario abrirlo mediante la siguiente línea de código:

```
OPEN(UNIT=10,FILE="C:\VectorB.txt",STATUS="UNKNOWN")
OPEN(UNIT=11,FILE="C:\MatrizA.txt",STATUS="UNKNOWN")
```

Es necesario asignar un número al archivo para identificarlo, este número es el que se indica con *UNIT=11*, se puede utilizar cualquier número mayor o igual a 10 para numerar al archivo, la ubicación del archivo dentro de la computadora o algún dispositivo externo (Disquete, CD, Memoria USB, etc.) se indica con *FILE="C:\MatrizA.txt"*, el programa se dirigirá a esta ruta para localizar el archivo, en caso de no encontrar el archivo o el dispositivo en donde se supone se encuentra este el programa no funcionara enviando un error, esto solo ocurre en la lectura, la extensión *.txt* del archivo no es obligatoria, se puede utilizar *.dat*, *.doc*, etc., la sentencia *STATUS="UNKNOWN"* se revisará mas adelante.

Igual que con la memoria dinámica, en que es necesario desalojar las variables para poder redimensionar los arreglos con nuevos valores, e necesario cerrar los

archivos en caso de ser necesario volver a acceder a ellos, esto se hace mediante la siguiente sentencia:

```
CLOSE(10)
```

Así, para leer del archivo los valores del vector B la sintaxis es la siguiente:

```
DO I=1,N
        READ(11,*)B(I)
END DO
```

Y para la Matriz A:

```
DO I=1,N
        READ(11,*)A(I,1:M)
END DO
```

El número del archivo se coloca en lugar del primer asterisco de los dos que acompañan la instrucción *READ(*,*)*, el funcionamiento en el caso de la matriz es exactamente el mismo, solo cambia la manera de escribirlo, si se desea, se puede pensar como un caso especial de la sentencia *DO*.

AI.IV.III.III Impresión en pantalla.

Para realizar la impresión de los valores contenidos en los arreglos vectoriales y matriciales los ciclos *DO* son exactamente los mismos solo es necesario cambiar la instrucción *READ(*,*)* por *WRITE(*,*),* quedando las líneas de código para un vector:

```
DO I=1,N
        WRITE(*,*)B(I)
END DO
```

Para una matriz:

```
DO I=1,N
        DO J=1,M
                WRITE(*,*)A(I,J)
        END DO
END DO
```

Estas últimas líneas imprimen los elementos de la matriz en forma de columna, si se desea que aparezcan en pantalla como matriz las líneas son:

```
DO I=1,N
        WRITE(*,*)A(I,1:M)
END DO
```

AI.IV.III.IV Impresión en archivos externos.

Para imprimir en archivos es necesario, igual que en la lectura, abrir el archivo deseado mediante la línea:

```
OPEN(UNIT=10,FILE="C:\Resultados.txt",STATUS="UNKNOWN")
```

Aquí la sentencia STATUS="UNKNOWN" es más importante que en la lectura, pues en caso de no existir el archivo en la ruta especificada el programa lo creará automáticamente.

Las líneas para imprimir los datos contenidos en un arreglo dentro de un archivo externo son las mismas que en la de impresión en pantalla, solo se necesita agregar él número del archivo en la parte reservada para ello.

```
DO I=1,N
    WRITE(10,*)B(I)
END DO
```

Para una matriz:

```
DO I=1,N
    WRITE(10,*)A(I,1:M)
END DO
```

AI.IV.III.V Ejemplo.

```
PROGRAM MATRIZ
IMPLICIT NONE
    INTEGER::I,J                                    !Contadores
    INTEGER::N                                      !Dimensión arreglos
    REAL(8),ALLOCATABLE,DIMENSION(:,:)::A,B         !Arreglos
        OPEN(UNIT=10,FILE="C:\Matrixa.txt",STATUS="UNKNOWN")
        OPEN(UNIT=11,FILE="C:\MatrixB.txt",STATUS="UNKNOWN")
        OPEN(UNIT=12,FILE="C:\Matrizc.txt",STATUS="UNKNOWN")
        READ(10,*)N
        ALLOCATE(A(N,N),B(N,N))
        DO I=1,N
            READ(10,*)A(I,1:N)
        END DO
        DO I=1,N
            READ(11,*)B(I,1:N)
        END DO
        ...
        ...
        DO I=1,N
            WRITE(12,*)C(I,1:N)
        END DO
        DEALLOCATE(A,B,C)
```

```
        CLOSE(10)
        CLOSE(11)
        CLOSE(12)
END PROGRAM
```

En este ejemplo la lectura de la dimensión N de los arreglos se hace desde el archivo Matriza.txt, la lectura se realiza línea por línea, entonces una vez que el programa lee el valor de N cambia de línea, es en esta que se encuentra el primer renglón de la matriz A dentro del archivo.

AI.IV.IV Operaciones con arreglos.

Los arreglos pueden utilizarse para hacer cualquier tipo de operación aritmética permitida en FORTRAN®, dependerá del problema que se pretenda resolver y del programador la forma en que estas se realicen.
Además los arreglos pueden interactuar con variables sencillas en cualquier momento.

Cuando interactuan aritméticamente dos variables de tipo arreglo, las operaciones se realizán elemento a elemento, contradiciendo las reglas matemáticas comúnmente conocidas, es decir, los arreglos aunque tienen formas vectoriales o matriciales no son operadas bajo dichas reglas.

AI.IV.IV.I Suma de Matrices en FORTRAN®.

La suma de matrices matemáticamente hablando se realiza elemento a elemento, por lo que no existe diferencia con lo que realiza la computadora:

```
PROGRAM SUMA
IMPLICIT NONE
    INTEGER::I,J                                    !Contadores
    INTEGER::N                                      !Dimensión arreglos
    REAL(8),ALLOCATABLE,DIMENSION(:,:)::A,B,C       !Arreglos
        OPEN(UNIT=10,FILE="C:\Matrixa.txt",STATUS="UNKNOWN")
        OPEN(UNIT=11,FILE="C:\MatrixB.txt",STATUS="UNKNOWN")
        READ(10,*)N
        ALLOCATE(A(N,N),B(N,N),C(N,N))
        DO I=1,N
            READ(10,*)A(I,1:N)
        END DO
        DO I=1,N
            READ(11,*)B(I,1:N)
        END DO
        DO I=1,N
            DO J=1,N
                C(I,J)=A(I,J)+B(I,J)
            END DO
        END DO
        DO I=1,N
```

```
            WRITE(12,*)C(I,1:N)
        END DO
        DEALLOCATE(A,B,C)
        CLOSE(10)
        CLOSE(11)
        CLOSE(12)
END PROGRAM
```

AI.IV.IV.II Multiplicación de matrices en FORTRAN®.

Según el álgebra la multiplicación de dos matrices A y B debe realizarse multiplicando los elementos A(i,k) por los elementos B(k,j) y sumar el producto de cada una de estas multiplicaciones generando un nuevo elemento C(i,j) de la matriz resultado, es lo que se conoce como renglón por columna.

Sin embargo al programar y solamente expresar el producto A(i,j) * B(i,j) se obtendrá una matriz C(i,j) cuyos elementos son muy diferentes de los que se obtendrían respetando el álgebra de matrices.

AI.IV.IV.III División de matrices FORTRAN®.

La división de matrices no esta definida matemáticamente hablando, sin embargo al programarla lo único que ocurre es que se divide el elemento A(i,j) entre su correspondiente B(i,j).

AI.V Programación estructurada.

En ocasiones durante un programa es necesario realizar un mismo cálculo varias veces, o puede ocurrir que un programa necesite de otro previamente elaborado, para esto existen los conceptos de funciones, módulos y subrutinas.

Un ejemplo sencillo son las permutaciones y combinaciones, ambas recurren al cálculo del factorial de un número, por lo que el programa que permita calcular el factorial se puede incluir dentro de una subrutina o un modulo.

AI.V.I Módulos.

En los módulos se pueden agrupar funciones y subrutinas que se utilicen constantemente en uno o varios programas.

Para insertar un modulo se debe dar click en ***"New"*** dentro del menú ***"File"***, con esto aparece la ventana que se muestra en la imagen, aquí se debe seleccionar la opción ***"Fortran Free Format Source File"***, indicar un nombre para el archivo correspondiente al modulo seleccionar la casilla de verificación ***"Add to Project"***, esto agregara una hoja de codigo nueva a la carpeta ***"Source Files"***.

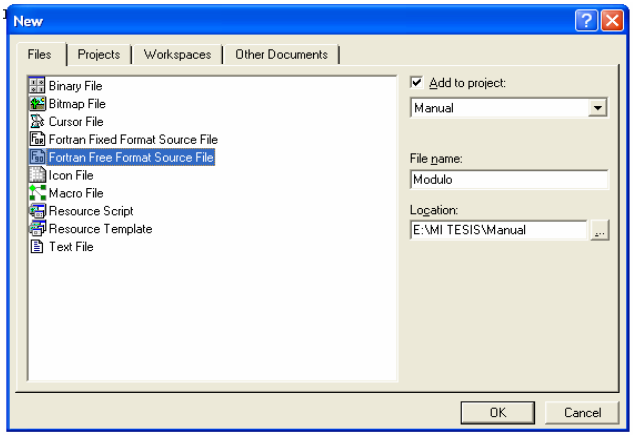


Figura AI.6 Insertar Módulo.

En la hoja de código el modulo debe tener las siguientes líneas al principio y final:

```
MODULE nombre
INTERFACE
…
…
END INTERFACE
END MODULE nombre
```

Dentro de las líneas anteriores se pueden incluir las funciones y subrutinas que sean necesarias.

Para que el programa principal reconozca el modulo es necesario incluir al principio del código la siguiente línea:

```
PROGRAM PRINCIPAL
USE nombre
IMPLICIT NONE
…
…
END PROGRAM
```

Cabe señalar que para utilizar subrutinas y funciones no es necesario incluir módulos, se pueden insertar directamente en el programa principal, sin embargo crear un modulo permite utilizar este en cualquier programa se realice.

AI.V.II Funciones.

Una función devuelve un solo valor para una variable dada, este depende de las variables que se relacionan con la función, puede o no estar dentro de un modulo.

En el caso de que se tenga una función dentro de un modulo la sintaxis en el programa principal es la siguiente:

```
PROGRAM PRINCIPAL
USE nombre
IMPLICIT NONE
…
WRITE(*,*)Pb(Rp,T,DENR_O,DENR_G)
…
END PROGRAM
```

La línea *WRITE(*,*)Pb(Rp,T,DENR_O,DENR_G)* indica que se escriba el valor de la variable Pb que depende de las variables Rp, T, DENR_O y DENR_G, entendiendo que los valores de estas últimas debieron leerse o calcularse previamente.

Ya dentro del módulo se debe escribir:

INICIO DEL MÓDULO.
```
MODULE nombre
INTERFACE
…
…
FUNCTION Pb(Rp,T,DENR_O,DENR_G)
IMPLICIT NONE
REAL(8),INTENT(IN)::Rp,T,DENR_O,DENR_G
REAL(8)::Pb
END FUNCTION
…
END INTERFACE
END MODULE nombre
```
FIN DEL MÓDULO.

```
FUNCTION Pb(Rp,T,DENR_O,DENR_G)
IMPLICIT NONE
REAL(8),INTENT(IN)::Rp,T,DENR_O,DENR_G
REAL(8)::Pb
Pb=18.2*(((Rp/DENR_G)**0.83)*(10**(0.00091*T-0.0125*(141.5/DENR_O-131.5)))-1.4)
RETURN
END FUNCTION
```

Dentro del código del módulo solo se declaran las variables a utilizar en la función, en la línea *REAL(8),INTENT(IN)::Rp,T,DENR_O,DENR_G* la palabra *INTENT(IN)* indica si la o las variables son de entrada, en caso de ser de salida se utiliza la palabra *INTENT(OUT)* y si es de ambos tipos *INTENT(INOUT)*, después, al final del modulo se repiten estos pasos y se define la operación con la que se evalúa la función.

AI.V.III Subrutinas.

Las subrutinas a diferencia de las funciones pueden ser programas completos en donde se calculen una o más variables.

El procedimiento para tener una subrutina dentro de un módulo es exactamente el mismo que para una función, solo que se sustituye la palabra *FUNCTION* por *SUBROUTINE* la diferencia esta en el programa principal, pues aquí es necesario mandar llamar a la subrutina indicando las variables con las que se va a trabajar.

```
PROGRAM PRINCIPAL
USE nombre
IMPLICIT NONE
…
CALL SUBROUTINE FACTORIAL(n,FACT)
…
END PROGRAM
```

Si no se desea utilizar un módulo que contenga las funciones y las subrutinas se pueden colocar dentro del mismo programa principal al término del código de este:

INICIO DEL PROGRAMA.

```
PROGRAM PRINCIPAL
IMPLICIT NONE
…
WRITE(*,*)Pb(Rp,T,DENR_O,DENR_G)
CALL SUBROUTINE FACTORIAL(n,FACT)
…
END PROGRAM
```

FIN DEL PROGRAMA.

```
FUNCTION Pb(Rp,T,DENR_O,DENR_G)
IMPLICIT NONE
REAL(8),INTENT(IN)::Rp,T,DENR_O,DENR_G
REAL(8)::Pb
Pb=18.2*(((Rp/DENR_G)**0.83)*(10**(0.00091*T-0.0125*(141.5/DENR_O-131.5)))-1.4)
RETURN
END FUNCTION

SUBROUTINE FACTORIAL(n,FACT)
IMPLICIT NONE
INTEGER::i,n,FACT
WRITE(*,*)"INDICA EL NUMER DEL CUAL DESEAS EL FACTORIAL"
READ(*,*)n
FACT=1
DO i=1,n
FACT=FACT*i
END DO
WRITE(*,*)"EL FACTROIAL DE ",n," ES ",FACT
END SUBROUTINE FACTORIAL
```

AI.V.IV Sentencia contains.

La sentencia contains permite insertar funciones o subrutinas desde el programa principal, evitando el uso de un módulo o la declaración de todas las variables involucradas.

INICIO DEL PROGRAMA.

```
PROGRAM PRINCIPAL
IMPLICIT NONE
…
WRITE(*,*)Pb
CALL SUBROUTINE FACTORIAL
…
…
…
```

```
CONTAINS
FUNCTION Pb(Rp,T,DENR_O,DENR_G)
Pb=18.2*(((Rp/DENR_G)**0.83)*(10**(0.00091*T-0.0125*(141.5/DENR_O-131.5)))-1.4)
END FUNCTION

SUBROUTINE FACTORIAL(n,FACT)
WRITE(*,*)"INDICA EL NUMER DEL CUAL DESEAS EL FACTORIAL"
READ(*,*)n
FACT=1
DO i=1,n
FACT=FACT*i
END DO
WRITE(*,*)"EL FACTROIAL DE ",n," ES ",FACT
END SUBROUTINE FACTORIAL
END PROGRAM
```

FIN DEL PROGRAMA.

Como se puede observar no es necesario declarar nuevamente las variables, estas se deben declarar en la sección destinada para ello dentro del programa principal.

AI.V Recomendación.

Para aprender a programar es necesario practicar durante horas con diferentes problemas, pues esto permitirá conocer la lógica de programación, cuando esto se logra, se es capaz de programar en casi cualquier lenguaje, solo es necesario conocer la sintaxis específica de cada uno de ellos y aplicar los conocimientos adquiridos.

Este pequeño manual no es suficiente para desarrollar dichas habilidades, solo pretende ser una guía para quien comienza a programar, por lo que se recomienda después de estudiarlo elaborar un programa capaz de realizar la multiplicación de matrices definida por el algebra y el método de Newton – Raphson pues aunque para un principiante son complejos, si se logran realizar permiten entender la lógica de programación.

AII. PROGRAMACIÓN EN VISUAL BASIC 6.0®.

AII.I Elementos básicos de programación en VISUAL BASIC 6.0®.

AII.I.I Inicio de un programa en VISUAL BASIC 6.0®.

Para comenzar a diseñar un programa en VISUAL BASIC 6.0® se recomienda crear una carpeta en donde almacenar todas las partes que van a formar parte de este, pues dependiendo de la aplicación desarrollada se pueden tener una gran cantidad de formularios y módulos.

Figura AII.1 Nuevo Proyecto.

Una vez que se ha iniciado VISUAL BASIC 6.0® aparece un asistente en el que se presentan varios tipos de proyecto, en este caso el que se utilizará es el nombrado como “EXE estándar”, con solo dar doble click o seleccionarlo y presionar “Abrir” se podrá comenzar un nuevo proyecto.

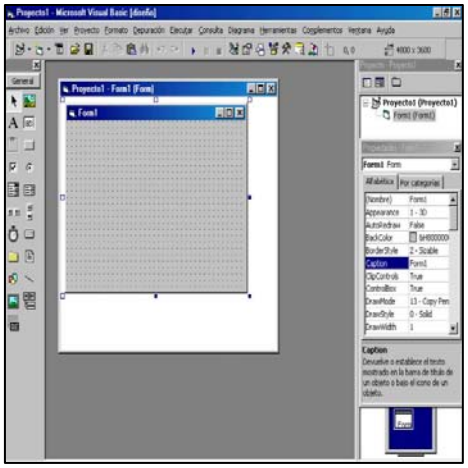


Figura AII.2 Espacio de Trabajo.

Después de esto aparecerá el espacio de trabajo en donde se puede diseñar el ambiente gráfico de la aplicación. En este se pueden identificar las ventanas de “Explorador de Proyectos”, “Propiedades”, “Cuadro de Herramientas”, además de las barras de menú y herramientas presentes en cualquier aplicación de WINDOWS®. La ventana de “Explorador de Proyectos” muestra los formularios y módulos que componen la aplicación, en la ventana de propiedades se muestran, como su nombre lo indica, las propiedades asociadas a cada uno de los objetos.

El “Cuadro de Herramientas” contiene cada uno de los objetos que se pueden insertar en la aplicación, aunque inicialmente solo se encuentran algunos de los objetos, se pueden agregar mas en cualquier momento.

La parte más importante en el espacio de trabajo en lo que se refiere al diseño gráfico de la aplicación es el “Formulario” pues es dentro de este en donde se van insertando los objetos que van a formar parte de la aplicación. Los Formularios son objetos que tienen propiedades que se pueden modificar según las necesidades de la aplicación.

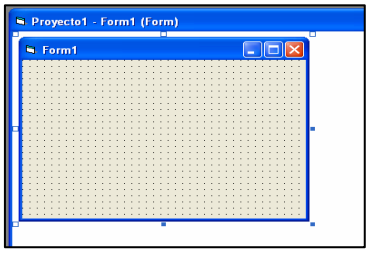


Figura AII.3 Formulario.

AII.I.II Objetos.

Para poder elaborar una aplicación básica solo se necesita de tres objetos:

1. Botones de Comando (CommandButton): En estos objetos se acostumbra asociar las líneas de código que permitan realizar las operaciones deseadas, a partir de aquí y por simplicidad solo se nombraran como botones.

2. Cajas de Texto (TextBox): Este tipo de objetos se utilizan para que el usuario inserte la información necesaria para la ejecución de la aplicación.

3. Etiquetas (Label): Se utilizan para poner mensajes, encabezados o rótulos en la aplicación o para imprimir los resultados obtenidos durante la ejecución.

AII.I.II.I Como insertar objetos.

Los objetos se encuentran agrupados generalmente en la barra de objetos ubicada a la izquierda de la ventana de trabajo.

Figura AII.4 Barra de Objetos.

Para insertar un objeto dentro de un formulario basta con dar click en el deseado y después dibujarlo de manera manual con las dimensiones requeridas, o dar doble click en el objeto, de esta forma automáticamente se inserta en el formulario con un tamaño preestablecido, aunque este se puede modificar.

AII.I.II.II Propiedades.

Cada objeto tiene propiedades, algunas de estas son compartidas por varias clases de objetos, por ejemplo, los formularios, los botones de comando y las etiquetas comparten la propiedad “Caption”, prácticamente todos los objetos comparten las propiedades “Height” y “Width” que se refieren al alto y ancho de un objeto.

Las propiedades se pueden modificar durante el diseño de la aplicación y también durante la ejecución de esta.

AII.I.III Código.

Desde el punto de vista de la programación estructurada el código contiene la serie de instrucciones que permitirán hacer las operaciones necesarias para resolver el problema para el cual se diseño la aplicación, sin embargo para la programación orientada a objetos, y en este caso para VISUAL BASIC 6.0® el código también se orienta al funcionamiento de los objetos y a modificar sus propiedades en el tiempo de ejecución.

Como se mencionó anteriormente un mensaje esta asociado con un procedimiento, de tal forma que cuando un objeto recibe un mensaje la respuesta a ese mensaje es ejecutar el procedimiento asociado, los métodos también se conocen como eventos.

Una vez que se inserta un objeto dentro del formulario, basta con dar doble click sobre este para que aparezca la pantalla del código, cada objeto tiene asociado un evento por default, sin embargo estos eventos se pueden modificar eligiendo de una gran variedad de estos y según las necesidades. Para volver a la pantalla de diseño basta con presionar el botón  "Ver objeto" ubicado en la parte superior del explorador de proyectos y para volver al código se debe dar click en el botón  "código" ubicado en el mismo lugar.

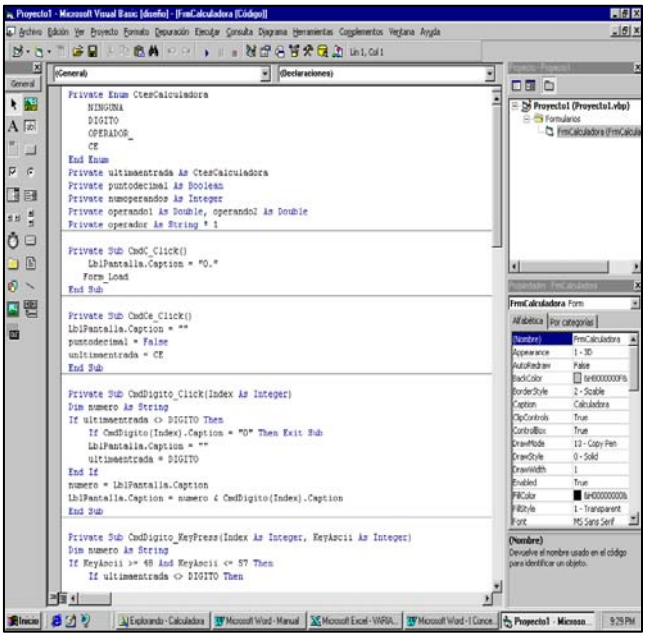


Figura AII.5 Pantalla de Código.

AII.I.IV Operadores lógicos, aritméticos y de comparación

AII.I.IV.I Operadores lógicos.

Los operadores con los que trabaja VISUAL BASIC 6.0® son los siguientes:

Operador	*Significado*
Not	**Negación**
And	**Y**
Or	**O**
Eqv	**Equivalencia**

Tabla AII.1 Operadores Lógicos.

AI.I.IV.II Operadores aritméticos.

Los operadores con los que trabaja VISUAL BASIC 6.0® son los siguientes:

Operador	*Significado*
+	Suma
-	Resta
*	Multiplicación
/	División
^	Exponenciación

Tabla AII.2 Operadores Aritméticos.

AI.I.IV.III Operadores de comparación.

Los operadores con los que trabaja VISUAL BASIC 6.0® son los siguientes:

Operador	*Significado*
=	Igual
<>	Diferente
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual

Tabla AII.3 Operadores Comparación.

AII.I.V Ejemplo.

El siguiente es un ejemplo sencillo que permitirá observar lo expuesto hasta el momento.

AII.I.V.I Resumen de objetos.

Objetos									
Propiedades	Formulario	Etiqueta	Etiqueta	Etiqueta	Etiqueta	Caja Texto	Caja Texto	Caja Texto	Botón Comando
Name	FrmPrimer	Label1	Label2	Label3	LblResultado	TxtA	TxtB	TxtC	CmdCalcular
Caption	Programa	Valor de A	Valor de B	Valor de C	Vacío	N/A	N/A	N/A	Calcular
Text	N/A	N/A	N/A	N/A	N/A	Vacío	Vacío	Vacío	N/A
Height	4035	495	495	495	495	495	495	495	495
Width	5175	1215	1215	1215	4695	1215	1215	1215	1815

Tabla AII.4 Objetos para Ejemplo.

Las abreviaturas Frm, Lbl, Txt y Cmd son solo prefijos que se anteponen al nombre de los objetos para facilitar la programación, esto es solo una recomendación que en caso de ignorarla no provoca ningún tipo de problema con el programa.

AII.I.V.II Código de la aplicación.

```
Private Sub CmdCalcular_Click()
   a = Val(TxtA.Text)
   b = Val(TxtB.Text)
   c = Val(TxtC.Text)
   d = (a + b) * c
   LblResultado.Caption = d
End Sub
```

AII.I.V.III Explicación.

El código se asocia al botón de comando *"CmdCalcular"* y este se relaciona con el evento *"Click"*, la primera y última línea se insertan automáticamente al dar doble click sobre el objeto.

En la segunda línea se indica que la variable *a* tomará el valor de la propiedad *"Text"* de la caja de texto *"TxtA"*, se repite para los valores de las variables *b* y *c*.

En la quinta línea el valor de la variable *d* se calcula con la operación *(a+b)*c*. Por último la propiedad *"Caption"* de la etiqueta *"LblResultado"* se modifica durante la ejecución de la aplicación imprimiendo el valor calculado para la variable *d*.

Así, cuando el usuario introduzca los valores que desee para las variables a, b y c, podrá conocer el valor de d con solo dar un click sobre el botón Calcular.

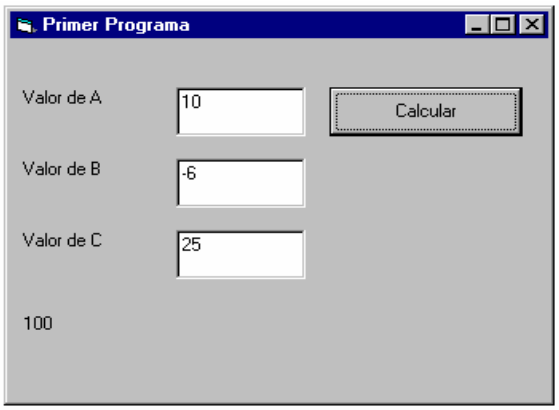


Figura AII.6 Ejemplo en Ejecución.

AII.I.VI Ejecución del programa.

Una vez que se diseño la parte gráfica del programa y se escribió el código asociado a cada objeto hay que ejecutarlo, para esto solo hace falta dar click en el botón ▶ ubicado en la barra de herramientas de la parte superior de la ventana de diseño. A diferencia de VISUAL FORTRAN®, en VISUAL BASIC 6.0® no es necesario compilar el programa antes de ejecutarlo, en caso de tener errores estos se detectaran en el momento de ejecutarlo, esto ocurre por que se asume que solo el programador tiene acceso al diseño de la aplicación y podrá corregir todos los errores que se puedan llegar a presentar.

Al usuario final solo se le debe proporcionar el archivo .exe, es decir el ejecutable de la aplicación, para crear este archivo es necesario dar click en el menú de archivo y dar click en la opción de *"Generar Proyecto1.exe"*, en ese momento VISUAL BASIC 6.0® comienza a compilar todo el programa buscando posibles errores que puedan causar problemas al usuario final, en caso de detectar estos no se crea el ejecutable y se indica la ubicación de estos errores.

El programa se puede ejecutar por pasos con solo presionar la tecla F8, con esto inicia la ejecución paso a paso, así se podrá observar como se van ejecutando una por una las líneas del programa, para cambiar de línea hay que presionar nuevamente la tecla F8, basta con colocar el apuntador del ratón sobre la variable deseada para observar el valor que tiene asignado en el momento de la ejecución

AII.II Variables y procedimientos.

AII.II.I Variables.

Una variable contiene un valor que puede modificarse a lo largo de la ejecución de la aplicación por medio de las operaciones

AII.II.I.I Tipos.

Los tipos de variables que se pueden utilizar en VISUAL BASIC 6.0® son:

Tipo	*Descripción*	*Car – Tipo*	*Rango*
INTEGER	ENTERO	%	DE -32,767 A 32,767
LONG	ENTERO LARGO	&	DE -2,147,483,648 A 2,147,483,647
SINGLE	PUNTO FLOTANTE SIMPLE PRECISION	!	DE -3.40E+38 A 3.40E+38
DOUBLE	PUNTO FLOTANTE DOBLE PRECISION	#	DE -1.79E+308 A 1.79E+308
CURRENCY	NÚMERO CON PUNTO DECIMAL FIJO	@	TIPO 922,337,203,685,477.5807
STRING	CADENA DE CARACTERES DE LONGITUD FIJA	NINGUNO	HASTA 64K
STRING	CADENA DE CARACTERES DE LONGITUD VARIABLE	$	HASTA 2^31 CARACTERES APROX
BYTE	CARÁCTER	NINGUNO	DE 0 A 255
BOOLEAN	BOOLEAN	NINGUNO	TRUE O FALSE
VARIANT	CON NÚMEROS	NINGUNO	VALOR NUMERICO INTERVALO DE TIPO DOBLE

Tabla AII.5 Tipos de Variables.

AII.II.I.II Declaración.

VISUAL BASIC 6.0® permite trabajar sin declarar variables, sin embargo esto puede ser motivo de errores graves en la aplicación.
Para forzar la declaración de variables se puede escribir la siguiente línea de código:

Option Explicit

La declaración de una variable se hace de la siguiente forma:

Dim A as Single o *Dim A!*
Dim B as Double o *Dim B#*

Esto significa que la variable *A* es de simple precisión y que la variable *B* es de doble precisión y se pueden declarar escribiendo el nombre completo del tipo o utilizando el “carácter – tipo” asociado a cada tipo de variable.

AII.II.II Procedimientos.

Se pueden clasificar en:

- Procedimientos para manipular cadenas de caracteres.
- Procedimientos para manipular expresiones numéricas.
- Procedimientos varios.

AII.II.II.I Procedimientos para manipular cadenas de caracteres.

Función Str.

Convierte una expresión numérica en una expresión de caracteres. Su sintaxis es:

Nombre variable = **Str**(número)

Función Val.

Da como resultado el valor numérico de una cadena de caracteres. Su sintaxis es:

Número = **Val**(expresión de cadena)

La función Val ignora los espacios en blanco y tabulaciones que puedan haber en el principio de la expresión de cadena.

```
Dim N As Integer
Dim CadenaA As String, CadenaB as String
CadenaA="   123"
CadenaB="   1000 Pts"
N=Val(CadenaA) 'N=123
Print n, Val(CadenaB) 'Escribe: 123  1000
```

AII.II.II.II Procedimientos para manipular expresiones numéricas.

Una expresión numérica puede ser una constante, una variable, una función o un conjunto de constantes, variables y funciones unidas por operaciones.

Funciones trigonométricas.

Estas funciones permiten obtener de manera automática los valores del seno, coseno, tangente y arco tangente. Su sintaxis es:

valor = **Sin**(ángulo)
valor = **Cos**(ángulo)
valor = **Tan**(ángulo)
ángulo = **Atn**(valor)

Donde ángulo es una expresión numérica que indica un ángulo en radianes y valor representa el seno, el coseno, la tangente o el ángulo.

Funciones logarítmica y exponencial.
Log da como resultado el logaritmo base e (ln) y Exp da como resultado el valor de número e elevado a la expresión. Su sintaxis es:

variable = **Log**(expresión)
variable = **Exp**(expresión)

La relación existente entre el logaritmo base e y el logaritmo decimal viene dada por:

$Log_{10}(n)$=**Log**(n)/**Log**(10)

Función Int.
Da como resultado el mayor número entero que sea menor o igual que el valor del argumento. Su sintaxis es:

variable = **Int**(expresión)

Función Abs.
Da como resultado el valor absoluto de una expresión. Su sintaxis es:

variable = **Abs**(expresión)

Función Sqr.
Da como resultado la raíz cuadrada de una expresión. Su sintaxis es:

variable = **Sqr**(expresión)

AII.II.II.III Procedimientos varios.
Función Rnd.
Esta función se utiliza para generar números aleatorios dentro de un intervalo definido. Para generar estos números aleatorios se utiliza la siguiente formula:

Int((limitesup-limiteinf+1)*__Rnd__+limiteinf)

Si se omite la función **Int** que esta al principio de la formula los números aleatorios serán de tipo real.

AII.III Estructuras de control.

Estas estructuras permiten controlar el orden con el que el programa realizará las operaciones indicadas, además en el caso de VISUAL BASIC 6.0® pueden servir para controlar el funcionamiento de los objetos.

AI.III.I Estructuras de decisión.

Por medio de condiciones limitadas mediante comparaciones de una variable con un valor patrón, o entre dos o más variables, se indica al programa que instrucciones debe o no realizar.

Para esto existe la estructura **IF … THEN … ELSE** cuya estructura en el código escrito en VISUAL BASIC 6.0® es:

```
IF expresiones lógicas, comparativas o ambas THEN
        instrucciones
        instrucciones
ELSE
        instrucciones
        instrucciones
END IF
```

Si se cumple la condición indicada se realizará el primer bloque de instrucciones y se omitirá el segundo, si no se cumple se omite el primer bloque y se realizan las instrucciones del segundo bloque. Se puede observar que solo se verifica que se cumpla una condición, pero puede darse el caso en que sea necesario verificar más condiciones para decidir que bloque de instrucciones realizar, cuando se presenta este caso existe la siguiente estructura de decisión:

```
IF expresión lógica, comparativa o ambas THEN
        instrucciones
ELSEIF expresión lógica, comparativa o ambas THEN
        instrucciones
ELSE
        instrucciones
END IF
```

Otra estructura utilizada en VISUAL BASIC 6.0® es **SELECT CASE**, esta estructura permite elegir entre dos o más opciones, la diferencia principal con el **IF … THEN … ELSE**, es que el **SELECT CASE** solo puede hacer uso de un valor específico para elegir entre una u otra opción, mientras que con IF el valor que toma la variable para poder decidir puede estar dentro de un rango de valores definidos por el usuario. La estructura es la siguiente:

```
SELECT CASE expresión
CASE case selector 1
        instrucciones
        instrucciones
CASE case selector 2
        instrucciones
        instrucciones
CASE ELSE
        instrucciones
        instrucciones
END SELECT
```

El CASE ELSE es opcional, si se utiliza se ejecutará el bloque de instrucciones que le corresponde solo si el valor de expresión esta fuera del rango de todos los case selector.

AI.III.II Estructuras de repetición o ciclos.

Permiten ejecutar un grupo de instrucciones durante un número finito de veces, para esto existen dos tipos de sentencia, los ciclos iterativos y los ciclos condicionados.

La estructura de los ciclos iterativos es la siguiente:

```
FOR contador=valor inicial TO valor final STEP incremento
        instrucciones
        instrucciones
NEXT contador
```

La estructura de los ciclos condicionados puede escribirse de dos formas, la primera es:

```
DO
        instrucciones
        instrucciones
LOOP UNTIL (condición)
```

Mientras la condición no se cumpla se repetirán las instrucciones dentro del ciclo, la segunda manera es:

```
DO UNTIL (condición)
        instrucciones
        instrucciones
LOOP
```

AII.IV Arreglos.

En cualquier lenguaje de programación se puede hacer uso de variables que reciben el nombre de arreglos, estos toman la forma de vectores y matrices y se pueden utilizar para almacenar valores provenientes de un sistema de ecuaciones lineales o no lineales, o valores provenientes de tablas de datos necesarios para realizar los cálculos requeridos por un programa.

VISUAL BASIC 6.0® no es la excepción y permite el manejo de vectores y matrices.

AII.IV.I Manejo de variables en forma de arreglo.

AII.IV.I.I Declaración de arreglos con dimensión fija.

Los arreglos pueden ser variables de tipo entero y real, con dimensión fija o dinámica y se declaran mediante la siguiente línea de código:

```
Dim A(5,5) as double
Dim B(4) as double
```

Estas líneas indican que la variable A es de tipo real y doble precisión, la sentencia *Dim A(5:5)* indica que se trata de un arreglo de tipo matricial con una dimensión de 5*5, es decir, puede almacenar 25 elementos, la siguiente línea indica que la variable B es de tipo real de doble precisión, la sentencia *Dim B(4)* indica que se trata de un arreglo en forma de vector con dimensión de 4 elementos.

AII.IV.II.II Declaración de arreglos con dimensión variable (Memoria dinámica).

Si se desea que la dimensión de los arreglos sea variable y se modifique durante la ejecución del programa la declaración se debe realizar de la siguiente forma:

```
Dim A() as double
```

Estas líneas indican el tipo de variable y su precisión con dimensión indefinida, adicionalmente la sentencia *ReDim A(variable,variable)* indica que la variable A es de tipo matricial y con un tamaño definido por la o las variables indicadas entre paréntesis.

AII.IV.II.III Memoria dinámica.

Una vez que se han declarado los arreglos de manera dinámica es necesario indicar al programa durante la ejecución la dimensión que estos tomarán, para estos es necesario hacer uso de una o dos variables, según sea el caso, de tipo entero que defina la dimensión de los arreglos, estas variables enteras deberán leerse durante la ejecución del programa.

Si los valores enteros que definen la dimensión de los arreglos ya ha sido introducida, la línea de código que permite dimensionar los arreglos es la siguiente:

```
ReDim A(N,M)
ReDim B(N)
```

Con esto se indica que a la variable A hay que alojar las variables enteras N y M para definir que esta es un arreglo matricial de dimensión N*M mientras que al vector B hay que alojar la variable N para que su dimensión sea de tamaño N.

AII.V Objetos avanzados.

En la sección AII.I.II se habló sobre los tres objetos básicos para poder realizar una aplicación sencilla, estos fueron los Botones de Comando (CommandButton), Cajas de Texto (TextBox) y Etiquetas (Label), sin embargo existen otros objetos que permiten realizar aplicaciones de mayor calidad gráfica y técnica:

1. Grid (MSFlexGrid): Presenta una tabla estilo EXCEL®, se puede utilizar para trabajar con matrices y vectores o para introducir tablas de datos.
2. Cuadro de Gráficos (PictureBox): En este tipo de objetos se pueden insertar imágenes desde archivo, pero su aplicación más importante es cuando se utilizan para graficar los resultados obtenidos durante la aplicación.
3. Botones de Opción (OptionButton): Cuando se encuentran dos o más de estos botones se puede seleccionar solamente uno, esto permite dar a los usuarios de la aplicación varias alternativas de ejecución.
4. Casillas de Verificación (ChekBox): Cuando se encuentran dos o más de estas casillas se pueden seleccionar una o más de ellas, esto permite dar a los usuarios de la aplicación varias alternativas de ejecución.

Aunque existen otros objetos, los anteriores son suficientes para realizar aplicaciones con muy buena presentación gráfica y excelentes resultados matemáticos.

Para identificar cada uno de estos objetos se recomienda nombrarlos mediante las siguientes claves:

Objeto		*Clave*	*Ejemplo de Nombre*
	CommandButton	Cmd	CmdCalcular
A	Label	Lbl	LblResultado
abl	TextBox	Txt	TxtDensidad
	CheckBox	Chk	ChkNegrita
	OptionButton	Opt	OptDivision
	PictureBox	Pic	PicGrafica
	MSFlexGrid	Grid	GridMatrizA
	Form	Frm	FrmPrincipal

Tabla AII.6 Claves de Objeto.

Cabe aclarar que lo anterior es solo una recomendación que facilita la identificación de cada uno de los objetos colocados dentro de un formulario y que el no seguirla no implica que la aplicación tendrá problemas en su ejecución.

AII.V.I MSFlexGrid.

Se trata de un objeto que permite tener una tabla del estilo de una hoja de cálculo, sin embargo cabe aclarar que al insertarla solo sirve como medio para imprimir resultados, el usuario no es capaz de interactuar con ella, es por esto que se debe introducir el código que permita realizar más tareas con ella, en este trabajo solo se utilizará como tabla de resultados o como una tabla en la que el usuario pueda insertar una serie de datos necesarios para la ejecución del programa.

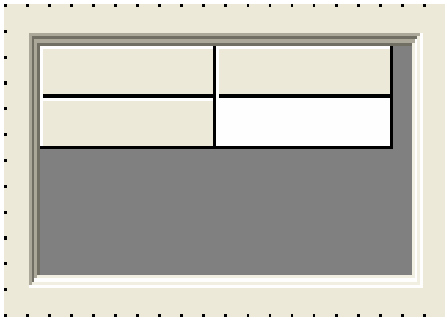


Figura AII.7 MSFlexGrid.

Este objeto no se encuentra inicialmente en la barra de objetos por lo que es necesario insertarlo, para esto basta con hacer click en el botón alterno del ratón sobre la parte vacía de la barra de objetos, escoger la opción de componentes, aparece un asistente, en la pestaña controles se encuentran todos los objetos que pueden agregarse, en este caso hay que localizar el ***"Microsoft FlexGrid Control 6.0"***, seleccionarlo y dar click en aplicar, esto colocará el icono  que corresponde al MSFlexGrid.

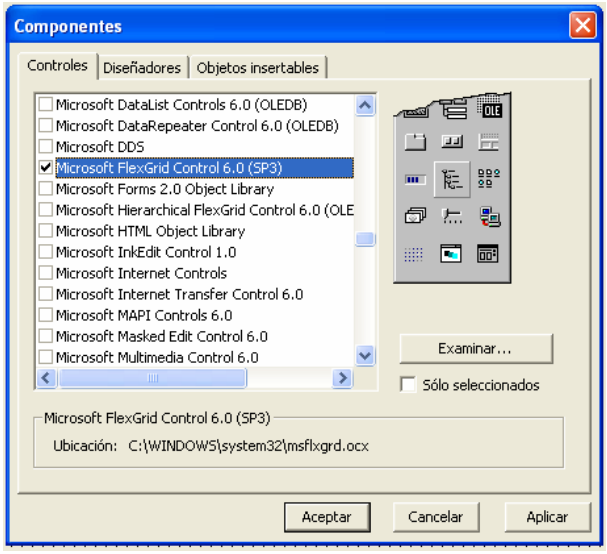


Figura AII.8 Componentes.

AII.V.I.I Propiedades del MSFlexGrid.

Para poder interactuar con este objeto existen una gran cantidad de propiedades que se pueden modificar en tiempo de diseño o de ejecución, algunas de ellas son:

1. Rows: esta propiedad define el número de renglones que tendrá la tabla.
2. FixedRows: define el número de renglones fijos, estos corresponden a las celdas de color gris, y que bajo ninguna circunstancia pueden ser modificadas por el usuario, estos renglones fijos deben ser como mínimo uno menos que el valor de Rows.
3. Cols: define el número de columnas.
4. FixedCols: define el número de columnas fijas, estas corresponden a las celdas de color gris, y que bajo ninguna circunstancia pueden ser

modificadas por el usuario, estas columnas fijas deben ser como mínimo una menos que el valor de Cols.

5. TextMatrix(row,col): sirve para escribir o leer valores de la celda indicada entre paréntesis.

6. Text: se utiliza para afectar el texto escrito en cualquier celda sin importar su ubicación por medio de renglones y columnas.

AII.V.I.II Escritura en MSFlexGrid.

Existen básicamente dos maneras de escribir en un Grid de VISUAL BASIC 6.0®, la primera es dando libertad al usuario de introducir la información requerida mediante el teclado y la segunda por medio de código en tiempo de ejecución.

AII.V.I.II.I Escritura mediante teclado.

En este caso es necesario utilizar el evento ***"KeyPress"***, que como lo indica su nombre responde al presionar cualquier carácter del teclado, además de algunos botones como el de suprimir, la barra espaciadora, el retroceso, etc., uso de la variable ***"KeyAscii"*** declarada automáticamente por el evento ***"KeyPress",*** los valores asociados a dicha variable son los correspondientes al código ASCII de cada carácter.

El código ASCII es un valor numérico asociado a cada carácter que se puede escribir por medio del teclado, por ejemplo, presionando y manteniendo la tecla ALT y en el teclado numérico presionar 64, se obtiene el símbolo @, repitiendo el mismo procedimiento pero con 92 se tiene \.

Para lograr que el usuario pueda escribir en la tabla el código más sencillo es:

```
Private Sub GridMatriza_KeyPress(KeyAscii As Integer)
   GridMatriza.Text = GridMatriza.Text + Chr(KeyAscii)
End Sub
```

Sin embargo se puede mejorar haciendo de la siguiente forma:

```
Select Case KeyAscii
Case 8
   If GridMatriza.Text = "" Then Exit Sub
   GridMatriza.Text = ""
Case 13
   Exit Sub
Case Else
   GridMatriza.Text = GridMatriza.Text + Chr(KeyAscii)
End Select
```

En el caso de que se presione la tecla suprimir a la variable ***"KeyAscii"*** se le asocia el valor 8 que corresponde al código ASCII de suprimir, esto permite borrar en la celda, de igual forma si se presiona la barra espaciadora se asigna el valor

de 13 a la variable, en caso de presionar cualquier otra tecla se imprimirá en la celda activa.

Para escribir mediante código se hace de la siguiente manera:

```
GridMatriza.TextMatrix(1,1)="Nombre"
GridMatriza.TextMatrix(1,1)=denr
```

Esto imprime en la celda (0,0) el texto Nombre, y en la celda el valor de la variable denr.

AII.V.I.II.II Lectura de datos de un MSFlexGrid.

En caso de querer leer el o los valores de una variable directamente de una tabla se escribe:

```
denr=GridMatriza.TextMatrix(1,1)
```

Entonces lee el valor de denr de la celda (0,1).

AII.V.I.II.III Renglones y columnas de un MSFlexGrid.

Antes de continuar es conveniente indicar que el MSFlexGrid tiene un renglón cero y una columna cero, que corresponde, en caso de contar con ellos, a un renglón y una columna fija, si no se tienen estos elementos fijos es necesario hacer algunas consideraciones al escribir el código.

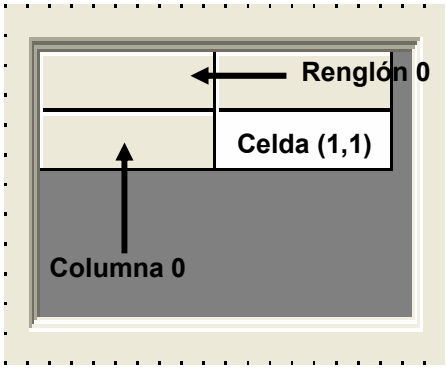


Figura AII.9 Renglón y Columna Fija.

Por ejemplo, si se utiliza una columna fija y un renglón fijo los datos se comienzan a introducir en la celda (1,1), si no se tienen las celdas fijas se introducen en la celda (0,0), esto puede provocar un poco de confusión en el momento de escribir el código correspondiente, pues generalmente se asocia este objeto con el uso de arreglos vectoriales y matriciales, sin embargo con práctica se llega a dominar.

Los renglones y columnas fijas se utilizan para colocar rótulos en las tablas de resultados, o para indicar al usuario que datos introducir en cada renglón o columna.

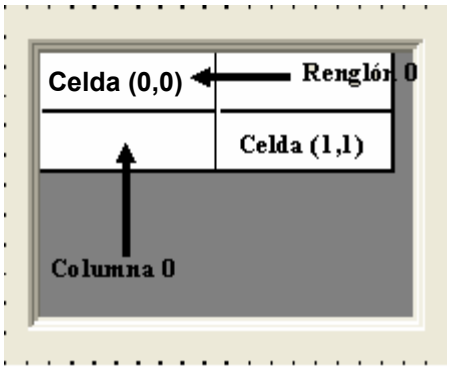


Figura AII.10 Renglón y Columna Libres.

AII.V.I.III Uso de arreglos con MSFlexGrid.

Evidentemente el MSFlexGrid se puede utilizar con los arreglos vectoriales o matriciales facilitando la lectura y asignación de valores a las variables, así como la presentación de resultados en una tabla.

La asignación de valores de una tabla a un arreglo se puede realizar de seis maneras distintas, tres para vectores y tres para matrices.

1. Considerando una tabla con una columna y un renglón fijos la forma de asignar valores a un arreglo vectorial es:

```
For i=1 to n
        a(i)=GridTabla.TextMatrix(i,1)
Next i
```

En este caso la variable *a* tomará los valores que se encuentren en los renglones *1* hasta *n* y que están dentro de la columna *1*.

2. Considerando una tabla con una columna fija y sin renglón fijo la forma de asignar valores a un arreglo vectorial es:

```
For i=0 to n-1
        a(i)=GridTabla.TextMatrix(i,1)
Next i
```

En este caso la variable *a* tomará los valores que se encuentren en los renglones 0 hasta *n-1* y que están dentro de la columna *1*.

3. Considerando una tabla sin columna y renglones fijos la forma de asignar valores a un arreglo vectorial es:

```
For i=0 to n-1
        a(i)=GridTabla.TextMatrix(i,0)
Next i
```

En este caso la variable *a* tomará los valores que se encuentren en los renglones 0 hasta *n-1* y que están dentro de la columna 0.

4. Considerando una tabla con una columna y un renglón fijos la forma de asignar valores a un arreglo matricial es:

```
For i =1 To n
        For j = 1 To m
                a(i, j) = GridTabla.TextMatrix(i, j)
        Next j
Next i
```

En este caso la variable *a* tomará los valores que se encuentren en los renglones *1* hasta *n* y la columna *1* hasta *m*. El orden de lectura es, primero todas las columnas del primer renglón, después todas las columnas del segundo renglón y así sucesivamente.

5. Considerando una tabla con una columna fija y sin renglón fijo la forma de asignar valores a un arreglo matricial es:

```
For i =0 To n-1
        For j = 1 To m
                a(i, j) = GridTabla.TextMatrix(i, j)
        Next j
Next i
```

En este caso la variable *a* tomará los valores que se encuentren en los renglones *0* hasta *n-1* y la columna *1* hasta *m*. El orden de lectura es, primero todas las columnas del primer renglón, después todas las columnas del segundo renglón y así sucesivamente.

6. Considerando una tabla sin columna y sin renglón fijo la forma de asignar valores a un arreglo matricial es:

```
For i =0 To n-1
        For j = 0 To m-1
                a(i, j) = GridTabla.TextMatrix(i, j)
        Next j
Next i
```

En este caso la variable *a* tomará los valores que se encuentren en los renglones *0* hasta *n-1* y la columna *0* hasta *m-1*. El orden de lectura es, primero todas las columnas del primer renglón, después todas las columnas del segundo renglón y así sucesivamente.

AII.V.I.IV Ejemplo.

Para realizar una suma de matrices se puede hacer uso de tres MSFlexGrid, uno para que el usuario introduzca los elementos de la matriz *A*, otro para la matriz *B* y uno para la matriz de resultado *C*, para hacerlo de manera dinámica el usuario debe tener la posibilidad de introducir el orden de las matrices, por lo que se le pueden solicitar mediante dos cajas de texto, una para el número de renglones y otra para las columnas, además de que los MSFlexGrid no deberán presentar renglones y columnas al inicio, estos se generaran en el momento que el usuario comience a ingresar el orden de las matrices.

Adicionalmente se utilizará un botón para que realice la lectura de los elementos de cada una de las matrices *A* y *B* realice la operación e imprima el resultado en un MSFlexGrid que representa la matriz *C*, en la siguiente tabla se presenta un resumen de los objetos:

AII.V.I.IV.I Resumen de objetos.

Propiedades	*Objetos*											
	Formulario	Etiqueta	Etiqueta	Etiqueta	Etiqueta	Etiqueta	Caja Texto	Caja Texto	Grid	Grid	Grid	Botón
Name	FrmSummatrices	Label1	Label2	Label3	Label4	Label5	TxtR	TxtC	GridA	GridB	GridC	CmdCalc
Caption	Suma de Matrices	# de Renglones	# de Columnas	Matriz A	Matriz B	Matriz C	N/A	N/A	Vacío	N/A	N/A	Calcular
Text	N/A	N/A	N/A	N/A	N/A	N/A	Vacío	Vacío	Vacío	Vacío	Vacío	N/A
Rows	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	0	0	N/A
Cols	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0	0	0	N/A

Tabla AII.7 Objetos para Ejemplo MSFlexGrid.

AII.V.I.IV.II Código de la aplicación.

A continuación se detalla una parte del código que hace funcionar esta aplicación.

```
Private Sub TxtC_Change()
GridA.Cols = Val(TxtC.Text)
GridB.Cols = Val(TxtC.Text)
End Sub
Private Sub TxtR_Change()
GridA.Rows = Val(TxtR.Text)
GridB.Rows = Val(TxtR.Text)
End Sub
```

Esta parte del código utiliza el evento ***change***, es decir, cuando cambia el contenido de la caja de texto ejecuta el código correspondiente, en este caso asigna el valor que el usuario introduce en la caja de texto correspondiente a los renglones y columnas de las matrices *A* y *B*. Una vez que se han generado los renglones de las matrices el usuario deberá introducir los elementos de cada una de estas, para poder escribir en las matrices es necesario adecuar el código presentado en la sección AII.V.I.III.I, entonces el usuario deberá hacer click en el botón calcular. A continuación se presenta una parte del código correspondiente a dicho botón:

```
n = Val(TxtR.Text)
m = Val(TxtC.Text)
ReDim a(n, m), b(n, m)
'Lectura de elementos de A
For i = 0 To n - 1
   For j = 0 To m - 1
      a(i, j) = GridA.TextMatrix(i, j)
   Next j
Next i
```

Asigna a las variables *n* y *m* que representan renglones y columnas respectivamente el valor de las cajas de texto correspondientes, después es necesario leer los elementos de cada matriz.

```
'Suma de A mas B

For i = 0 To n - 1
   For j = 0 To m - 1
      c(i, j) = a(i, j) + b(i, j)
   Next j
Next i
```

Como la suma de matrices se realiza elemento a elemento basta con sumar las variables *a* y *b*.

```
'Impresión en matriz C

For i = 0 To n - 1
   For j = 0 To m - 1
      GridC.TextMatrix(i, j) = c(i, j)
   Next j
Next i
```

Esta última parte imprime los valores que resultan de la suma de *a* y *b* que se encuentran contenidos en la variable *c*.

AII.V.II PictureBox.

El PictureBox es el objeto comúnmente usado para realizar gráficas en VISUAL BASIC 6.0®, adicionalmente se pueden utilizar para mostrar imágenes y mediante sus métodos crear gráficos como líneas, cuadrados, rectángulos, círculos o elipses; aunque con un poco de creatividad puede combinar varios de estos para realizar dibujos, se inserta haciendo doble click en el icono correspondiente  , en este trabajo se utilizará el PictureBox como un sistema cartesiano de coordenadas para graficar los resultados obtenidos durante la ejecución de una aplicación.

AII.V.II.I Propiedades del PictureBox.

Para poder interactuar con este objeto existen propiedades que se pueden modificar en tiempo de diseño o de ejecución, las dos más importantes son:

1. Scale: define la escala que tendrá el PictureBox así como la colocación del origen del sistema de coordenadas.
2. DrawWidth: define el ancho de línea.

AII.V.II.II Métodos gráficos.

Como se mencionó anteriormente, en el control PictureBox puede dibujar formas como puntos, líneas, círculos, elipses, arcos, cuadrados o rectángulos, además

permite escribir sobre él. Estos métodos gráficos también son aplicables a los formularios.

- Puntos: Se realizan mediante el método Pset:

  PictureBox.Pset (x,y),vbred

  Donde PictureBox es el nombre del Cuadro de Imagen, de un formulario, o si se omite se hará referencia al formulario actual (activo). x, y son las coordenadas donde se colocará el punto, estas pueden estar dadas por un número fijo o por alguna o algunas variables, estas pueden ser incluso arreglos, *vbred* se refiere al color que tendrá el punto o línea trazado, en este caso es rojo, si se omite entonces el color será negro, algunos colores mas son vbyellow, vbblue, vbmagenta, etc.

- Líneas: Se utiliza el método Line:

  PictureBox.Line (x_1, y_1)*-*(x_2, y_2)*,vbred*

  Donde x_1 y y_1 son las coordenadas del punto inicial y las del punto final x_2 y y_2, el color es opcional.

- Círculos: Se realizan mediante el método Circle:

  PictureBox.Circle (x,y), radio,vbred

  x, y son las coordenadas del centro del círculo, radio el tamaño del radio y el color es opcional.

- Elipses: Se realizan mediante el método Circle:

  PictureBox.Circle (X,Y), radio, vbred, [inicio], [fin], aspecto

  Donde radio es la distancia más grande de la elipse, inicio y fin se colocan si se desea realizar sólo un arco, y aspecto es la relación o proporción de la distancia vertical con respecto a la horizontal, los valores mayores a uno dibujan elipses alargadas verticalmente, mientras que los valores entre cero y uno dibujan elipses alargadas horizontalmente. No se aceptan valores negativos.

- Arcos: También se utiliza, por supuesto, el método gráfico Circle:

  PictureBox.Circle (x.y), radio, vbred, inicio, fin, [aspecto]

Donde inicio y fin son los ángulos de inicio y fin del arco en radianes y aspecto es opcional y se coloca si desea dibujar el arco de una elipse.

- Cuadrados o Rectángulos: Se realizan mediante el método Line:

```
PictureBox.Line (x,y)-Step(Ancho,Alto), vbred, B
```

Donde las coordenadas x, y son las coordenadas de la esquina superior izquierda. Si desea rellenar el cuadrado, debe cambiar la sentencia *B* por *BF*.

- Escribir Texto: Se escribe texto mediante el método Print. Para establecer el lugar donde se colocará el texto se utilizan las instrucciones Objeto.CurrentX y Objeto.CurrentY.

```
PictureBox.CurrentX = CoordenadaX
PictureBox.CurrentY = CoordenadaY
PictureBox.Print "Escriba texto"
```

AII.V.II.III Gráficas en PictureBox.

Inicialmente el PictureBox tiene el origen en el extremo superior izquierdo, por lo que el sistema cartesiano estaría definido de la siguiente forma:

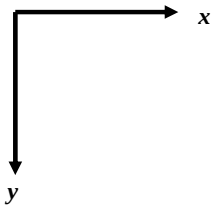


Figura AII.11 Sistema Cartesiano PictureBox.

Por lo general cuando se desea graficar algunos resultados se hace uso de el primer cuadrante de un sistema cartesiano o de los cuatro cuadrantes, para esto se hace uso de la propiedad Scale que ubica el origen en el punto que se desee además de dar dimensiones al PictureBox.

La forma en que se colocaría el origen para el primer cuadrante es la siguiente:

```
PictureBox.Scale (0, 50)-(50, 0)
```

Esto significa que el PictureBox tendrá el origen en el extremo inferior izquierdo y que cincuenta unidades al extremo inferior derecho y cincuenta unidades al extremo superior izquierdo.

Para graficar se puede utilizar alguno de los métodos mostrados en la sección anterior, sin embargo el más recomendable es el de puntos.

AII.V.II.IV Ejemplo.

El siguiente código permite graficar una línea recta con pendiente positiva, ordenada al origen igual a cinco en el primer cuadrante.

```
Private Sub Command1_Click()
Dim x() As Double, y() As Double
PicGrafica.Scale (0, 100)-(100, 0)
PicGrafica.Line (0, -100)-(0, 100)
PicGrafica.Line (-100, 0)-(100, 0)
ReDim x(50), y(50)
For i = 1 To 50
   x(i) = i
   y(i) = x(i) + 5
   PicGrafica.PSet (x(i), y(i))
Next i
End Sub
```

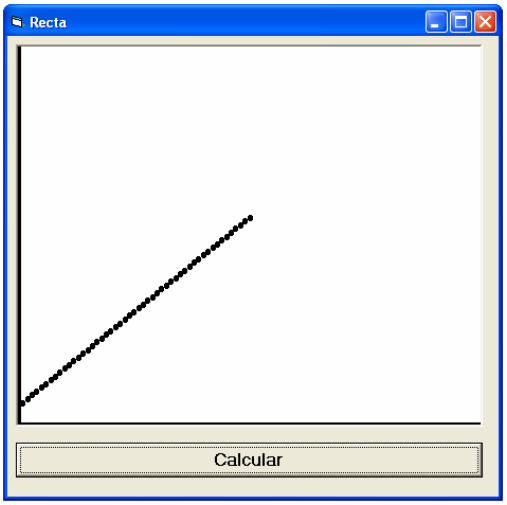


Figura AII.12 Ejemplo en Ejecución.

AII.V.III OptionButton.

Estos objetos se encuentran regularmente en las aplicaciones de WINDOWS®, OFFICE®, etc., y aparecen en grupos por lo menos de dos elementos, se utilizan para delimitar las opciones con las que cuenta el usuario permitiendo solo elegir una de las presentadas.

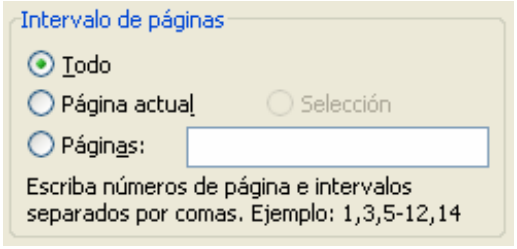


Figura AII.13 OptionButton.

Se pueden insertar haciendo doble click en el icono correspondiente  .

AII.V.III.I OptionButton con IF…THEN…ELSE.

Una manera de utilizar los OptionButton es indicando en el código que si se encuentra seleccionado ejecute ciertas instrucciones o no. Las siguientes líneas ejemplifican esto:

```
If OptRecta = True Then
…
…
ElseIf OptcCircunferencia=True Then
…
…
ElseIf OptcCircunferenci=True Then
…
…
End If
```

Según el OptionButton seleccionado ejecutara las instrucciones correspondientes.

AII.V.III.I OptionButton con el Evento Click.
Utilizando el evento Click o DblClick se puede indicar que instrucciones realizar con el OptionButton, generalmente estas van dirigidas a deshabilitar o habilitar uno o más objetos en tiempo de ejecución, las siguientes lineas ejemplifican esto:

```
Private Sub OptDr_Click()
TxtRa.Visible = True
TxtCa.Visible = True
Label1.Visible = True
Label2.Visible = True
Label6.Visible = True
CmdDr.Enabled = True
CmdDr.Visible = True
OptSuma.Enabled = False
OptMult.Enabled = False
OptTrans.Enabled = False
OptEscal.Enabled = False
OptDr.Enabled = False
End Sub
```

La propiedad Visible permite que un objeto se vea o no en el formulario, mientras que la propiedad Enabled deshabilita los objetos, es decir, si se trata de un botón el usuario no obtendra respuesta de este al hacer click sobre de él.

AII.V.IV CheckBox.

Estos objetos, al igual que los OptionButton, se encuentran regularmente en las aplicaciones de WINDOWS®, OFFICE®, etc., se utilizan para ofrecer una o más opciones al usuario permitiendo elegir alguna opciones presentadas.

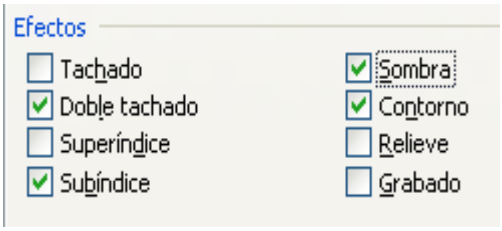


Figura AII.14 CheckBox.

AII.V.VI.I CheckBox con IF…THEN…ELSE.
Las siguientes líneas muestran el uso de este objeto:

```
If ChkBin.Value = Checked Or ChkLey.Value = Checked Then
…
…
End if
```

Significa que si una o la otra casilla esta verificada ejecute el código correspondiente.

AII.VI Recomendación.

En el caso de VISUAL BASIC 6.0® programar se vuelve un poco mas complejo, pues como se ha visto a lo largo de esta sección es necesario indicar a los objetos que acciones realizar y como hacerlo, a diferencia de la programación estructurada que solo se enfoca a la solución directa del problema que se pretende resolver.

Este pequeño manual no es suficiente para desarrollar las habilidades necesarias para realizar programación orientada a objetos, solo pretende ser una guía para quien comienza a en este campo, por lo que se recomienda después de estudiarlo elaborar un programa capaz de realizar la multiplicación de matrices definida por el algebra, el método de Newton – Raphson y una aplicación capas de realizar la gráfica de un función, pues aunque para un principiante son complejos, si se logran realizar permiten entender la lógica de la programación orientada a objetos.

AIII. MATLAB 7.0®.

AIII.I Introducción.

El nombre MATLAB 7.0® es la abreviatura de "MATrix LABoratory". Se trata de un programa para realizar cálculos numéricos con vectores y matrices, también puede trabajar con números escalares tanto reales como complejos, con cadenas de caracteres y con otras estructuras de información más complejas. La función más atractiva es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB 7.0® tiene también un lenguaje de programación propio.

AIII.II Espacio de trabajo.

El espacio de trabajo con el que se cuenta en el momento de abrir MATLAB 7.0® es el que se muestra en la figura:

Esta configuración puede ser cambiada fácilmente por el usuario, es posible que en muchos casos lo que aparezca sea muy diferente. En este caso, una vista similar se puede conseguir con Desktop Layout/Default, en el menú View.

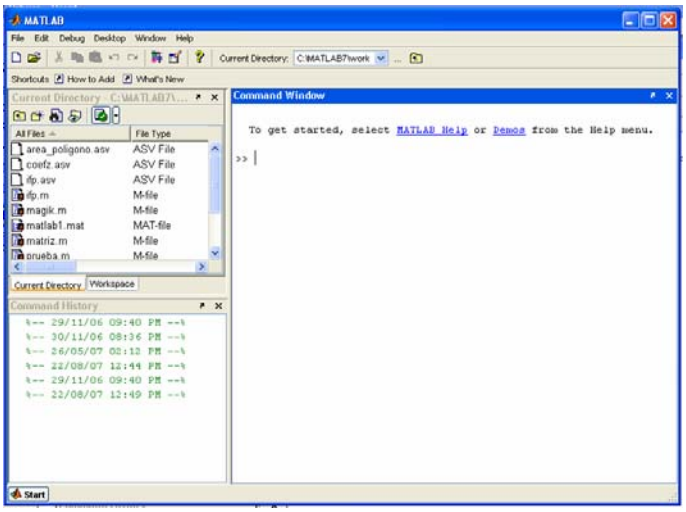


Figura AIII.1 Espacio de Trabajo.

La parte más importante de la ventana inicial es la ventana Command Window, que aparece en la parte derecha. En esta sección se ejecutan los comandos de MATLAB 7.0® a continuación del prompt, que no es mas que un aviso característico representado por >>, esto indica que el programa está preparado para recibir instrucciones.

En la parte izquierda de la pantalla aparecen dos ventanas: en la parte superior aparece la ventana Current Directory, que se puede alternar con Workspace

haciendo click en la pestaña correspondiente. La ventana Current Directory muestra los archivos del directorio activo o actual. El directorio activo se puede cambiar desde Command Window, o desde la propia ventana con los métodos de navegación de directorios propios de Windows.

Haciendo doble click sobre alguno de los archivos *.m del directorio activo se abre el editor de archivos de MATLAB 7.0®, que es la herramienta fundamental para la programación. El Workspace contiene información sobre todas las variables que se hayan definido en la sesión y permite ver y modificar las matrices con las que se esté trabajando.

En la parte inferior aparece la ventana Command History que muestra los últimos comandos ejecutados en la Command Window. Estos comandos se pueden volver a ejecutar haciendo doble click sobre ellos. Haciendo click sobre un comando con el botón derecho del ratón se muestra un menú contextual con las posibilidades disponibles en ese momento. Para editar uno de estos comandos hay que copiarlo antes a la Command Window.

En la parte inferior izquierda de la pantalla aparece el botón Start, con una función similar a la del botón Inicio de Windows. Start da acceso inmediato a ciertas capacidades del programa. Entre ellas se encuentran algunos demos y aplicaciones de gran utilidad, por ejemplo se cuenta con herramientas como los spline, ecuaciones de tipo financiero, estadístico, etc.

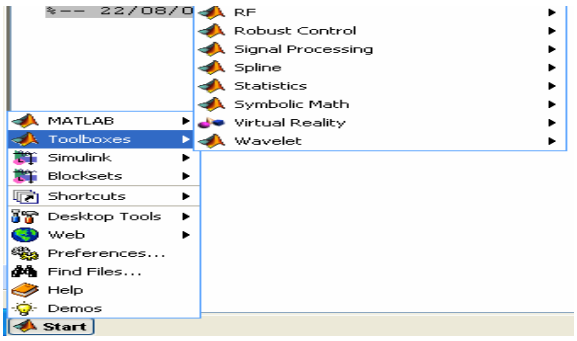


Figura AIII.2 Botón Start.

AIII.III Asignación de valores.

Existe diferentes formas de asignar valores a una variable, en esta sección se muestran algunas de las mas sencillas, antes de continuar cabe mencionar que MATLAB 7.0® reconoce la diferencia entre mayúsculas y minúsculas, por ejemplo las variables densidad, Densidad y DENSIDAD son tres variables diferentes, por lo que es importante tener cuidado cuando estas se escriben. La forma más sencilla de definir una matriz es usar una lista de números:

```
A=[3.5];
B=[1.5,3.1];
C=[-1,-0,0;1,1,0;1,-1,0;0,0,2];
```

Estas instrucciones son ejemplos de la asignación de valores, consiste de un nombre de variable seguido de un signo igual y de los datos que se asignarán a la variable, dichos valores se encierran en corchetes en orden por fila; las filas se separan con signos de punto y coma, y los valores de cada fila se separan mediante comas o espacios, el punto y coma hace que solo almacene los valores

en la variable, si se omite los almacena y además los imprime en pantalla. Un valor puede contener un signo menos y un punto decimal, pero no puede contener una coma. Entonces la variable A se trata de un vector con una dimensión, la variable B de un vector de dos elementos y la variable C una matriz de orden $(4*3)$, si se trata de un número escalar se introduce de la misma manera pero sin hacer uso de los corchetes. Otra forma de asignar valores es introduciendo las filas en forma de lista:

```
C=[-1,-0,0
     1,1,0
     1,-1,0
     0,0,2];
```

Para desplegar los valores almacenados en una variable, basta con escribir su nombre y presionar la tecla enter:

```
>> B
B =
   1.5000    3.1000
>> C
C =
   -1     0     0
    1     1     0
    1    -1     0
    0     0     2
```

Es posible modificar el contenido de los arreglos en uno o varios elementos, por ejemplo si se quiere cambiar el elemento (1,2) de la variable C anteriormente definida se procede de la siguiente manera:

```
>> C(1,2)=2.5
C =
  -1.0000    2.5000         0
   1.0000    1.0000         0
   1.0000   -1.0000         0
         0         0    2.0000
```

Si se teclea ***who*** aparecen las variables que se han utilizado hasta el momento:

```
>> who
Your variables are:
A  B  C
```

Si se teclea ***whos*** no solo se muestran los nombres de las variables, también aparecen las dimensiones, la clase, y el número de bytes que utilizan:

```
>> whos
Name      Size                Bytes  Class
   A         1x1                 8    double array
   B         1x2                16    double array
   C         4x3                96    double array
Grand total is 15 elements using 120 bytes
```

AIII.IV Valores y matrices especiales.

AIII.IV.I Valores especiales.

Algunos valores y constantes utilizados en MATLAB 7.0® son:

Valores Especiales	
pi	**Representa π**
i,j	**Representa el valor**
Inf	**Representa infinito**
NaN	**No es un número**
ans	**Un valor calculado pero no almacenado en un nombre de variable**

Tabla AIII.1 Valores Especiales.

AIII.IV.II Matrices especiales.

AIII.IV.II.I Elementos continuos.

Se pueden generar arreglos sin necesidad de teclear todos los valores, siempre y cuando estos cumplan una cierta continuidad, por ejemplo, si se quiere tener un arreglo que contenga los números del cero al diez se escribe como:

```
>> x=0:10
x =
    0    1    2    3    4    5    6    7    8    9   10
```

También se pueden introducir números no continuos pero con espaciamiento uniforme, por ejemplo generar un arreglo que contenga números de cinco a cincuenta y que vayan de cinco en cinco:

```
>> y=5:5:50
y =
    5   10   15   20   25   30   35   40   45   50
```

AIII.IV.II.II Matriz transpuesta.

Es muy útil no solo para matrices, también para vectores, pues permite presentar estos en forma de lista. Por ejemplo si se quiere representar la variable x anteriormente definida en forma de lista se escribe:

```
[x']
ans =
    1
    2
    3
    4
    5
    6
    7
    8
    9
   10
```

Si se desea escribir una tabla completa y que involucre mas de una variable se escribe:

```
[x' y']
ans =
    1     5
    2    10
    3    15
    4    20
    5    25
    6    30
    7    35
    8    40
    9    45
   10    50
```

Es importante recordar que ans no es una variable.

AIII.IV.II.III Matriz identidad.

Se puede tener una matriz identidad de cualquier tamaño sin necesidad de teclear cada uno de los elementos:

```
>> I=eye(10)
I =
    1    0    0    0    0    0    0    0    0    0
    0    1    0    0    0    0    0    0    0    0
    0    0    1    0    0    0    0    0    0    0
    0    0    0    1    0    0    0    0    0    0
    0    0    0    0    1    0    0    0    0    0
    0    0    0    0    0    1    0    0    0    0
    0    0    0    0    0    0    1    0    0    0
    0    0    0    0    0    0    0    1    0    0
    0    0    0    0    0    0    0    0    1    0
    0    0    0    0    0    0    0    0    0    1
```

AIII.IV.II.IV Matriz inversa.

Se puede obtener fácilmente la inversa de una matriz mediante:

```
>> inv(A)
ans =
    4.5000    1.5000    0.5000
   -1.5000   -0.5000   -0.5000
    0.5000    0.5000   -0.5000
```

AIII.IV.II.V Determinante.

El determinante se puede calcular de la siguiente manera:

```
>> det(A)
ans =
    2
```

AIII.V Operaciones con escalares y arreglos.

No existe una gran diferencia entre la forma en que MATLAB 7.0® trabaja y la forma en que lo hacen los lenguajes de programación comunes.

AIII.V.I Operaciones con escalares.

La forma de operar con escalares en MATLAB 7.0® no es diferente de las vistas anteriormente en FORTRAN® y VISUAL BASIC 6.0®, en la siguiente tabla se resumen las operaciones aritméticas:

Operador	*MATLAB*
Suma	**a+b**
Resta	**a-b**
Multiplicación	**a*b**
División	**a/b**
Exponenciación	**a^b**

Tabla AIII.2 Operaciones Escalares.

AIII.V.II Operaciones con arreglos.

Las operaciones entre arreglos en MATLAB 7.0® se realizan elemento a elemento igual que en los lenguajes anteriormente vistos, por ejemplo:

```
C(1)=A(1)*B(1);
C(2)=A(2)*B(2);
C(3)=A(3)*B(3);
C(4)=A(4)*B(4);
C(5)=A(5)*B(5);
```

Sin embargo la multiplicación de matrices la puede realizar elemento a elemento o según las reglas del algebra de matricial. En la siguiente tabla se resumen las operaciones entre arreglos:

Operador	*MATLAB*
Suma	**a+b**
Resta	**a-b**
Mult. Elemento a Elemento	**a.*b**
Multiplicación	**a*b**
Div. Elemento a Elemento	**a./b**
Exp. Elemento a Elemento	**a.^b**

Tabla AIII.3 Operaciones con Arreglos.

Entonces definiendo las siguientes matrices y realizando la multiplicación elemento a elemento se tiene:

```
>> a=[1,2,3;4,5,6;7,8,9]
>> b=[9,8,7;6,5,4;3,2,1]
>> c=a.*b
c =
     9    16    21
    24    25    24
    21    16     9
```

En cambio si se realiza según el algebra matricial, se tiene:

```
>> c=a*b
c =
    30    24    18
    84    69    54
   138   114    90
```

Es evidente la ventaja que esto tiene, sin embargo hay que ser cuidadosos con lo que se desea realizar pues pueden cometerse errores importantes.

AIII.V.III Operadores lógicos.

Los operadores con los que trabaja MATLAB 7.0® son los siguientes:

Operador	*Significado*
~	**Negación**
&	**Y**
|	**O**

Tabla AIII.4 Operadores Lógicos.

AIII.V.IV Operadores de comparación.

Los operadores con los que trabaja MATLAB 7.0® son los siguientes:

Operador	*Significado*
==	**Igual**
~=	**Diferente**
<	**Menor**
>	**Mayor**
<=	**Menor o igual**
>=	**Mayor o igual**

Tabla AIII.5 Operadores Comparación.

AIII.VI Gráficos con MATLAB 7.0®.

AIII.VI.I Gráficos 2D.

MATLAB 7.0® cuenta con cinco funciones básicas para crear gráficos en dos dimensiones. Estas funciones se orientan principalmente por el tipo de escala que utilizan en los ejes de abscisas y de ordenadas, en la siguiente tabla se resumen dichas funciones:

Funciones Gráficas Básicas 2D	
Plot()	**Crea un gráfico a partir de vectores y/o columnas de matrices, con escalas lineales sobre ambos ejes.**
plotyy()	**Dibuja dos funciones con dos escalas diferentes para las ordenadas, una a la derecha y otra a la izquierda de la figura.**
loglog()	**Con escala logarítmica en ambos ejes.**
semilogx()	**Con escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas.**
semilogy()	**Con escala lineal en el eje de abscisas y logarítmica en el eje de ordenadas.**

Tabla AIII.6 Funciones Gráficas 2D.

AIII.VI.I.I Función plot().

Basta con escribir esta función e indicar las variables a graficar para que se genere el gráfico deseado, este se presenta en una ventana nueva e independiente al entorno en donde se trabaja normalmente, dicha ventana cuenta con sus propias opciones y menús.

AIII.VI.I.II Función title.

Esta función permite agregar un titulo al gráfico.

AIII.VI.I.III Función xlabel e ylabel.

Para agregar los rótulos correspondientes a los ejes se hace uso de las funciones xlabel e ylabel.

AIII.VI.I.IV Función grid.

Esta función agrega una cuadrícula en la gráfica.

AIII.VI.I.V Ejemplo.

Definiendo los vectores x, y que representen la función $y = x^2$, y para realizar la gráfica correspondiente las líneas son:

```
>> plot(x,y)
>> title('Gráfica de un Parábola')
>> xlabel('Valores de x')
>> ylabel('Valores de y')
>>grid
```

En la ventana se pueden modificar todos los elementos de la gráfica según se necesite, de manera similar a como se haria en EXCEL®.

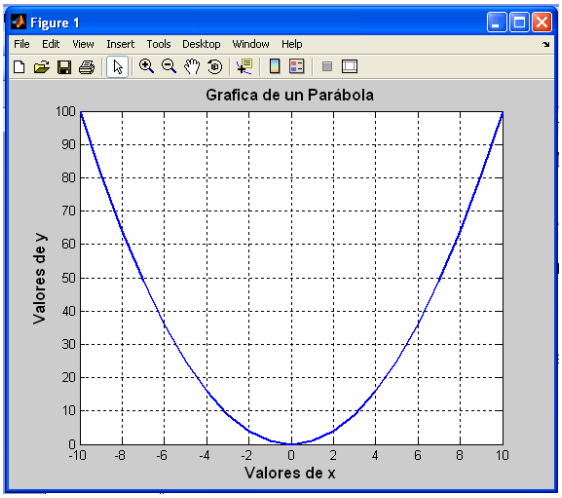


Figura AIII.3 Ejemplo Gráfica 2D..

AIII.VI.II Gráficos 3D.

MATLAB 7.0® también es capaz de realizar gráficos en tres dimensiones.

AIII.VI.II.I Función 3plot.

Esta función es parecida a la tratada en la sección anterior, esta dibuja una línea que une los puntos $(x(1),y(1),z(1)),(x(2),y(2),z(2)),\ldots$, y la proyecta sobre un plano para poderla presentar en la pantalla.

AIII.VI.II.I.I Ejemplo.

A continuación se va a realizar un ejemplo sencillo consistente en dibujar un cubo. Para ello se define una matriz A cuyas columnas son las coordenadas de los vértices, y cuyas filas son las coordenadas x, y, z de los mismos.

```
>> A=[0 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0
0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1
0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0];
>> plot3(A(1,:)',A(2,:)',A(3,:)')
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
```

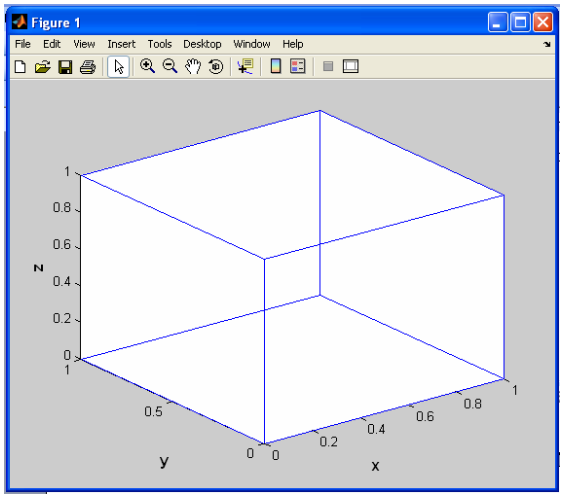


Figura AIII.4 Ejemplo Gráfica 3D.

AIII.VI.II.II Funciones meshgrid y mesh.

Ahora se verá con detalle cómo se puede dibujar una función de dos variables $z = f(x,y)$ sobre un dominio rectangular. Sean x e y dos vectores que contienen las coordenadas en una y otra dirección de la retícula sobre la que se va a dibujar la función, y sean las matrices X (cuyas filas son copias de x) e Y (cuyas columnas son copias de y). Estas matrices se crean con la función meshgrid y representan respectivamente las coordenadas x e y de todos los puntos de la retícula. La matriz de valores Z se calcula a partir de las matrices de coordenadas X e Y. Finalmente hay que dibujar esta matriz Z con la función mesh, cuyos elementos son función elemento a elemento de los elementos de X e Y.

AIII.VI.II.II.I Ejemplo.

Se realizará la gráfica de la función $sen(r)/r$ en donde $r = \sqrt{x+y}$; para evitar dividir por 0 se suma al denominador el número eps, que representa el machina epsilon mencionado en el capitulo I.

```
>>u=-8:0.5:8; v=u;
[U,V]=meshgrid(u,v);
R=sqrt(U.^2+V.^2)+eps;
W=sin(R)./R;
mesh(W)
```

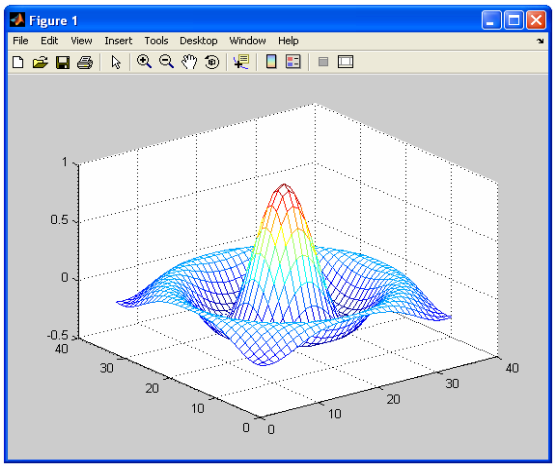


Figura AIII.5 Ejemplo Superficie 3D.

AIII.VII Programación con MATLAB 7.0®.

MATLAB 7.0® permite al usuario elaborar programas como lo haría en un lenguaje de programación, sin embargo no tiene las capacidades brindadas por estos.

*AIII.VII.I Archivos *.m.*

Los archivos con extensión (.m) son archivos de texto sin formato (archivos ASCII) que constituyen el centro de la programación en MATLAB 7.0®. Estos archivos se crean y modifican con un editor de textos cualquiera. En el caso de MATLAB 7.0®, lo mejor es utilizar su propio editor de textos, y que a su vez es también un compilador. Para abrir dicho editor basta con hacer click en el icono  del ambiente de trabajo y aparecerá una ventana independiente que corresponde a dicho editor. Es en esta ventana en donde se puede comenzar a escribir el código correspondiente. Una vez que se ha escrito el código de la aplicación se puede ejecutar desde el menú debug del mismo editor.

AIII.VII.II Lectura y escritura.

En esta sección se muestra una forma sencilla de leer variables desde teclado y escribir mensajes en la pantalla. Aunque existen otros modos más generales y complejos de hacerlo.

AIII.VII.II.I Función input.

Esta función permite imprimir un mensaje en la línea de comandos y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario. Después de imprimir el mensaje, el programa espera que el usuario teclee el valor numérico o la expresión. Cualquier expresión válida de MATLAB 7.0® es aceptada por este comando. El usuario puede teclear simplemente un vector o una matriz. En cualquier caso, la expresión introducida es evaluada con los valores actuales de las variables de MATLAB 7.0® y el resultado se devuelve como valor de retorno. Por ejemplo:

```
>> n = input('Teclee el número de ecuaciones')
```

AIII.VII.II.II Función disp.

Esta función permite imprimir en pantalla un mensaje de texto o el valor de una matriz, pero sin imprimir su nombre. En realidad, siempre imprime vectores y/o matrices, las cadenas de caracteres son un caso particular de vectores. Por ejemplo:

```
>> disp('El programa ha terminado')
>> A=rand(4,4)
>> disp(A)
```

AIII.VII.III Estructuras de control.

Estas estructuras permiten controlar el orden con el que el programa realizará las operaciones indicadas.

AIII.VII.III.I Estructuras de decisión.
Por medio de condiciones limitadas mediante comparaciones de una variable con un valor patrón, o entre dos o más variables, se indica al programa que instrucciones debe o no realizar. Para esto existe la estructura **IF … THEN … ELSE** cuya estructura en el código escrito en MATLAB 7.0® es:

```
IF condición
        instrucciones
        instrucciones
ELSE
        instrucciones
        instrucciones
END
```

Si se cumple la condición indicada se realizará el primer bloque de instrucciones y se omitirá el segundo, si no se cumple se omite el primer bloque y se realizan las instrucciones del segundo bloque. Se puede observar que solo se verifica que se cumpla una condición, pero puede darse el caso en que sea necesario verificar mas condiciones para decidir que bloque de instrucciones realizar, cuando se presenta este caso existe la siguiente estructura de decisión:

```
IF condición 1
        instrucciones
ELSEIF condición 2
        instrucciones
ELSEIF condición 2
        instrucciones
END
```

AIII.VII.III.I Estructuras de repetición o ciclos.
Permiten ejecutar un grupo de instrucciones durante un número finito de veces, para esto existen dos tipos de sentencia, los ciclos iterativos y los ciclos condicionados. La estructura de los ciclos iterativos es la siguiente:

```
FOR i=valor inicial:incremento:valor final
        instrucciones
        instrucciones
end
```

Si se omite el incremento automáticamente lo toma como uno.

La estructura de los ciclos condicionados se escribe:

```
WHILE condición
        instrucciones
        instrucciones
end
```

AIII.VII.IV Lectura de datos por medio de archivos externos.

Para leer la información de un archivo y ocuparla para realizar cálculos se debe escribir lo siguiente:

```
load('c:\presiones.txt')
```

AIII.VII.V Ejemplo 1.

Las siguientes líneas muestran el código para obtener el factorial de un número.

```
n=input('Indica el número del que deseas el factorial ')
if n<=0
   disp('No existe el factorial de números negativos')
elseif n==0
   disp('El factorial es igual a 1')
else
   fact=1
   for i=1:n
      fact=fact*i
   end
   disp('El factorial es igual a')
   disp(fact)
end
```

Estas líneas imprimirían lo siguiente en la pantalla de MATLAB 7.0®:

```
Indica el número del que deseas el factorial 5
n =
     5
fact =
     1
fact =
     1
fact =
     2
⋮
fact =
   120
El factorial es igual a
   120
```

Se puede ver que escribe la asignación de datos a una variable y los resultados de cada una de las operaciones realizadas, para evitar esto solo hay que agregar ; a las líneas que llevan a cabo estas funciones y se omitirá la impresión de esta información:

```
n=input('Indica el número del que deseas el factorial ');
if n<=0
   disp('No existe el factorial de números negativos')
elseif n==0
   disp('El factorial es igual a 1')
else
   fact=1;
   for i=1:n
```

```
        fact=fact*i;
    end
    disp('El factorial es igual a')
    disp(fact)
end
```

AIII.VII.VI Ejemplo 2.

El siguiente ejemplo realiza el cálculo del IPR con la gráfica correspondiente:

```
clc
disp('       Programa realizado por Juan Carlos Sabido')
disp('Tesis: Programacion Avanzada para Ingenieros Petroleros')
disp('=======================================================')
disp('')
disp('1.-Cualquier calculo')
disp('2.-Ejemplo')
disp('')
disp('')
opcion=input('Elije la opción que deseas')

if opcion==1
        clear
        clc
          disp('       Programa realizado por Juan Carlos Sabido')
          disp('Tesis: Programación Avanzada para Ingenieros Petroleros')
          disp('=======================================================')
          disp('')
        pws=input('Introduce la presion estatica ');
        pwf1=input('Introduce Pwf ');
        qo1=input('Introduce qo ');
          %calculo del ipr para yac. bajosaturado
          pendiente=(pwf1-pws)/(qo1)
        ipr=-1/pendiente
        disp('El IPR (J) es de:'),ipr
        n=input('Cuantos gastos deseas obtener ');
        disp('Introduce las Pwf que quieres evaluar ')
        pwf=input('Hazlo en forma de vector [1 2 3 4 5 ... n] ');
        for i=1:n
          qo(i)=(pwf(i)-pws)*-ipr;
        end
        disp('Gastos:'),qo
          plot(qo,pwf,'r*--')
          title('IPR')
elseif opcion==2
        clear
        clc
        disp('       Programa realizado por Juan Carlos Sabido')
          disp('Tesis: Programación Avanzada para Ingenieros Petroleros')
          disp('=======================================================')
          disp('')
         pws=3620;
         pwf1=3500;
         qo1=7.91;
         disp('La Presión Estatica es de: '),pws
         disp('La Presión de Fondo Fluyendo conocida es: '),pwf1
```

```
            disp('El Gasto a esta Pwf es: '),qo1
            %calculo del ipr para yac. bajosaturado
            pendiente=(pwf1-pws)/(qo1)
          ipr=-1/pendiente
          disp('El IPR (J) es de: '),ipr
          %lectura de datos
            load('f:\presiones.txt')

            for i=1:12
                    qo(i)=(presiones(i)-pws)*-ipr;
            end
          disp('Gastos:'),qo
            plot(qo,presiones,'b*--')
            title('IPR')
else
            clear
          clc
          disp('       Programa realizado por Juan Carlos Sabido')
             disp('Tesis: Programación Avanzada para Ingenieros Petroleros')
             disp('=====================================================')
             disp('')
             disp('La opcion no existe')
end
```

La instrucción clc solo limpia la pantalla, mientras que clear borra el valor de las variables, todas las funciones vistas anteriormente se pueden incluir en el código, por ejemplo la gráfica del IPR se genera desde el tiempo de ejecución de este programa.

AIV. POLINOMIOS DE LEGENDRE.

AIV.I Algoritmo.

Estos polinomios se escriben como:

$$P_0(x) = 1$$
$$P_1(x) = x$$
$$P_2(x) = \frac{1}{2}\left[3x^2 - 1\right]$$
$$P_3(x) = \frac{1}{2}\left[5x^3 - 3x\right]$$
$$P_4(x) = \frac{1}{8}\left[35x^4 - 30x^2 + 3\right]$$
$$\vdots$$

Para cualquier polinomio de Legendre de grado superior se puede obtener mediante la siguiente formula:

$$nP_n(x) - (2n-1)xP_{n-1}(x) + (n-1)P_{n-2}(x) = 0 \quad \text{…AIV.1}$$

O de manera alternativa:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n\left(x^2-1\right)^n}{dx^n} \quad \text{…AIV.2}$$

Una propiedad importante de este tipo de polinomios es la ortogonalidad:

$$\int_{-1}^{1} P_m(x)P_n(x)dx = 0, \ n \neq m \quad \text{…AIV.3}$$

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \frac{2}{2n+1}, \ m = n \quad \text{…AIV.4}$$

Esto indica que la integral en $[-1,1]$ de dos polinomios de Legendre distintos es igual a cero.

AV. MALLAS CARTESIANAS Y RADIALES.

AV.I Introducción.

La simulación numérica de yacimientos es una de las herramientas más utilizadas en la industria petrolera. Surge como una necesidad de darle solución a la ecuación de difusión.

Se divide el yacimiento en forma discreta, para lo cual se hace uso de una malla, esto es la discretización del yacimiento. Una vez discretizado el yacimiento se le asignan propiedades promedio a cada una de las celdas y posteriormente se aplica la ecuación de difusión, la solución de esta, se realiza por medio de diferencias finitas, esto conduce a un sistema de ecuaciones lineales que se resuelve de manera simultanea para cada una de las celdas, esto permite predecir el comportamiento del yacimiento bajo diferentes esquemas de producción del mismo. La división del yacimiento depende de los datos que se tienen del yacimiento y del tipo de simulación a realizar, para esto se tiene una clasificación de simuladores de yacimientos basado en el número de dimensiones y la geometría de las celdas de estudio.

- Simulador de una dimensión: Este modelo se puede aplicar si se tiene un yacimiento en el que predomina el flujo en una dirección, esto ocurre por ejemplo, cuando se realiza una inyección de gas, inyección o entrada de natural de agua.

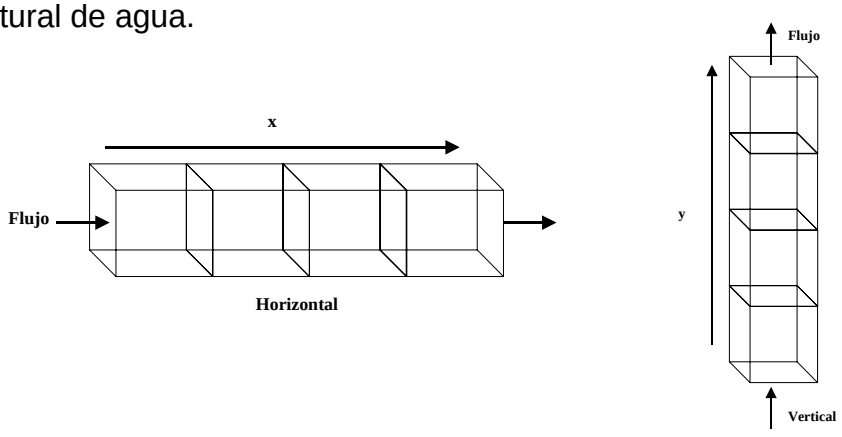


Figura AV.1 Simulador Unidimensional.

➢ Simulador de una dimensión radial: Este es útil para evaluar zonas de drene de un pozo y se conocen como pruebas de formación, incremento y decremento de presión.

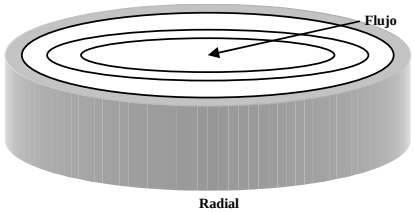


Figura AV.2 Simulador Unidimensional Radial.

➢ Simulador areal: Es un simulador en dos dimensiones y se utiliza cuando se tienen variaciones de las propiedades del yacimiento en dos direcciones *(x,y)*, además se pueden considerar los efectos gravitacionales al asignar diferentes profundidades a las celdas del modelo. Se aplica en yacimientos en donde generalmente los espesores son pequeños con respecto a su área y no existe efecto marcado de estratificación, son aquellos yacimientos donde se desean evaluar efectos de drene por zonas.

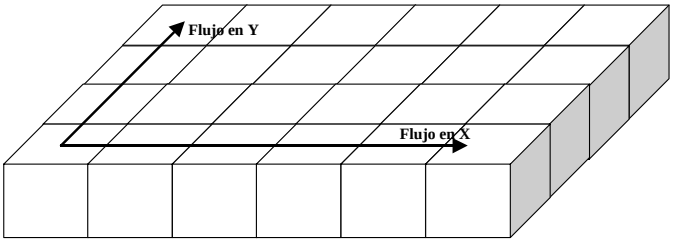


Figura AV.3 Simulador Areal.

➢ Simulador Transversal: Se trata también de un simulador en dos dimensiones, en donde las propiedades de las capas varían en *(x,z)*. Es útil por la versatilidad que tienen para describir la distribución vertical de saturaciones en el frente (gas y/o agua), además de que se pueden obtener las curvas de permeabilidad relativa.

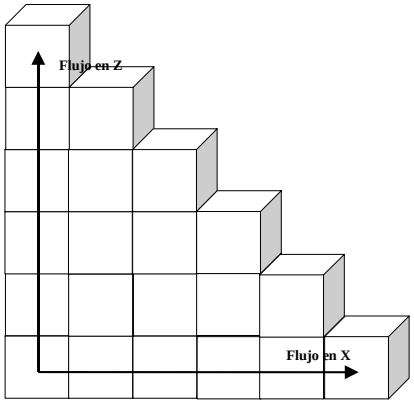


Figura AV.4 Simulador Transversal.

➢ Simulador de dos dimensiones en forma radial: Es útil para simular la conificación del agua, adicionalmente puede analizar con mayor detalle los cambios bruscos de saturación y presión que ocurren la cercanía del pozo.

➢ Simulador de tres dimensiones: Este es el mas complejo ya que incluye la mayor parte de las fuerzas que se presentan en el yacimiento, es decir además de considerar los efectos de barrido areal toma en cuenta los de barrido vertical. Se utiliza en yacimientos con geología compleja que puede tener como consecuencia el movimiento de fluidos a través del medio poroso en varias direcciones. Puede utilizarse con una malla cartesiana o una cilíndrica.

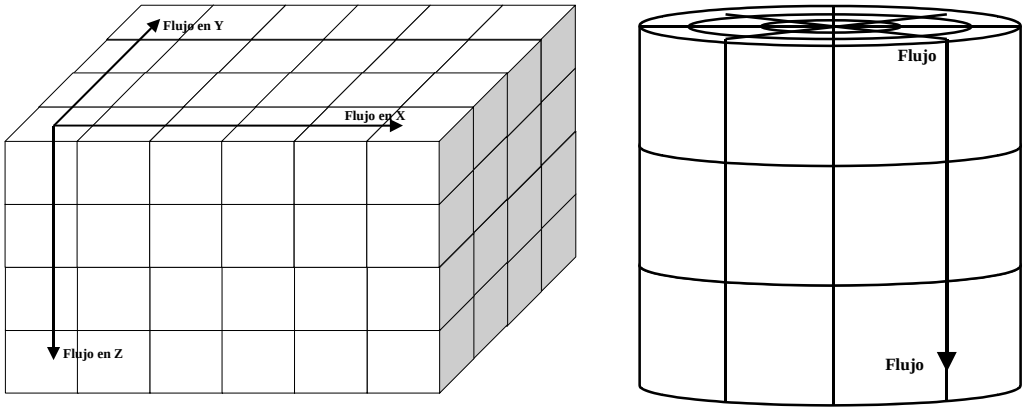


Figura AV.5 Simulador Tridimensional.

Aunque existen otro tipo de mallas, las anteriores son las más utilizadas actualmente en los simuladores comerciales, debido a la estabilidad numérica presentada durante su solución, las siguientes son mallas generadas por un simulador comercial:

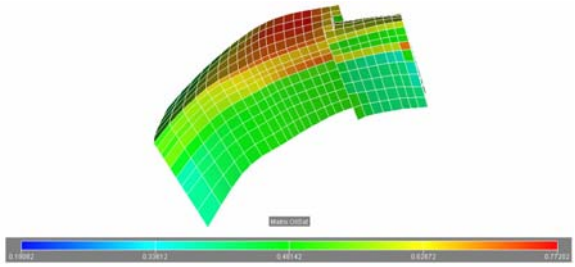


Figura AV.6 Malla Simulador Comercial.

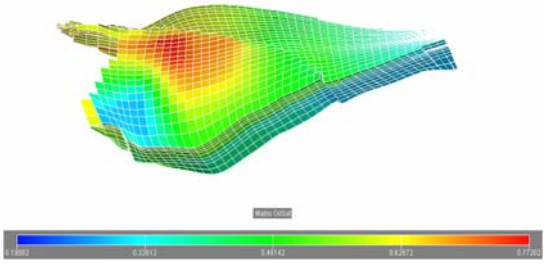


Figura AV.7 Malla Simulador Comercial.

AV.II Malla cartesiana.

Las mallas cartesianas toman su nombre de la forma en que se identifican los nodos, en tales casos la malla esta formada por pequeños cubos, estos se apilan para darle forma al yacimiento, a cada una de las celdas se le asignan propiedades petrofísicas promedio del área que representa, las propiedades pueden ser asignadas al centro de la celda o en las esquinas.

Este tipo de mallas es el que se utiliza con mayor frecuencia, los puntos de la malla son las intersecciones de los planos coordenados. Un punto o nodo tendrá dos puntos vecinos cuando se trate de una dimensión, cuatro en dos dimensiones y seis en tres dimensiones. Las mallas cartesianas se clasifican en nodos distribuidos y nodos centrados.

AV.II.I Nodos distribuidos.

Considerando una sola dimensión en x, los nodos distribuidos tienen la siguiente forma:

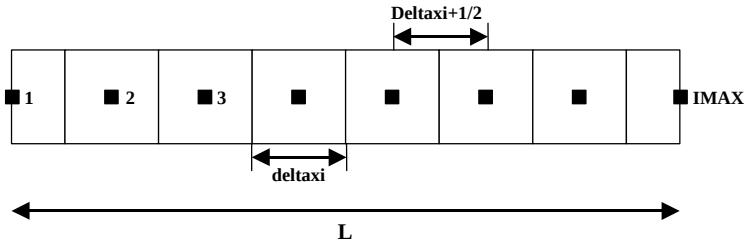


Figura AV.8 Nodos Distribuidos.

AV.II.I.I Algoritmo.[10]

Considerando que el número total de nodos que se requieren es igual $IMAX$, y el espaciamiento entre nodos es Δx se tiene:

$$\Delta x = \frac{L}{IMAX - 1} \quad \text{...AV.1}$$

Entonces la posición de los nodos es:

$$x_i = (i-1)\Delta x \, , i = 1, 2, \ldots, IMAX \quad \text{...AV.2}$$

Las fronteras se ubican mediante la ecuación:

$$x_{i+\frac{1}{2}} = \left(i - \frac{1}{2} \right) \Delta x \, , i = 1, 2, \ldots, IMAX \quad \text{...AV.3}$$

Para el volumen cada celda se debe considerar que las celdas en las fronteras (nodo 1 y nodo $IMAX$) es la mitad del volumen de las celdas internas:

$$Vr_i = \begin{cases} \frac{A\Delta x}{2}, i = 1, IMAX \\ A\Delta x, i = 2, 3, \ldots IMAX - 1 \end{cases} \quad \text{...AV.4}$$

AV.II.II Nodos centrados.

Para este tipo de malla los nodos 1 e $IMAX$ no se encuentran en las fronteras y el volumen de sus celdas es igual al que tienen las celdas internas.

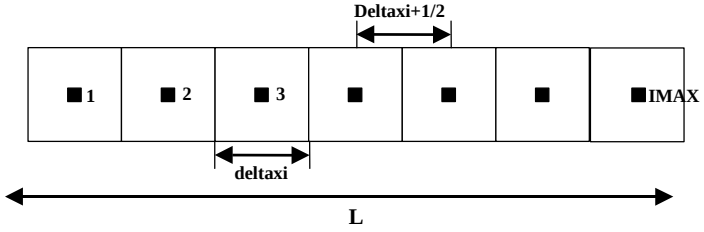


Figura AV.9 Nodos Centrados.

AV.II.II.I Algoritmo.[10]

Retomando que el número total de nodos que se requieren es igual $IMAX$, y el espaciamiento entre nodos es Δx se tiene:

$$\Delta x = \frac{L}{IMAX} \ldots \text{AV.5}$$

Entonces la posición de los nodos es:

$$x_i = \left(i - \frac{1}{2} \right) \Delta x \, , i = 1, 2, \ldots, IMAX \ \ldots \text{AV.6}$$

Las fronteras se ubican mediante la ecuación:

$$x_{i+\frac{1}{2}} = i \Delta x \, , i = 1, 2, \ldots, IMAX \ \ldots \text{AV.7}$$

El volumen de cada celda i esta dado por:

$$Vr_i = A \Delta x_i \, , i = 1, 2, \ldots, IMAX \ \ldots \text{AV.8}$$

AV.III Malla radial.

Este tipo de malla se utiliza para los estudios de flujo de fluidos en un solo pozo, esto incluye comportamiento individual del pozo, estrategias de terminación, conificación de agua y/o gas.

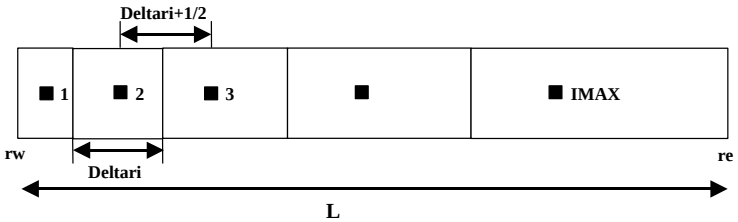


Figura AV.8 Malla Radial.

Como se observa en la figura, a diferencia de las mallas cartesianas en el que el tamaño de las celdas es uniforme a lo largo de la longitud L, en las mallas radiales es necesario seguir ciertas reglas.

AV.III.I Algoritmo.[10]

Método 1.

Los nodos son espaciados logarítmicamente a partir del pozo y hacia la frontera externa por medio de la expresión:

$$r_{i+1} = \psi r_i, i = 1, 2, \ldots, IMAX - 1 \ldots \text{AV.9}$$

En donde:

$$\psi = \left(\frac{r_e}{r_w} \right)^{\frac{1}{IMAX}} \ldots \text{AV.10}$$

$$r_1 = \frac{\psi \ln \psi}{(\psi - 1)} r_w$$

Las fronteras son definidas mediante una media logarítmica:

$$r_{i+\frac{1}{2}} = \frac{r_{i+1} - r_i}{\ln \psi} \ldots \text{AV.11}$$

Por ultimo el volumen de cada celda i es:

$$Vr_i = \pi \left(r^2_{i+\frac{1}{2}} - r^2_{i-\frac{1}{2}} \right) \Delta z \ldots \text{AV.12}$$

Método 2.

Los nodos están dados por:

$$r_1 = r_w e^{\left[\frac{\psi}{2} \right]} \ldots \text{AV.13}$$

$$r_i = r_1 e^{(i-1)\psi}$$

En donde:

$$\psi = \frac{1}{IMAX} \ln \left(\frac{r_e}{r_w} \right) \ldots \text{AV.14}$$

Las fronteras se calculan mediante.

$$r_{i+\frac{1}{2}} = r_w e^{i\psi} \ldots \text{AV.15}$$

El volumen de cada celda i es:

$$Vr_i = \pi \left(r^2_{i+\frac{1}{2}} - r^2_{i-\frac{1}{2}} \right) \Delta z \ldots \text{AV.16}$$

AVI. ECUACIÓN DE DIFUSION PARA FLUJO DE FLUIDOS EN MEDIOS POROSOS.

AVI.I Introducción.[10]

Para obtener la ecuación general que describe el comportamiento del flujo de fluidos a través de medios porosos, se hace uso de diferentes principios físicos:

- Conservación de masa (Ecuación de Continuidad).
- Conservación de Momento (Ley de Darcy).
- Ecuación de Estado.
- Conservación de Energía.

AVI.II Ecuación general.

Considerando la ecuación:

$$\frac{\partial}{\partial x}\left(\frac{k\rho}{\mu}\frac{\partial p}{\partial x}\right)\pm\tilde{q}_m=\frac{\partial(\phi\rho)}{\partial t}\quad\text{...AVI.1}$$

Esta ecuación rige el flujo de una sola fase en una dimensión en forma horizontal, asumiendo que se aplica la ecuación de Darcy. Entonces, desarrollando la ecuación AVI.1 derivando como producto de funciones se tiene:

$$\frac{k\rho}{\mu}\frac{\partial}{\partial x}\frac{\partial p}{\partial x}+\frac{k}{\mu}\frac{\partial p}{\partial x}\frac{\partial\rho}{\partial x}\pm\tilde{q}_m=\left[\phi\frac{\partial\rho}{\partial t}+\rho\frac{\partial\phi}{\partial t}\right]\quad\text{...AVI.2}$$

Agrupando y aplicando la regla de la cadena:

$$\frac{k\rho}{\mu}\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right)+\frac{k}{\mu}\frac{\partial p}{\partial x}\frac{\partial\rho}{\partial p}\frac{\partial p}{\partial x}\pm\tilde{q}_m=\left[\phi\frac{\partial\rho}{\partial p}\frac{\partial p}{\partial t}+\rho\frac{\partial\phi}{\partial p}\frac{\partial p}{\partial t}\right]$$

$$\frac{k\rho}{\mu}\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right)+\frac{k}{\mu}\frac{\partial \rho}{\partial p}\left(\frac{\partial p}{\partial x}\right)^2 \pm \tilde{q}_m = \left[\phi\rho\left(\frac{1}{\rho}\frac{\partial \rho}{\partial p}+\frac{1}{\phi}\frac{\partial \phi}{\partial p}\right)\frac{\partial p}{\partial t}\right] \ldots \text{AV.3}$$

El gradiente de presión al cuadrado es, por lo general, muy pequeño, entonces puede ser despreciado. Como la compresibilidad del fluido y de la roca se definen como:

$$c_r = \frac{1}{\phi}\left(\frac{\partial \phi}{\partial p}\right)_T$$
$$c_f = \frac{1}{\rho}\left(\frac{\partial \rho}{\partial p}\right)_T \quad \ldots \text{AVI.3}$$

Y sabiendo que:

$$c_t = c_r + c_f \ldots \text{AVI.4}$$

Se puede sustituir en AVI.2

$$\frac{k\rho}{\mu}\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right) \pm \tilde{q}_m = \left[\phi\rho c_t \frac{\partial p}{\partial t}\right] \ldots \text{AVI.5}$$

Dividiendo AVI.5 entre la densidad a condiciones estándar, se tiene:

$$\frac{k\rho}{\mu}\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right) \pm \frac{\tilde{q}_m}{\rho_s} = \left[\phi c_t \frac{\rho}{\rho_s}\frac{\partial p}{\partial t}\right] \ldots \text{AVI.6}$$

Considerando que el factor de volumen del fluido esta dado por:

$$B = \frac{\rho_s}{\rho} \ldots \text{AVI.7}$$

Sustituyendo en AVI.6:

$$\frac{k\rho}{\mu}\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right) \pm \frac{\tilde{q}_m}{\rho_s} = \left[\frac{\phi c_t}{B}\frac{\partial p}{\partial t}\right] \ldots \text{AVI.8}$$

Sin considerar fuentes ni sumideros y considerando k y μ constantes la ecuación AVI.8 queda como:

$$\frac{\partial^2 p}{\partial x^2} = \left[\frac{\phi\mu c_t}{k}\frac{\partial p}{\partial t}\right] \ldots \text{AVI.9}$$

Que es la ecuación de difusión para flujo monofásico en una sola dimensión.

AVII. ECUACIÓN DE DIFUSIÓN FLUJO RADIAL.

AVII.I Introducción.

La necesidad de conocer las diferentes características del yacimiento a través de métodos indirectos como es el caso de la medición de la presión, dio origen a las técnicas de análisis de variación de presión, las cuales son soluciones de las ecuaciones diferenciales parciales que describen el paso de fluidos a través de medios porosos. Para obtener este tipo de ecuaciones se hace uso de tres ecuaciones fundamentales:

- Conservación de masa (Ecuación de Continuidad).

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\rho v_r\right) = -\frac{\partial}{\partial t}\left(\phi\rho\right) \ldots \text{AVII.1}$$

- Conservación de Momento (Ley de Darcy).

$$v_r = -\frac{k_r}{\mu}\frac{\partial p}{\partial r} \ldots \text{AVII.2}$$

- Ecuación de Estado.

$$\rho = \rho_0\left[1 + c\left(p - p_0\right)\right] \ldots \text{AVII.3}$$

AVII.II Ecuación general.

Finalmente la ecuación de difusión se obtiene la al combinar la ecuación de estado AVII.1 la ecuación de movimiento AVII.2, y la ecuación de continuidad AVII.3:

$$\frac{1}{r}\frac{\partial}{\partial r}\left\{r\rho_0\left[1 + c\left(p - p_0\right)\right]\left(-\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right)\right\} = -\frac{\partial}{\partial t}\left\{\phi\rho_0\left[1 + c\left(p - p_0\right)\right]\right\} \ldots \text{AVII.4}$$

Cancelando ρ_0 que es una constante en ambos términos y desarrollando las parciales, con respecto al espacio (la radial) y al tiempo teniendo en cuenta que la presión es función del tiempo y del espacio, y la porosidad es función del tiempo se tiene:

$$\frac{1}{r}\left\{r\frac{\partial}{\partial r}\left[\left[1+c\left(p-p_0\right)\right]\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right]+\left[\left[1+c\left(p-p_0\right)\right]\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right]\frac{\partial r}{\partial r}\right\}=\phi\frac{\partial}{\partial t}\left[1+c\left(p-p_0\right)\right]+\left[1+c\left(p-p_0\right)\right]\frac{\partial\phi}{\partial t}$$

…AVII.5

Desarrollando las derivadas parciales:

$$\frac{1}{r}\left\{r\left[\left[1+c\left(p-p_0\right)\right]\frac{\partial}{\partial r}\left(\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right)+\frac{k_r}{\mu}\frac{\partial p}{\partial r}\frac{\partial}{\partial r}\left[1+c\left(p-p_0\right)\right]\right]+\left[\left[1+c\left(p-p_0\right)\right]\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right]\right\}=$$

$$\phi c\frac{\partial p}{\partial t}+\left[1+c\left(p-p_0\right)\right]\frac{\partial\phi}{\partial t}$$

…AVII.6

Calculando $\frac{\partial}{\partial r}\left[1+c\left(p-p_0\right)\right]=c\frac{\partial p}{\partial r}$:

$$\frac{1}{r}\left\{r\left[\left[1+c\left(p-p_0\right)\right]\frac{k_r}{\mu}\frac{\partial^2 p}{\partial r^2}+\frac{k_r}{\mu}\frac{\partial p}{\partial r}c\frac{\partial p}{\partial r}\right]+\left[\left[1+c\left(p-p_0\right)\right]\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right]\right\}=\phi c\frac{\partial p}{\partial t}+\left[1+c\left(p-p_0\right)\right]\frac{\partial\phi}{\partial t}$$ …AVII.7

Factorizando y simplificando ambos términos por $\left[1+c\left(p-p_0\right)\right]$:

$$\frac{1}{r}\left\{r\left[\frac{k_r}{\mu}\frac{\partial^2 p}{\partial r^2}+\frac{k_r}{\mu}\frac{c}{1+c\left(p-p_0\right)}\left(\frac{\partial p}{\partial r}\right)^2\right]+\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right\}=\phi\frac{c}{1+c\left(p-p_0\right)}\frac{\partial p}{\partial t}+\frac{\partial\phi}{\partial t}$$ …AVII.8

Ahora, la porosidad es función de la presión, y esta del tiempo, por lo cual si se desea calcular $\frac{\partial\phi}{\partial t}$ se utiliza la regla de la cadena:

$$\frac{\partial\phi}{\partial t}=\frac{\partial\phi}{\partial p}\frac{\partial p}{\partial t}$$ …AVII.9

Obteniendo con ello:

$$\frac{1}{r}\left\{r\left[\frac{k_r}{\mu}\frac{\partial^2 p}{\partial r^2}+\frac{k_r}{\mu}\frac{c}{1+c\left(p-p_0\right)}\left(\frac{\partial p}{\partial r}\right)^2\right]+\frac{k_r}{\mu}\frac{\partial p}{\partial r}\right\}=\phi\frac{c}{1+c\left(p-p_0\right)}\frac{\partial p}{\partial t}+\frac{\partial\phi}{\partial p}\frac{\partial p}{\partial t}$$ …AVII.10

Considerando gradientes de presión pequeños $\frac{\partial p}{\partial r} \approx 0$, se tiene que $\left(\frac{\partial p}{\partial r}\right)^2 \approx 0$ y se puede despreciar este término.

Ahora la compresibilidad de la formación está dada por:

$$c_f = \frac{1}{\phi}\frac{\partial \phi}{\partial p} \ldots \text{AVII.11} \;\therefore\; \frac{\partial \phi}{\partial p} = c_f \phi \ldots \text{AVII.12}$$

Considerando:

$$c' = \frac{c}{1 + c\left(p - p_0\right)} \ldots \text{AVII.13}$$

Con esto se puede escribir la ecuación AVII.9 en la forma:

$$\frac{1}{r}\left\{ r\frac{k_r}{\mu}\frac{\partial^2 p}{\partial r^2} + \frac{k_r}{\mu}\frac{\partial p}{\partial r}\right\} = \phi c'\frac{\partial p}{\partial t} + cf\,\phi\frac{\partial p}{\partial t} \ldots \text{AVII.14}$$

Y agrupando

$$\frac{1}{r}\left\{ r\frac{k_r}{\mu}\frac{\partial^2 p}{\partial r^2} + \frac{k_r}{\mu}\frac{\partial p}{\partial r}\right\} = \phi\frac{\partial p}{\partial t}\left(c' + cf\right) \ldots \text{AVII.15}$$

Si se agrupa en una sola constante la compresibilidad, se tiene:

$$c_t = c' + c_f \ldots \text{AVII.16}$$

$$\frac{1}{r}\left\{ r\frac{k_r}{\mu}\frac{\partial^2 p}{\partial r^2} + \frac{k_r}{\mu}\frac{\partial p}{\partial r}\right\} = \phi c_t\frac{\partial p}{\partial t} \ldots \text{AVII.17}$$

Si $\frac{k_r}{\mu}$ es constante:

$$\frac{1}{r}\left\{ r\frac{\partial^2 p}{\partial r^2} + \frac{\partial p}{\partial r}\right\} = \frac{\phi\mu ct}{k}\frac{\partial p}{\partial t} \ldots \text{AVII.18}$$

Con esto se obtiene la ecuación:

$$\frac{\partial^2 p}{\partial r^2} + \frac{1}{r}\frac{\partial p}{\partial r} = \frac{\phi \mu ct}{k}\frac{\partial p}{\partial t} \quad \ldots \text{AVII.19}$$

La cual, es una EDP del tipo parabólico.

AVIII. MÉTODO DE THOMAS.

AVIII.I Introducción.

El método de Thomas es un algoritmo utilizado para dar solución a sistemas de ecuaciones lineales que tienen la forma tridiagonal:

$$\begin{bmatrix} b_1 & c_1 & & & & & & & \\ a_2 & b_2 & c_2 & & & & & & \\ & a_3 & b_3 & c_3 & & & & & \\ & & a_4 & b_4 & c_4 & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \vdots \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \quad \text{...AVIII.1}$$

Este tipo de sistemas se tiene cuando se aproxima la ecuación de difusividad por medio de diferencias finitas (Capítulo VII) para un solo fluido y en una dirección.

Este algoritmo es un caso especial de la factorización LU por medio del método de Crout.

La ventaja es que no es necesario almacenar toda la matriz, solo los elementos que se encuentran en las tres diagonales, es decir, a_i, b_i, c_i, en forma de vectores, esto permite el ahorro de recursos computacionales además de tiempo en el cálculo.

AVIII.II Algoritmo.[9]

Para $i=1$ se tiene:

$$w_1 = \frac{c_1}{b_1} \ldots \text{AVIII.2}$$

Y

$$g_1 = \frac{d_1}{b_1} \ldots \text{AVIII.3}$$

Para $i = 2,3,\ldots,n-1$:

$$w_i = \frac{c_i}{b_i - a_i w_{i-1}} \ldots \text{AVIII.4}$$

Para $i = 2,3,\ldots,n$:

$$g_i = \frac{d_i - a_i g_{i-1}}{b_i - a_i w_{i-1}} \ldots \text{AVIII.5}$$

Realizando sustitución hacia atrás:

Para $i = n$

$$x_n = g_n \ldots \text{AVIII.6}$$

Y para $i = n-1, n-2, n-3, \ldots, 2,1$

$$x_i = g_i - w_i x_{i+1} \ldots \text{AVIII.7}$$

AIX. INTEGRAL EXPONENCIAL.

AIX.I Algoritmo.

Si el argumento x es menor o igual a 1.0 ($x \leq 1.0$) entonces:

$$E_i[x] = a_0 + (a_1 * x) + (a_2 * x^2) + (a_3 * x^3) + (a_4 * x^4) + (a_5 * x^5) - \log(x) \ldots \text{AIX.1}$$

En donde:

$$\begin{aligned} a_0 &= -0.57721566 & a_3 &= 0.05519968 \\ a_1 &= 0.99999193 & a_4 &= -0.00976004 \\ a_2 &= -0.24991055 & a_5 &= -0.00107857 \end{aligned}$$

Si el argumento x es mayor a 1.0 y menor que 10.0 ($1.0 < x < 10.0$) entonces:

$$E_i[x] = \frac{x^4 + (a_1 * x^3) + (a_2 * x^2) + (a_3 * x) + a_4}{\left(x^4 + (b_1 * x^3) + (b_2 * x^2) + (b_3 * x) + b_4\right)(x * e^x)} \ldots \text{AIX.2}$$

En donde:

$$\begin{aligned} a_1 &= 8.57332874 & b_1 &= 9.573322345 \\ a_2 &= 18.05901697 & b_2 &= 25.632956148 \\ a_3 &= 8.634760892 & b_3 &= 21.0996530827 \\ a_4 &= 0.2677737343 & b_4 &= 3.95849669228 \end{aligned}$$

En cualquier otro caso diferente a los anteriores:

$$E_i[x] = \frac{x^2 + (a_1 * x) + a_2}{\left(x^2 + (b_1 * x) + b_2\right)(x * e^x)} \ldots \text{VIII.3}$$

En donde:

$$a_1 = \ 4.03640 \quad b_1 = \ 5.03637$$
$$a_2 = \ 1.15198 \quad a_2 = \ 4.19160$$

AX. APLICACIÓN.

AX.I Introducción.

Adicional a este trabajo escrito, se elaboró un programa que contiene algunas de las aplicaciones tratadas en el capítulo VIII, además de algunos de los métodos numéricos presentados.

Este programa se incluye dentro del CD que acompaña al trabajo escrito y se encuentra en la carpeta ***Aplicación,*** se puede identificar por el icono  y el nombre ***Kon-1***, dentro de la misma carpeta se encuentra otra con el nombre ***Ejemplos y Resultados,*** aquí se encuentran una serie de archivos que contienen los datos de los ejemplos mostrados en este trabajo, y que, son necesarios para mostrar el funcionamiento del programa.

AX.II Presentación.

Cuando se inicia la aplicación aparece una pantalla que muestra los datos que describen el programa y el motivo de su creación, dicha imagen permanecerá por diez segundos antes de mostrar el menú principal en donde se podrá elegir alguna de las opciones disponibles.

Figura AX.1 Presentación.

AX.III Archivo.

Esta es una opción que se encuentra dentro de cada una de las aplicaciones que componen el programa, en ella se encuentran ejemplos previamente cargados dentro de la aplicación, o la manera de abrir los archivos que contengan dichos ejemplos, de igual forma se pueden abrir archivos

Figura AX.2 Archivo.

generados por el usuario y que contengan la información que desea procesar para cada aplicación.

AX.III.I Abrir.

Esta opción permite abrir archivos que contengan la información necesaria para algunas de las aplicaciones. En algunas otras, esta opción no aparece. Es importante recalcar que se debe respetar el formato de los archivos de ejemplo que se incluyen en la carpeta correspondiente, pues de no hacerlo el programa al abrir archivos que no cumplan con el orden establecido no funcionara correctamente.

AX.III.II Reporte.

Esta opción permite obtener, en algunos casos, un sencillo reporte de los resultados obtenidos durante la ejecución del programa, estos se envían a un archivo que al igual que los de ejemplo se pueden ver fácilmente con ayuda del bloc de notas de WINDOWS®.

AX.III.III Ejemplo.

Algunas de las aplicaciones no cuentan con las opciones descritas en los puntos AX.III.I y AX.III.II, esto es debido a que la complejidad de la información ya sea de entrada o salida no es muy extensa, sin embargo cuentan con un ejemplo, este se carga simplemente con hacer click sobre la opción correspondiente.

AX.III.IV Salir.

Esta opción permite salir de la aplicación en el momento en que se desee.

AX.IV Acceso a las aplicaciones.

Las aplicaciones contenidas dentro del programa se muestran en el menú principal, con solo hacer click sobre el botón correspondiente se podrá acceder directamente a cada una de ellas, excepto en las contenidas en el botón de métodos numéricos, pues en este caso se abrirá un menú adicional que permitirá elegir alguna de las posibilidades. En cada una de las aplicaciones existe un botón que brinda la posibilidad de volver al menú principal en cualquier momento.

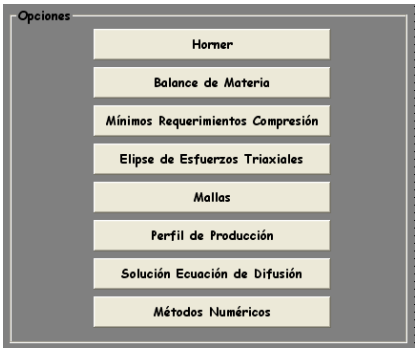


Figura AX.3 Menú Principal.

AX.V Recomendaciones.

La aplicación es muy sencilla de utilizar, pero es muy importante introducir la información correcta, de lo contrario, será difícil obtener buenos resultados, e incluso provocar que el programa falle por completo mandando mensajes de error. Esto a pesar de que se busco cuidar todos los detalles para el buen funcionamiento de la aplicación.

REFERENCIAS.

1.- Huerta, Antonio, Rodríguez Antonio, "Métodos Numéricos, Introducción, aplicaciones y programación", Ediciones UPC, Año 1998.

2.- Nakamura, Shoichiro, "Métodos Numéricos Aplicados con Software", Prentice – Hall Hispanoamérica, S.A. Año 1992.

3.- Maron, J. Maron, López J. Robert, "Análisis Numérico, Un enfoque Práctico", Compañía Editorial Continental, S.A. de C.V., Año 1995.

4.- Guzmán Guzmán, Roberto Ariel, "TESIS: Programación Avanzada Aplicada a Ingeniería Petrolera", Facultad de Ingeniería UNAM, Año 2001.

5.- Ceballos, Francisco Javier, "Curso de Programación de Visual Basic 6", Editorial Alfaomega, Año 2004.

6.- Perry, Greg, "Aprendiendo Visual Basic 5 en 24 Horas", Editorial Prentice Hall, Año 1998.

7.- Apuntes de la Clase de Computadoras y Programación de la Facultad de Ingeniería Impartida en el Semestre 2001-2.

8.- Apuntes de la Clase de Recursos Computacionales de la Facultad de Ingeniería Impartida en el Semestre 2004-2 por el M en I. Edgar Meza.

9.- Apuntes de la Clase de Programación Avanzada de la Facultad de Ingeniería Impartida en el Semestre 2006-1 por el Dr. Víctor Hugo Arana.

10.- Apuntes de la Clase de Simulación Numérica de Yacimientos de la Facultad de Ingeniería Impartida en el Semestre 2006-2 por el Dr. Víctor Hugo Arana.

11.- Apuntes de la Clase de Recursos Computacionales de la Facultad de Ingeniería Impartida por la M en I. Ruth Virginia Wilson.

12.- Gerald F. Curtis, "Análisis Numérico", Editorial Alfaomega, Año 1991.

13.- Rodríguez Nieto, Rafael, "Evaluación de la Producción", Septiembre 1982, Facultad de Ingeniería UNAM.

14.- Andrew H. Sherman, "ALGORITHM 533, NSPIV, A Fortran Subroutine for Sparse Gaussian Elimination UIT Partial Pivoting", "ACM Transactions on Mathematical Software Vol.4, No. 4", diciembre de 1978.

15.-Rodríguez de la Garza, Fernando, "Simulación Numérica Yacimientos", Apuntes encontrados en la dirección de Internet http://www.mmc.igeofcu.unam.mx/cursos/sny/SNY_Cap7.pdf.

16.- Apuntes de la Clase Caracterización de Formaciones de la Facultad de Ingeniería Impartida en el Semestre 2006-1 por el Ing. Manuel Villamar.

17.- Apuntes de la Clase Conducción y Manejo de la Producción de la Facultad de Ingeniería Impartida en el Semestre 2007-1 por el M. en I. José Angel Gómez Cabrera.

18.- McCain, William D., "The Properties of Petroleum Fluids", PennWell Publishing Company, Segunda Edición 1990 U.S.A.

19.-Etter, Delores M., "Solución de Problemas de Ingeniería con Matlab", Pearson Prentice Hall, Segunda Edición 2004, México D.F.

20.- Ramirez Briceño, Efraín, Perez Araiza, Omar, "TESIS: Ingeniería de Pozos", Facultad de Ingeniería, UNAM, Año 2001.

21.- Apuntes de la Clase Terminación y Reparación de Pozos de la Facultad de Ingeniería Impartida en el Semestre 2005-1 por el M. en I. Martín Terrazas.

http://www.uv.es/~diaz/mn/node13.html

http://www.monografias.com/trabajos/progestructu/progestructu.shtml

http://es.wikipedia.org/wiki/Fortran

http://es.wikipedia.org/wiki/No_linealidad

http://es.wikipedia.org/wiki/Interpolaci%C3%B3n