UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA

APUNTES DE

COMPUTACION APLICADA A LA INGENIERIA PETROLERA

JUAN M. BERLANGA

DIVISION DE INGENIERIA EN CIENCIAS DE LA TIERRA
DEPARTAMENTO DE EXPLOTACION DEL PETROLEO

FI/DICT/85-018

P R E F A C I O

En febrero de 1979 se firmó un convenio de colaboración entre la UNAM, PEMEX, IMP y el CIPM (Colegio de Ingenieros Petroleros de México). El objeto del convenio ha sido elevar el nivel académico de los alumnos del área de Ingeniería Petrolera en la Facultad de Ingeniería, tanto de licenciatura como de posgrado, así como crear el Doctorado, y promover la superación de un mayor número de profesionales que laboran en la industria petrolera, por medio de cursos de actualización y especialización.

Uno de los programas que se están llevando a cabo a nivel de licenciatura, dentro del marco del Convenio, es la elaboración y actualización de apuntes de las materias de la carrera de Ingeniero Petrolero. Con esto se pretende dotar al alumno de más y mejores medios para elevar su nivel académico, a la vez que proporcionar al profesor material didáctico que lo auxilie en el proceso enseñanza-aprendizaje. En cada caso particular de apuntes se presenta información sobre las personas que los han elaborado o han participado en alguna forma en su preparación.

DEPARTAMENTO DE EXPLOTACION DEL PETROLEO

I N T R O D U C C I O N

En los momentos actuales, se presenta cada vez con más frecuencia, entre los directivos de diversas industrias, la necesidad de tomar decisiones a corto o mediano plazo, las cuales pueden llegar a involucrar inversiones del orden de los cientos o miles de millones de pesos. Tales decisiones están soportadas, en la mayoría de las veces, por estudios o trabajos técnicos los cuales a su vez se encuentran restringidos por el factor tiempo. Bajo estas circunstancias resulta imprescindible poder contar con herramientas que permitan efectuar los estudios necesarios en tiempos restringidos. Hacia esta misma conclusión se llega cuando se considera la gran cantidad de información involucrada, tal como es el caso particular de los estudios que se realizan para la industria petrolera. Una herramienta que ha evolucionado y que continúa evolucionando al ritmo de las necesidades actuales y que ha probado ser eficaz en la solución de numerosos problemas, es la computadora.

Fué quizás tal concepción simplista lo que motivó a la División de Ingeniería en Ciencias de la Tierra de la Facultad de Ingeniería de la Universidad Autónoma de México a incorporar, dentro de los planes de estudio de la carrera de ingeniero petrolero, la materia de Computación Aplicada a la Ingeniería Petrolera.

Consecuentemente, y con el objeto de darle un carácter más formal a la materia y de proporcionarle un medio de estudio al alumno, se decidió elaborar este conjunto de apuntes. De ninguna manera se pretende que éste sea un trabajo concluído. Con el tiempo surgirán errores. Nuevos conceptos harán que se modifiquen o agreguen algunos capítulos.

El esfuerzo en la realización de la tarea fue compartido con mi primera generación de alumnos. Junto con ellos, los ejercicios fueron elaborados y corregidos. No puedo dejar de expresar que en esta empresa, tanto ellos como yo, fungimos como "conejillos de indias".

Juan M. Berlanga

Ciudad Universitaria

Diciembre, 1980.

CONTENIDO

CAPITULO I

INTRODUCCION

Todos aquellos ingenieros que por el carácter de su trabajo emplean la computadora deben saber, al menos básicamente, cuales son las restricciones de la computadora; su eficiencia en la ejecución de operaciones tales como la suma y la multiplicación; cuantos números pueden ser representados en una computadora; como evitar los errores de truncamiento en la elaboración de programas, etc.

El tema de este capítulo gira precisamente alrededor de estos conceptos. El propósito aquí es familiarizar al lector con el ambiente de la computadora.

Números en Punto Flotante.

Uno de los aspectos más importantes en el ambiente de las computadoras es el de la aproximación de los números reales o los números en punto flotante. Tal aproximación puede observarse en el cálculo simple de una fracción. Por ejemplo

$\frac{1}{3}$ igual a 0.333333...

En el mundo real y finito de las computadoras, este número puede ser representado únicamente con cierta precisión

$\frac{1}{3}$ aproximadamente igual a 0.333333

Diversos métodos han sido propuestos para la representación de números reales en computadoras. El más empleado de ellos es el de los números en punto flotante. Los números en punto flotante forman un conjunto F, el cual está caracterizado por cuatro parámetros: un número base B, un número de precisión t y un rango de exponente (L, U), de tal forma que cada número en punto flotante x en F puede representarse como:

$$x = \pm \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \frac{d_3}{\beta^3} + \frac{d_4}{\beta^4} + \ldots + \frac{d_t}{\beta^t}\right) \cdot \beta^e$$

donde los enteros $d_1, d_2, \ldots, d_t$ satisfacen que:

$$0 \leq d_i \leq \beta - 1 \quad \forall\, i = 1, \ldots, t$$

$$y \; L \leq e \leq U$$

En todos aquellos casos donde x sea distinta de cero, $x \in F$ y $d_1 \neq 0$, se dice que, según la representación anterior, x está normalizada.

A "e" se le nombra exponente, y al número

$f = \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_t}{\beta^t}\right)$ se le llama "fracción".

La tabla que a continuación se muestra presenta algunos ejemplos de los parámetros más empleados en la representación en Punto Flotante.

COMPUTADORA	β	t	L	U	MACHEPS*
UNIVAC 1108	2	27	-128	127	1.49×10^{-8}
Honeywell 6000	2	27	-128	127	1.49×10^{-8}
PDP-11	2	28	-128	127	7.45×10^{-9}
Control Dta 6600	2	48	-976	1070	7.11×10^{-15}
Cray-1	2	48	-16384	8191	7.11×10^{-15}
Illiac-IV	2	48	-16384	16383	7.11×10^{-15}
Setun (rusa)	3	18	?	?	7.74×10^{-9}
Burroughs B5500	8	13	-51	77	1.46×10^{-11}
Hewlett Packard HP-45	10	10	-98	100	1.0×10^{-9}
IBM 360 y 370 (short precition)	16	6	-64	63	9.54×10^{-7}
IBM 360 y 370 (long precition)	16	14	-64	63	2.22×10^{-16}
Maniac II	65536	2.69	-7	7	7.25×10^{-9}

*La columna macheps representa un valor aproximado de β^{1-t}

El conjunto de números en punto flotante F, no es contínuo, ni siquiera finito, y contiene exactamente $2(\beta-1)\,\beta^{t-1}\,(U-L+1) + 1$ números; además, estos números no se encuentran igualmente espaciados a lo largo de su rango de valores. Unicamente para potencias sucesivas de β, el espaciamiento es el mismo.

Según la fórmula anterior

F_{IBM} contiene $1.7293823 \times 10^{19} + 1$ números

F_{univac} contiene $3.4359738 \times 10^{10} + 1$ números

F_{CDC} contiene $5.7617928 \times 10^{17} + 1$ números

La figura 1 muestra un conjunto hipotético F de 33 puntos para el caso de un sistema donde $\beta = 2$, $t = 3$, $L = -1$, y $U = 2$. En la misma figura puede observarse que no todos los números reales pueden ser representados en este sistema hipotético. Por lo tanto, cada número en F debe de representar un intervalo completo de números reales. Si x es un número real el cual cae dentro de un cierto intervalo de valores en F, entonces representaremos con $f_\ell(x)$ al número en F más cercano a x.

El error relativo en la aproximación, puede demostrarse, queda expresado como una función de los parámetros β y t:

$$\left|\frac{f_\ell(x) - x}{x}\right| \leq \frac{1}{2}\beta^{1-t}$$

Considérese el ejemplo siguiente. El número decimal 0.1 es seleccionado frecuentemente como incremento en ciertos algoritmos iterativos. Pregunta: ¿són 10 pares de tamaño 0.1 equivalentes a un par de tamaño 1.0? y la respuesta es ¡No!, no al menos en un sistema de punto flotante cuya base sea dos ($\beta = 2$), o una potencia de 2. Esto es debido a que 0.1 no tiene una representación finita en potencias de $\frac{1}{2}$, esto es:

$$\frac{1}{10} = \frac{0}{2^1} + \frac{0}{2^2} + \frac{0}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \ldots$$

$$(0.1)_{10} = (0.000110011001100\ldots)_2$$

$$(0.121212121212121\ldots)_4$$

$$(0.063146314631463\ldots)_8$$

$$(0.199999999999999\ldots)_{16}$$

donde los subíndices denotan la base β. Las cantidades a la derecha han

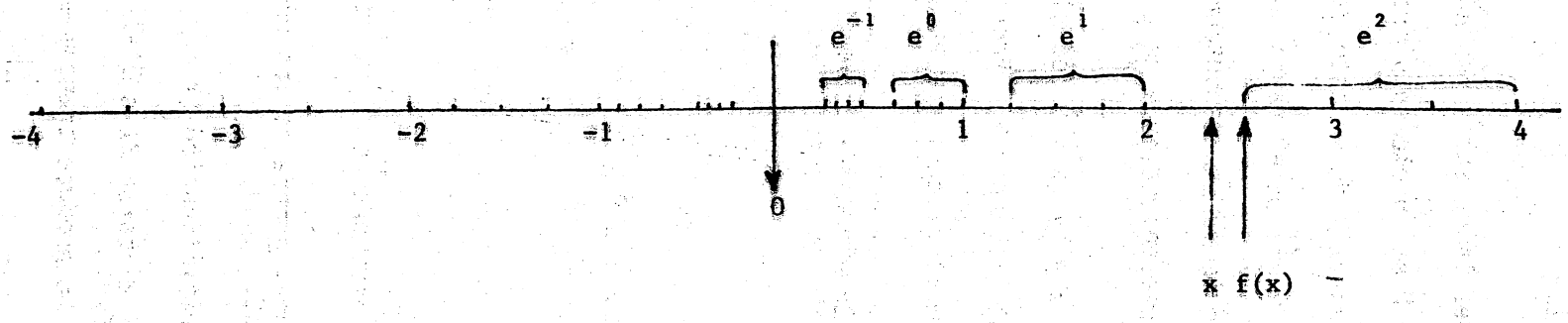


Figura 1. Conjunto hipotético de 33 números en punto flotante. Números entre llaves están igualmente espaciados. Un valor real x es aproximado por el número f(x) más cercano en el conjunto.

sido truncadas después de t dígitos, y cuando 10 de ellas se han sumado, el resultado no ha sido 1.

Otra operación común es la de sumar dos números en punto fijo x e y cuyo resultado $x \oplus y \notin F$ es un elemento frecuentemente no en F: el valor real de la suma es aproximado en este caso por $f_\ell(x + y)$. Lo ideal sería que si x + y estuviera en el rango de F, se tendría $x + y = x \oplus y = f_\ell(x + y)$. En la mayoría de las computadoras, este ideal es obtenido o casi obtenido para ciertos valores de x y y.

La diferencia entre $x \oplus y$ y $x + y$ es el error de redondeo introducido por la suma en punto flotante $\oplus$. Propiedades similares pueden observarse en la resta, multiplicación y en la división.

En el ejemplo de los 33 elementos podemos observar que:

$$\frac{5}{4} \in F, \frac{3}{8} \in F$$

$$\frac{5}{4} + \frac{3}{8} = \frac{13}{8} \neq \frac{5}{4} \oplus \frac{3}{8} = \frac{3}{2} \text{ ó } \frac{7}{4}$$

$$\frac{13}{8} - \frac{3}{2} = \text{error de truncamiento o de redondeo}$$

y $\quad \frac{7}{2} + \frac{7}{2} = 7$ no está dentro de F, ya que 7 es mayor que el más grande de los números generados (Overflow).

La operación de la multiplicación $(x \cdot y)$ puede producir "Overflows" más frecuentemente, ya que ella abarca 2t o 2t-1 dígitos significativos. Por lo tanto la multiplicación produce truncamientos con más frecuencia

que la adición.

NOTA.- En el ambiente de una computadora las operaciones en punto flotante de suma y multiplicación son conmutativas.

La exactitud de la operación suma en punto flotante puede ser caracterizada por el término "Machine-epsilon", o sea el número ε más pequeño, tal que: $1 \oplus \varepsilon > 1$

Existen diversas formas de computar ε (o un valor aproximado de ε)

```
      EPS   = 1
   10 EPS   = 0.5 * EPS
      EPSP1 = EPS + 1.
      IF(EPSP1.GT.1) GO TO 10
```

El siguiente es un ejemplo de error por truncamiento. Sea la función e^x. Deseamos programar un algoritmo que permita efectuar el cálculo de e^x para cualquier número x en punto flotante.

Expandiendo e^x en series se tiene:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots.$$

Si $\beta = 10$ y $t = 5$ caracterizan el sistema, y se desea calcular el valor de $e^{-5.5}$, substituyendo en la serie se tiene

```
e^-5.5 =  1.0000
         -5.5000
          15.125
         -27.730
          38.129
```

$$
\begin{array}{r}
-41.942 \\
38.446 \\
-30.208 \\
20.768 \\
-12.692 \\
6.9803 \\
-3.4902 \\
1.5997 \\
\vdots \\
\hline
0.0026363
\end{array}
$$

La suma ha sido calculada empleando únicamente 25 términos ya que los términos subsecuentes no la modifican. ¿Es la respuesta satisfactoria? Sabemos que el resultado correcto es

$$e^{-5.5} = 0.00408677$$

Nótese, por otro lado, que algunos de los términos son mayores, por varias veces, a la respuesta final. Por ejemplo el número 38.129 tiene ya, en sí mismo, un error de truncamiento tan grande como el resultado final. En efecto, el cuarto dígito decimal se ha perdido

$$38.129\ ?$$

mismo que juega un papel importante en el resultado final.

Una solución, aunque costosa, sería la de efectuar las operaciones empleando un número mayor de dígitos significativos. Sin embargo, una solución más práctica sería la de calcular $e^{5.5}$ y luego obtener su

recíproco,

$$e^{-5.5} = \frac{1}{e^{5.5}} = \frac{1}{1 + 4.4 + 15.125 + \ldots} = \underbrace{0.0040865}_{t = 5}$$

con lo cual el error se reduciría a un 0.007 por ciento.

Como conclusión podemos observar que el cómputo de ciertas operaciones no tiene que ser necesariamente complicado para incurrir en serios errores de truncamiento.

Ejercicio # 1

Evaluación de la Función Error.

El propósito de este ejercicio es mostrar que algunas funciones matemáticas pueden ser, en ocasiones, difíciles de calcular y que diversas aproximaciones pueden proporcionar resultados completamente diferentes.

Un ejemplo de tales funciones es la función error, empleada en diversos problemas de Ingeniería Petrolera, Estadística, etc.; la cual se define como:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, dt$$

Dado que muchos sistemas computacionales tienen esta función implementada, el problema puede darse por resuelto. Pero supongamos que la función no está disponible en uno de estos sistemas y que uno trata por diferentes medios de computarla.

A continuación se mencionan tres métodos, los cuales varían en eficiencia, exactitud y requerimientos de memoria. Escriba un programa el cual emplee cada uno de los métodos siguientes en el cálculo de la función error. Deberá evaluar erf(x) para argumentos de x en el rango $0 \leq x \leq 5$, a intervalos de 0.5. En aquellos métodos donde se consideran series infinitas, dé el resultado empleando 5 términos y 10 términos. Evalúe la función error empleando la función ERF( ) del sistema y compárela con sus resultados calculados.

Use simple-precisión en todos sus cálculos. Entregue el listado del programa con resultados, así como una breve descripción de éstos.

METODO 1

La serie de Taylor es frecuentemente empleada en la representación de muchas funciones. Por ejemplo, empleando tal serie en la función exponencial, uno puede substituir en la función error

$$\text{erf}(x) = 2/\sqrt{\pi} \int_0^x (1 - t^2 + t^4/2! + \ldots)$$

e integrar cada término.

Escriba la serie resultante y úsela en la evaluación de erf (x).

METODO 2

Una razón por la cual la serie de Taylor es inexacta para valores grandes de x es que existe un error substancial por cancelación, debido a la naturaleza alternante de la serie. Si se integrara por partes, se obtendría la serie

$$\text{erf}\ (x) = \frac{2x\ e^{-x^2}}{\sqrt{\pi}} \left(1 + \frac{2x^2}{1\cdot 3} + \frac{(2x^2)^2}{1\cdot 3\cdot 5} + \frac{(2x^2)^3}{1\cdot 3\cdot 5\cdot 7} + \ldots\right)$$

Esta serie no es alternante y por lo tanto no sufre del problema de cancelación. Use esta serie en la evaluación de erf (x). ¿Qué opina de ella desde el punto de vista de su eficiencia?

METODO 3

Finalmente, uno puede intentar una función de aproximación. Por ejemplo, una función racional en ocasiones es más exacta cuando el mismo número de coeficientes (comparada con la serie) es empleado.

Tal aproximación podría ser:

$$\text{erf}(x) \simeq 1 - \frac{1}{(1 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4)^4}$$

donde

$a_1 = 0.278393$ $\qquad$ $a_2 = 0.230389$

$a_3 = 0.000972$ $\qquad$ $a_4 = 0.078108$

Evalúe erf (x) y diga si ésta es más exacta o no que la aproximación de los dos casos anteriores.

CAPITULO II

SISTEMA DE ECUACIONES LINEALES Y NO-LINEALES

En prácticamente todas las ramas de la ingeniería uno de los problemas más frecuentemente encontrados es el de la solución de sistemas de ecuaciones lineales. Tales sistemas se representan en forma matricial como Ax = b, donde A indica una matriz cuadrada de orden nxn, b un vector columna de n términos independientes y x un vector columna de n componentes desconocidos.

En cursos de Algebra Lineal el lector ha podido aprender diferentes técnicas para la solución de tales sistemas. La regla de Cramer y el método de eliminación Gaussiana son dos ejemplos de dichas técnicas, las cuales no analizaremos en este curso. La gran mayoría de los programas de cómputo diseñados para la solución de sistemas de ecuaciones lineales se basan precisamente en el método de eliminación Gaussiana. Un ejemplo de estos programas son los programas DECOMP y SOLVE, enlistados al final del capítulo. Una ventaja de los programas DECOMP y SOLVE sobre otros programas convencionales es la de, aparte de evaluar el vector columna x, poder calcular el número de la condición de la matriz A. Este número indica qué tan cercana está la matriz de ser singular. Un número de condiciones muy "grande" indicaría una matriz sumamente inestable, es decir, que cualquier pequeño cambio en alguno de sus coeficientes produciría en x un resultado totalmente diferente. Este número asigna de cierto modo un grado de confiabilidad en la evaluación del vector x. A una matriz no-singular, el programa DECOMP le aso-

cia un número de condición igual a 1, y a una matriz singular un número de condición igual a 10^{32}. Cualquier otro número entre 1 y 10^{32} indicaría la posición relativa de la matriz con respecto a la no-singularidad o a la singularidad.

El número de la condición de la matriz A se define como

$$\text{Cond (A)} = \frac{\max_x \frac{||Ax||}{||x||}}{\min_x \frac{||Ax||}{||x||}}$$

donde $||\ x\ ||$ indica la norma del vector x.

Es importante mencionar que aún cuando DECOMP y SOLVE pueden resolver cualquier sistema de forma Ax = b, existen otros algoritmos que son más eficaces bajo determinadas circunstancias particulares. Por ejemplo si la matriz A fuera tridiagonal, el algoritmo de Thomas proporcionaría la solución del vector x a través de un número de operaciones mucho menor al requerido por el método de eliminación Gaussiana.

Algoritmo de Thomas.

Sea A una matriz tridiagonal de forma

$$A = \begin{vmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & c_3 & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot \\ & & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & & & a_n & b_n \end{vmatrix}$$

entonces A puede ser factorizada como el producto de dos matrices, A = LU donde

$$L = \begin{vmatrix} \alpha_1 & & & & & \\ a_2 & \alpha_2 & & & & \\ & a_3 & \alpha_3 & & 0 & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ 0 & & & & \cdot & \cdot \\ & & & & & a_n & \alpha_n \end{vmatrix}$$

y

$$U = \begin{vmatrix} 1 & \beta_1 & & & & \\ & 1 & \beta_2 & & 0 & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ 0 & & & & \cdot & \cdot \\ & & & & & 1 & \beta_{n-1} \\ & & & & & & 1 \end{vmatrix}$$

Si esto es cierto cada elemento de la matriz LU será igual al elemento respectivo en la matriz A,

$$LU = \begin{vmatrix} \alpha_1 & 1\beta_1 & & & & \\ a_2 & a_2\beta_1+\alpha_2 & \alpha_2\beta_2 & & & \\ & a_3 & a_3\beta_2+\alpha_3 & \alpha_3\beta_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-1} & a_{n-1}\beta_{n-2}+\alpha_{n-1} & \alpha_{n-1}\beta_{n-1} \\ & & & & a_n & a_n\beta_{n-1}+\alpha_n \end{vmatrix}$$

es decir,

$\alpha_1 = b_1$

$\alpha_i \beta_i = c_i$, $i = 1, 2, \ldots, n-1$

$a_i \beta_{i-1} + \alpha_i = b_i$, $i = 2, 3, \ldots, n$

despejando α_i y β_i del conjunto de ecuaciones anterior

$\alpha_1 = b_1$

$\beta_i = c_i/\alpha_i$, $i = 1, 2, \ldots, n-1$

$\alpha_i = b_i - a_i\beta_{i-1}$, $i = 2, 3, \ldots, n$

obtenemos expresiones para las α'_s y β'_s, elementos de las

matrices L y U.

Un sistema de ecuaciones Ax = f puede expresarse como L Ux = f. Haciendo Ux = y se obtendrá el sistema Ly = f, el cual se resuelve en forma directa por sustitución hacia adelante

$$y_1 = f_1/\alpha_1$$

$$y_i = \frac{f_i - a_i y_{i-1}}{\alpha_i} \quad , \; i = 2, 3, \ldots, n$$

Una vez calculado el vector y se puede evaluar directamente x en el sistema Ux = y, por substitución hacia atrás

$$x_n = y_n$$

$$x_i = y_i - \beta_i \, x_{i+1} \quad , \quad i = n-1, n-2, \ldots, 1$$

En el apéndice A se describen algoritmos similares para la solución de sistemas de ecuaciones lineales cuyas matrices presentan formas bi-tridiagonal, tri-tridiagonal y penta-diagonal.

Aplicación de la Solución de Sistemas de Ecuaciones Lineales al Problema del Flujo Transitorio Uni-dimensional en Yacimientos Confinados

La solución de la ecuación uni-dimensional de difusión hidráulica es requerida en problemas de flujo lineal transitorio (una fase) a través de medios porosos.

$$\frac{\partial^2 p}{\partial x^2} = \frac{1}{\eta} \frac{\partial p}{\partial t}$$

es la ecuación de difusión donde

p - presión

x - coordenada

t - tiempo

η - coeficiente de difusión

Aproximado por diferencias finitas sabemos que

$$\frac{\partial^2 p}{\partial x^2} \simeq \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{\Delta x^2}$$

$$\frac{\partial p}{\partial t} \simeq \frac{p_{i,j+1} - p_{i,j}}{\Delta t}$$

Sustituyendo en la ecuación de difusión podemos resolver explícitamente para $p_{i,j+1}$.

Obviamente es necesario conocer las condiciones de frontera para que el problema esté bien definido; sean éstas por ejemplo

$p(0,t) = p(1,t) = p^*$, $\forall t > 0$

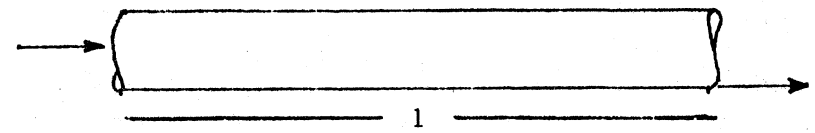
$p(x,0) = p_0(x)$, $\forall x$

Substituyendo en la ecuación de difusión las aproximaciones por diferencias finitas, obtendremos en forma explícita

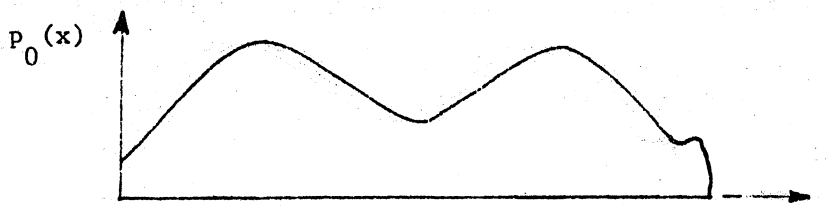
$$p_{i,j+1} = p_{i,j} - \frac{\Delta t \eta}{\Delta x^2} (p_{i+1,j} - 2p_{i,j} + p_{i-1,j})$$

donde $p_{i,j}$, es el valor de la presión P en el nodo $i\Delta p$, $j\Delta t$

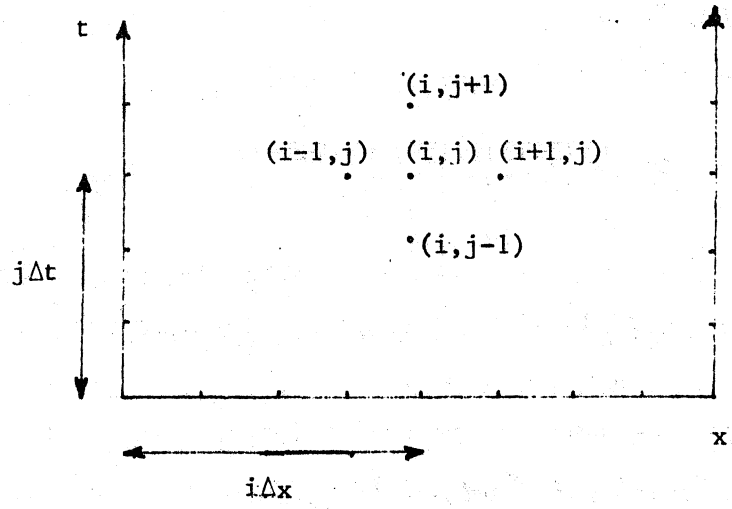
La fórmula anterior representa un esquema explícito de fácil solución, pero el cual no ofrece las mejores características de convergencia y estabi-



(a)



(b)



(c)

Figura 2. (a) Sección cilíndrica del material representando condiciones frontera. (b) $p_0(t)$ es la función condiciones iniciales. (c) Posición de los nodos donde una solución aproximada será calculada

lidad.

DEFINICION.

Convergencia: Si en un punto (x_i, t_j) se obtiene que

$$\lim_{\Delta x,\, \Delta t \to 0} |P_{i,j} - P(x_i, t_j)| \cong 0$$

para un determinado esquema, entonces se dice que el esquema es convergente.

Estabilidad:

Fijos Δx y Δt, y observando el comportamiento del algoritmo a medida que $t \to \infty$, es decir, observando que los errores no se amplifiquen cuando $t \to \infty$, entonces, se dice que el esquema es estable.

$$\lim_{t \to \infty} |P_{i,j} - P(x_i, t_j)| < 1$$

En el caso particular del esquema explícito anterior, si $\Delta t/\Delta x^2 \leq 1/2$, entonces habrá convergencia y estabilidad en el algoritmo.

Existen otros esquemas en los cuales los valores de Δt y Δx pueden ser seleccionados independientemente. Aproximando la derivada $\partial^2 p/\partial x^2$ por incrementos en el tiempo j+1, y la derivada $\partial p/\partial t$ por incremento en el tiempo j+1, tenemos:

$$\partial^2 p/\partial x^2 = \frac{p_{i+1,j+1} - 2p_{i,j+1} + p_{i-1,j+1}}{\Delta x^2}$$

$$\partial p/\partial t = \frac{p_{i,j+1} - p_{i,j}}{\Delta t}$$

o bien

$$\frac{P_{i,j+1} - P_{i,j}}{\eta\Delta t} = \frac{P_{i+1,j+1} - 2P_{i,j+1} + P_{i-1,j+1}}{\Delta x^2}$$

Esta ecuación tiene como incógnitas $p_{i,j+1}$, $p_{i-1,j+1}$ y $p_{i+1,j+1}$ y puede escribirse de la manera siguiente:

$$P_{i-1,j+1} - \rho P_{i,j+1} + P_{i+1,j+1} = \frac{-\Delta x^2}{\eta\Delta t} P_{i,j}, \quad \text{para } i = 1, 2, \ldots, n-1.$$

donde $\rho = \frac{1}{\eta\Delta t} + \frac{2}{\Delta x^2}$

además, dadas las condiciones frontera

$$p_{0,j} = p^* = p_{n,j} \quad , \quad \forall j$$

y $p_{i,0} = p_0 \; i\Delta x \quad , \quad \forall i$

El sistema de ecuaciones anterior puede escribirse en forma matricial como

$$\begin{vmatrix} -\rho & 1 & & & & \\ 1 & -\rho & 1 & & & \\ & 1 & -\rho & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -\rho & 1 \\ & & & & 1 & -\rho \end{vmatrix} \begin{vmatrix} P_{1,j+1} \\ P_{2,j+1} \\ P_{3,j+1} \\ \vdots \\ P_{n-2,j+1} \\ P_{n-1,j+1} \end{vmatrix} = -\frac{\Delta x^2}{\eta\Delta t} \begin{vmatrix} P_{1,j} + \frac{\eta\Delta t}{\Delta x^2} p^* \\ P_{2,j} \\ \vdots \\ P_{n-2,j} \\ P_{n-1,j} + \frac{\eta\Delta t}{\Delta x^2} p^* \end{vmatrix}$$

El resultado es un sistema tridiagonal, cuya solución es fácil de hallar empleando el Algoritmo de Thomas.

Este esquema, puede demostrarse, es incondicionalmente estable.

Es posible, más aún, introducir un refinamiento extra (Esquema Crank-Nicolson), tomando diferencias centrales en el tiempo

$$j + \frac{1}{2}$$

$$\frac{\partial p}{\partial t} = \frac{P_{i,j+1} - P_{i,j}}{\Delta t}$$

y promediando la aproximación de $\frac{\partial^2 p}{\partial^2 x}$ en el tiempo $j+\frac{1}{2}$, con lo que se obtendría:

$$\frac{P_{i,j+1} - P_{i,j}}{\eta \Delta t} = \frac{\frac{1}{2}(P_{i+1,j+1} - 2P_{i,j+1} - P_{i-1,j+1}) + \frac{1}{2}(P_{i+1,j} - 2P_{i,j} + P_{i-1,j})}{\Delta x^2}$$

Puede demostrarse que este sistema, como el anterior, es tridiagonal.

Ejercicio # 2

Un ingeniero petrolero requiere diseñar la sección de la torre de perforación mostrada en la figura 3

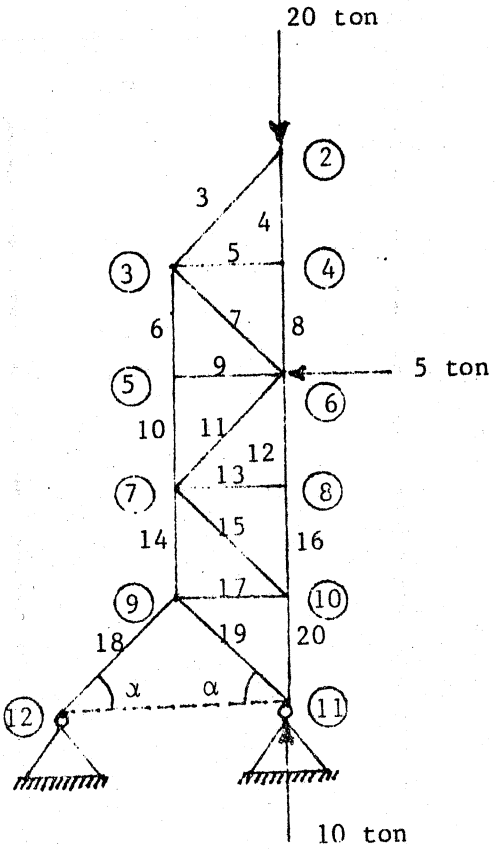


Figura 3. Diseño torre de perforacion

Como parte del diseño resulta necesario conocer las fuerzas que actúan en cada pieza a fin de poder determinar el diámetro de la pieza. En la parte superior de la estructura una fuerza de 20 toneladas simula el efecto del levantamiento de la tubería de perforación cuando ésta se introduce o extrae. En la parte media de la estructura una fuerza de 5

toneladas simula el efecto debido al apoyo de las tuberías contra la estructura. Y, en la parte inferior, una fuerza de 10 toneladas simula el peso de la estructura.

Si F_x denota las componentes de las fuerzas horizontales y Fy las componentes de las fuerzas verticales, y si se consideran condiciones estáticas de equilibrio, el problema puede plantearse para su solución de la manera siguiente:

Junta 2 $\{ \sum Fy = f_4 + rf_3 - 20 = 0$

Junta 3 $\begin{cases} \sum Fx = rf_3 + f_5 + rf_7 = 0 \\ \sum Fy = f_6 + rf_7 - rf_3 = 0 \end{cases}$

Junta 4 $\begin{cases} \sum Fx = f_5 = 0 \\ \sum Fy = f_8 - f_4 = 0 \end{cases}$

Junta 5 $\begin{cases} \sum Fx = f_9 = 0 \\ \sum Fy = f_{10} - f_6 = 0 \end{cases}$

Junta 6 $\begin{cases} \sum Fx = f_9 + rf_7 + rf_{11} - 5 = 0 \\ \sum Fy = f_{12} + rf_{11} - rf_7 - f_8 = 0 \end{cases}$

Junta 7 $\begin{cases} \sum Fx = f_{13} + rf_{11} + rf_{15} = 0 \\ \sum Fy = f_{14} + rf_{15} - rf_{11} - f_{10} = 0 \end{cases}$

Junta 8 $\begin{cases} \sum Fx = f_{13} = 0 \\ \sum Fy = f_{16} - f_{12} = 0 \end{cases}$

Junta 9 $\begin{cases} \sum Fx = rf_{18} - rf_{19} - f_{17} = 0 \\ \sum Fy = rf_{18} + rf_{19} - f_{14} = 0 \end{cases}$

$$\text{Junta 10} \quad \begin{cases} \sum Fx = f_{17} + rf_{15} = 0 \\ \sum Fy = f_{20} - rf_{15} - f_{16} = 0 \end{cases}$$

$$\text{Junta 11} \quad \{ \sum Fy = 10 - rf_{19} - f_{20} = 0$$

Escriba un programa que emplee las subrutinas DECOMP y SOLVE en la solución del problema planteado.

Métodos Iterativos para la Solución de Sistemas de Ecuaciones No-Lineales.

Sea

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ f_2(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned} \qquad (1)$$

un sistema de n ecuaciones no-lineales, real, con n incógnitas, donde la función $f_i(x) = 0$ es la i-ésima función real no lineal, y sea $\alpha = [\alpha_1, \ldots, \alpha_n]^t$ el vector solución buscado. Si el vector α es la solución, entonces se debe satisfacer que $f_i(\alpha) = 0$, $\forall\ i = 1, \ldots, n$

Consideremos ahora las n funciones $F_i(x)$ definidas de tal manera que

$$x_i = F_i(x), \quad \forall\ i = 1, \ldots, n \qquad (2)$$

las cuales implican que

$$f_j(F_1(x), F_2(x), \ldots, F_n(x)) = 0, \ \forall\ j = 1, \ldots, n$$

Lo que se pretende aquí es reordenar el sistema (1) en un nuevo sistema más "conveniente" (2).

En particular, observando la expresión (2), se tiene que $\alpha_i = F_i(\alpha)$. Sea el vector $x_0 = [x_{1,0}, \ldots, x_{n,0}]^t$ una aproximación inicial del vector solución α.

Aproximaciones sucesivas pueden definirse a partir de la función o sistema (2), de la manera siguiente:

$$x_{i,k+1} = F_i(x_k) \qquad (3)$$

Supongamos ahora la existencia de una región R donde $|x_j - \alpha_j| \leq h$, $\forall\ j = 1, \ldots, n$, y que para toda $x \in R$ existe un número positivo $\mu < 1$, tal que

$$\sum_{j=1}^{n} \left| \frac{\partial F_i(x)}{dx_j} \right| \leq \mu, \quad \forall\ i = 1, \ldots, n.$$

Demostraremos a continuación que si la aproximación inicial del vector α (vector x_0) está dentro de la región R, entonces la función (3) converge a la solución del sistema (1), esto es

$$\lim_{k \to \infty} x_k = \alpha$$

Empleando el teorema del valor-medio tenemos que:

$$x_{i,k} = F_i(x_{k-1})$$

$$\alpha_i = F_i(\alpha)$$

$$x_{i,k} - \alpha_i = F_i(x_{k-1}) - F_i(\alpha)$$

$$x_{i,k} - \alpha_i = \sum_{j=1}^{n} (x_{j,k-1} - \alpha_j) \frac{\partial F_i}{\partial x_j} \left[\alpha + \zeta_{i,k-1} (x_{k-1} - \alpha)\right] \qquad (4)$$

donde

$$0 < \zeta_{i,k-1} < 1$$

Tomando valor absoluto en ambos lados de la expresión (4)

$$\left| x_{i,k} - \alpha_i \right| = \left| \sum_{j=1}^{n} (x_{j,k-1} - \alpha_j) \frac{\partial F_i}{\partial x_j} \right|$$

y considerando la desigualdad del triángulo

$$\left| x_{i,k} - \alpha_i \right| \leq \sum_{j=1}^{n} \left| x_{j,k-1} - \alpha_j \right| \left| \frac{\partial F_i}{\partial x_j} \right|$$

Ahora, si el punto $x_{j,k-1}$ cae dentro de la región R, se tiene que

$$\left| x_{i,k} - \alpha_i \right| \leq h \sum_{j=1}^{n} \left| \frac{\partial F_i}{\partial x_j} \right| \leq \mu h < h \ , \quad \forall\ i = 1, \ldots, n$$

mostrándose con ésto que el punto x_k cae también dentro de la región R. Igualmente, de la expresión anterior

$$\left| x_{i,k} - \alpha_i \right| \leq \sum_{j=1}^{n} \left| x_{j,k-1} - \alpha_j \right| \left| \frac{\partial F_i}{\partial x_j} \right|$$

y procediendo por inducción, tenemos que

$$\left| x_{i,k} - \alpha_i \right| \leq \sum_{j=1}^{n} \max_{j} \left(\left| x_{j,k-1} - \alpha_j \right| \right) \left| \frac{\partial F_i}{\partial x_j} \right|$$

donde "max" representa el mayor de todos los valores absolutos $\left| x_{j,k-1} - \alpha_j \right|$. Efectuando operaciones

$$\left| x_{i,k} - \alpha_i \right| \leq \max_{j} \left(\left| x_{j,k-1} - \alpha_j \right| \right) \sum_{j=1}^{n} \left| \frac{\partial F_i}{\partial x_j} \right|$$

ó $\quad \left|x_{i,k} - \alpha_i\right| \leq \mu \max_j \left(\left| x_{j,k-1} - \alpha_j\right|\right) \leq \mu^k h$

y ya que si ésto es cierto, también se debe de cumplir que

$$\left|x_{j,k-1} - \alpha_j\right| \leq \mu \max_j \left(\left|x_{j,k-2} - \alpha_j\right|\right) \ldots \text{ etc,}$$

hasta $\left|x_{j,1} - \alpha_j\right| \leq \mu \max_j \left(\left|x_{j,0} - \alpha_j\right|\right) \leq \mu h$

Finalmente, $\mu < 1$ implica que $\lim_{k\to\infty} \mu^k h = 0$, o lo que es lo mismo

que $\lim_{k\to\infty} \left|x_{i,k} - \alpha_i\right| = 0$

Si en particular las funciones $F_i(x)$ fuesen lineales, entonces una solución más simple sería emplear el método de Jacobi.

Ejemplo.

Sean $f_1(x_1, x_2) = \frac{1}{2} \operatorname{sen} x_1 x_2 - \frac{x_2}{4\pi} - \frac{x_1}{2} = 0$

y $\quad f_2(x_1, x_2) = (1 - \frac{\pi}{4})(e^{2x_1} - e) + ex_2/\pi - 2ex_1 = 0$

Reescribiendo las ecuaciones en la forma del sistema (2)

$$x_1 = F_1(x_1, x_2) = \operatorname{sen} x_1 x_2 - x_2/2\pi$$

$$x_2 = F_2(x_1, x_2) = 2\pi x_1 - (\pi - 1/4)(e^{2x_1 - 1} - 1)$$

Escogiendo como valores iniciales $x_{1,0} = 0.4$ y $x_{2,0} = 3.0$

Iteración Jacobi

$$\begin{cases} x_{1,1} = F_1(x_{1,0}, x_{2,0}) = \operatorname{sen}(1.2) - 3/2\pi = .455 \\ x_{2,1} = F_2(x_{1,0}, x_{2,0}) = 2\pi(.4) - (\pi - 1/4)(e^{-.2} - 1) = 3.03 \end{cases}$$

Iteración Gauss-Seindel

$$x_{1,1} = F_1(x_{1,0}, x_{2,0})$$
$$x_{2,1} = F_2(x_{1,1}, x_{2,0})$$
$$x_{1,2} = F_1(x_{1,1}, x_{2,1})$$
$$x_{2,2} = F_2(x_{1,2}, x_{2,1})$$
$$\vdots \qquad \vdots$$

Con lo cual la convergencia se aceleraría.

Método de Newton-Raphson en la Solución de Sistema de Ecuaciones No-Lineales.

Partiendo del sistema de ecuaciones no-lineales (1), podemos definir la matriz $\emptyset(x)$ cuyos elementos estén dados por la expresión

$$\emptyset_{ij}(x) = \frac{\partial F_i(x)}{\partial x_j}$$

El determinante de la matriz $\emptyset(x)$ no es otro que el Jacobiano del sistema evaluado en el vector x.

Definamos la función vectorial f(x) como

$$f(x) = (f_1(x), \ldots, f_n(x))^t$$

El proceso iterativo de Newton Raphson puede iniciarse considerando un vector inicial

$$x_0 = (x_{1,0}, \ldots, x_{n,0})^t$$

y un esquema

$$x_{k+1} = x_k + \delta_k,$$

donde δ_k vendría a ser la solución del sistema de ecuaciones lineales

$$\emptyset(x_k)\ \delta_k = -f(x_k)$$

Aplicación al Diseño de un Sistema de Recolección y Distribución de Gas*

El problema del diseño de sistemas de recolección de gas puede resolverse por medio del método de Stoner el cual está basado en la solución de ecuaciones que simulan el flujo de gas en sistemas de recolección. Este método tiene la ventaja de que incorpora en el sistema elementos tales como tuberías, compresoras, válvulas y pozos.

Modelo.

Considérese el sistema de recolección de gas mostrado en la figura 4, el cual consiste de una red de tuberías, compresoras, pozos, etc.

El régimen a considerar es el de flujo permanente. Simplificando el sistema, este puede representarse por nodos y conectores. Los nodos representan puntos donde los elementos del sistema se inician o terminan. Los conectores son los medios que permiten el intercambio de masa de un nodo a otro (figura 5).

Denominemos con la letra P al conjunto de conectores y con la letra n al número de nodos.

*Tomás L.J., 1974. Transporte de Gas en Régimen Permanente. Proyecto D-341A. Publicación No. 74BH/164. Instituto Mexicano del Petróleo.

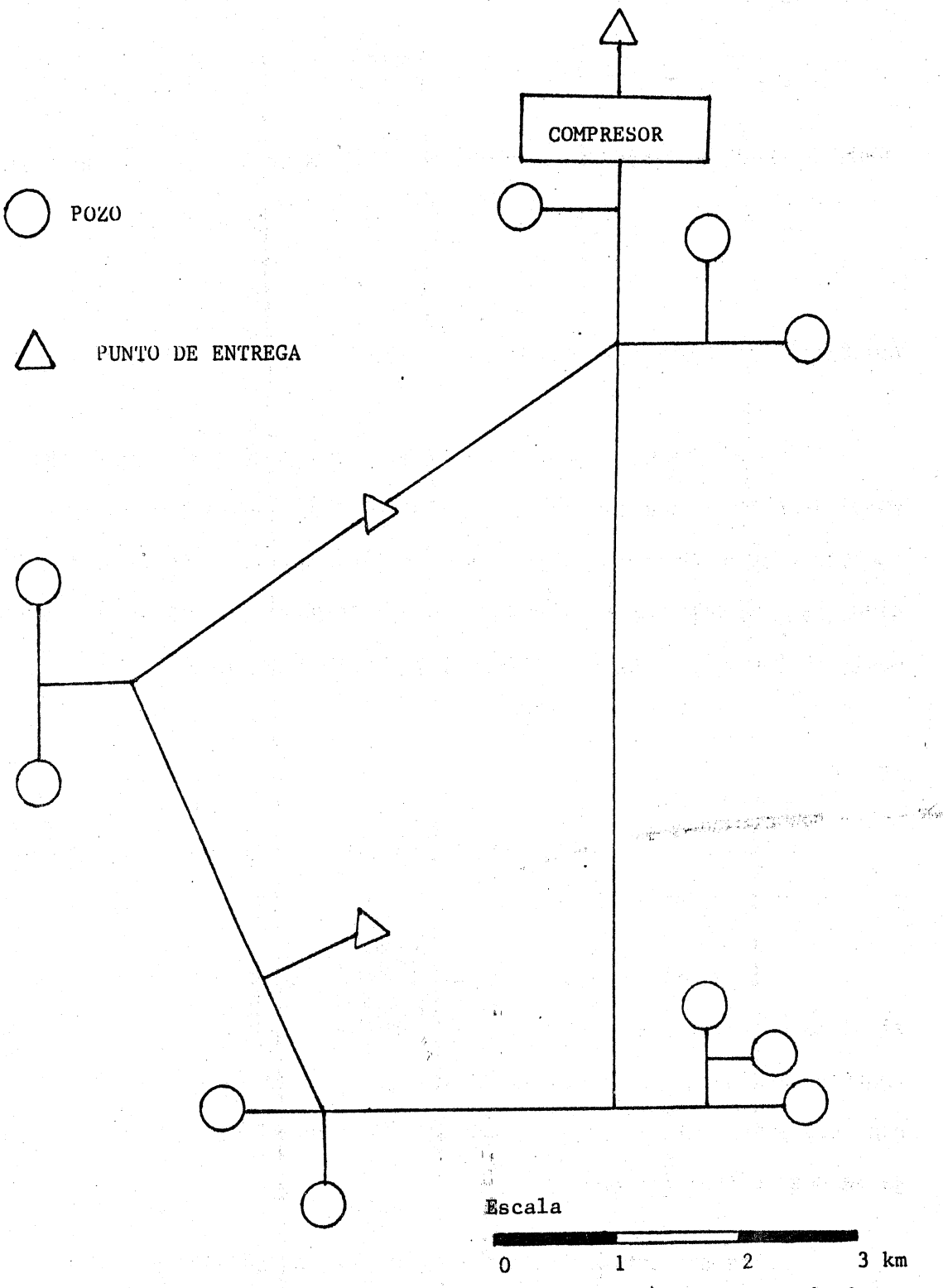


Figura 4. Representación esquemática de un sistema de recolección de gas.

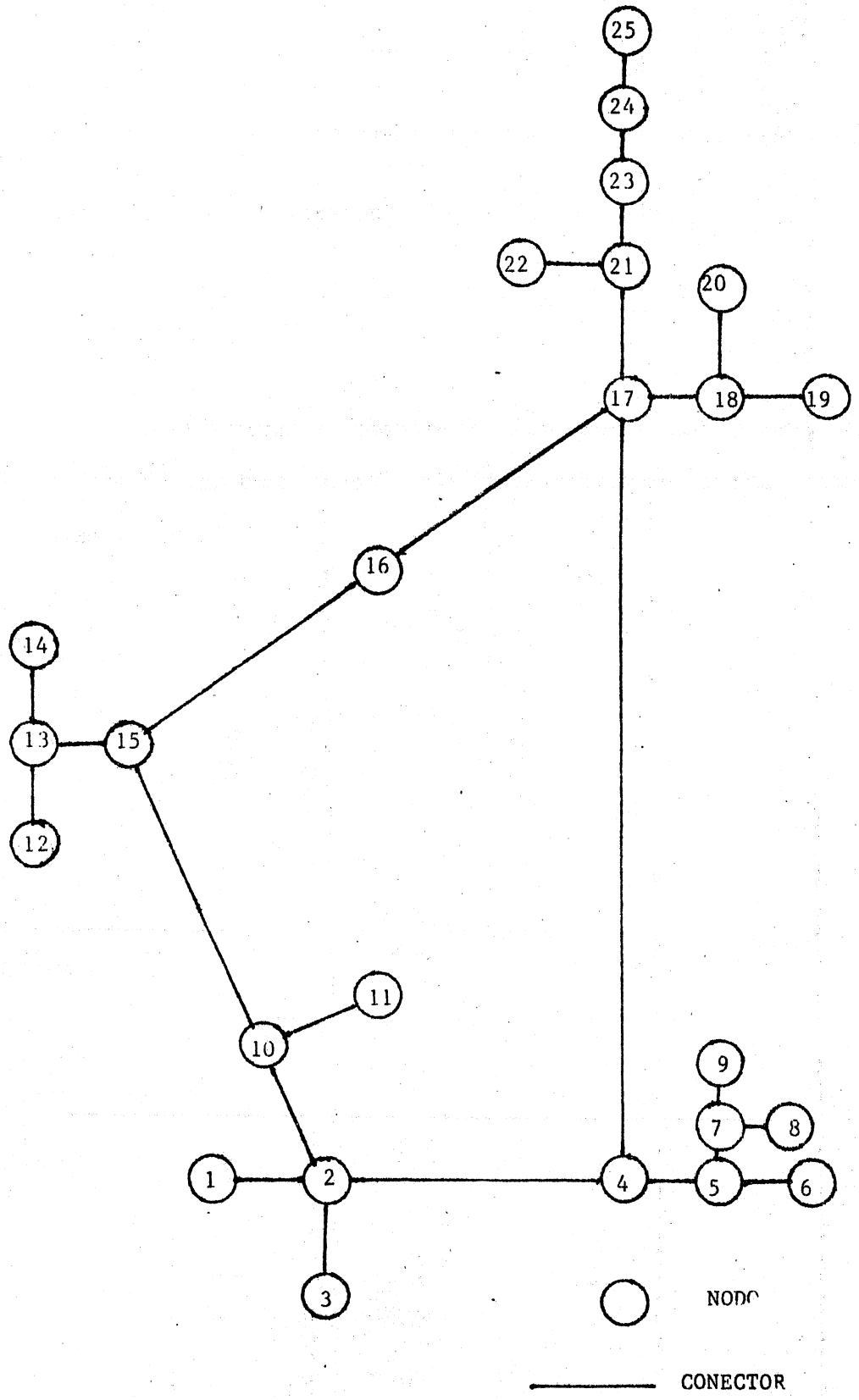


Figura 5. Representación de un sistema de recolección de gas por medio de nodos y conectores.

En un sistema como el descrito, se debe satisfacer la ley de conservación de masa en cada uno de los nodos

$$F_i = \sum_{j/(i,j)\in P} S_{ij} q_{ij} + Q_i = 0, \quad \forall\ i = 1, \ldots, n \qquad (5)$$

donde

S_{ij} es una variable que indica el sentido del flujo

si $S_{ij} = 1$ el flujo va del nodo i al nodo j, y

si $S_{ij} = -1$ el flujo va del nodo j al nodo i

q_{ij} es el gasto de gas que fluye a través del conector de los nodos i y j.

Q_i indica la adición o extracción de masa al sistema a través del nodo i.

Por ejemplo, en el caso del nodo 1 se tendría como ecuación de conservación de masa (figura 5)

$$F_1 = S_{12}\, q_{12} + Q_1 = 1$$

Cada gasto q_{ij} puede expresarse en términos de las diferencias de presiones en los extremos del conector como

$$q_{ij} = C_{ij} \left| P_i^2 - P_j^2 \right|^{\eta} \qquad (6)$$

donde

C_{ij} es un coeficiente de transmisión de la tubería, el cual depende de la geometría del tubo, de las condiciones de flujo y de la composición del gas.

P_i es la presión en el nodo i, y

n es un exponente que depende de la forma de la ecuación.

Substituyendo la expresión (6) en (5) se obtiene

$$F_i = \sum_{j/i,j)\varepsilon P} S_{ij} C_{ij} |P_i^2 - P_j^2|^n + Q_i = 0, \; i \varepsilon n \qquad (7)$$

Considerando el sistema total de nodos y conectores, la suma de los "flujos exteriores" (adición o extracción de masa) deberá ser igual a cero.

$$\sum_{i=1}^{n} Q_i = 0 \qquad (8)$$

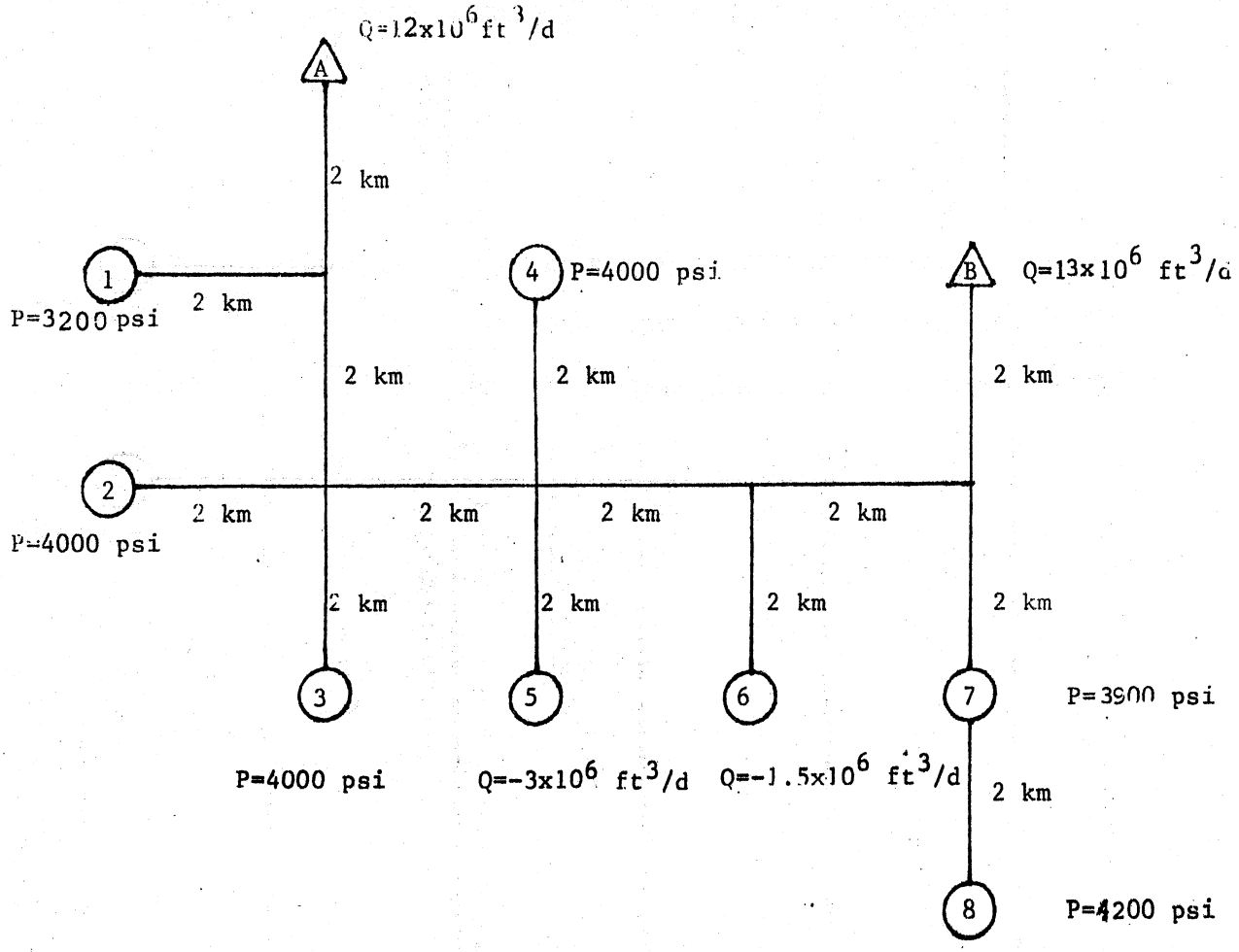


Figura 6. Campo "Las Margaritas". Estudio de un diseño de recolección de gas.

El problema aquí consiste en estimar el conjunto de gastos y presiones, Q_i y P_i para cada nodo tal que éstos satisfagan las expresiones (7 y 8).

El sistema de ecuaciones (7), (la ecuación (8) está implícita en el sistema (7)), define un sistema de n ecuaciones con 2n incógnitas. Por lo tanto, para poder resolver el sistema será necesario asignar n valores, obteniéndose con ello un sistema compatible de n ecuaciones con n incógnitas.

Sin embargo, aún cuando el sistema es compatible, éste es no-lineal, por lo que resulta necesario emplear, en su solución, un método de solución de sistemas de ecuaciones no-lineales, tal como el de Newton-Raphson.

El procedimiento de solución propuesto es el siguiente:

(I).- Asignar valores a n variables, presiones y/o gastos.

(II).- Asignar valores iniciales a las n variables restantes.

(III).- Substituir los valores de los pasos (I) y (II) en el sistema (7) y obtener el valor correspondiente de F_i, $\forall\ i = 1, .., n$

(IV).- Probar si $\max_j |F_i|$ es menor o igual a cierta tolerancia fijada. Si esto ocurre el problema se da por resuelto, siendo los valores del paso (II) la solución. Si esto no ocurre, continuar con el paso siguiente.

(V).- Calcular el valor de las derivadas parciales

$$\left|\frac{\partial F_i}{\partial x_j}\right|_{x_k}$$

y resolver el sistema

$$\emptyset(x_k)\ \delta_k = -f(x_k)$$

donde δ_k representa al vector de las incógnitas.

(VI).- Calcular el nuevo valor de las incógnitas según el esquema

$$x_{k+1} = x_k + \delta_k$$

(VII).- Regresar al paso (III) y repetir el procedimiento.

Observaciones.

Es conveniente hacer notar que el sistema de ecuaciones del paso (V) es lineal y que la matriz resultante contiene una gran cantidad de elementos nulos. Puede observarse también que la estructura de la matriz no varía a lo largo del proceso iterativo.

Ejercicio # 3

El siguiente ejercicio es un ejemplo de aplicación a un caso real.

El campo Las Margaritas, situado en el Distrito Frontera Noreste, está produciendo gas a través de 8 pozos y 17 tuberías según el arreglo de la figura 6. El número entre paréntesis indica la longitud en kilómetros de cada línea. Todos los diámetros interiores son de 2 pulgadas.

La caída de presión en la tubería puede calcularse según la fórmula

$$q_{ij} = 842.69\ E \left| \frac{(P_i^2/Z_i) - (P_j^2/Z_j)}{0.6215\ L_{ij}} \right|^{0.5} d_{ij}^{2.665}$$

donde

q_{ij} = gasto de gas en ft^3/d

E = factor de eficiencia

P_i = presión en el nodo i en psi

Z_i = factor de desviación del gas a la presión P_i

L_{ij} = longitud de la línea en km.

d_{ij} = diámetro de la línea en in.

Usando como base el sistema de recolección de la figura (7), se desea

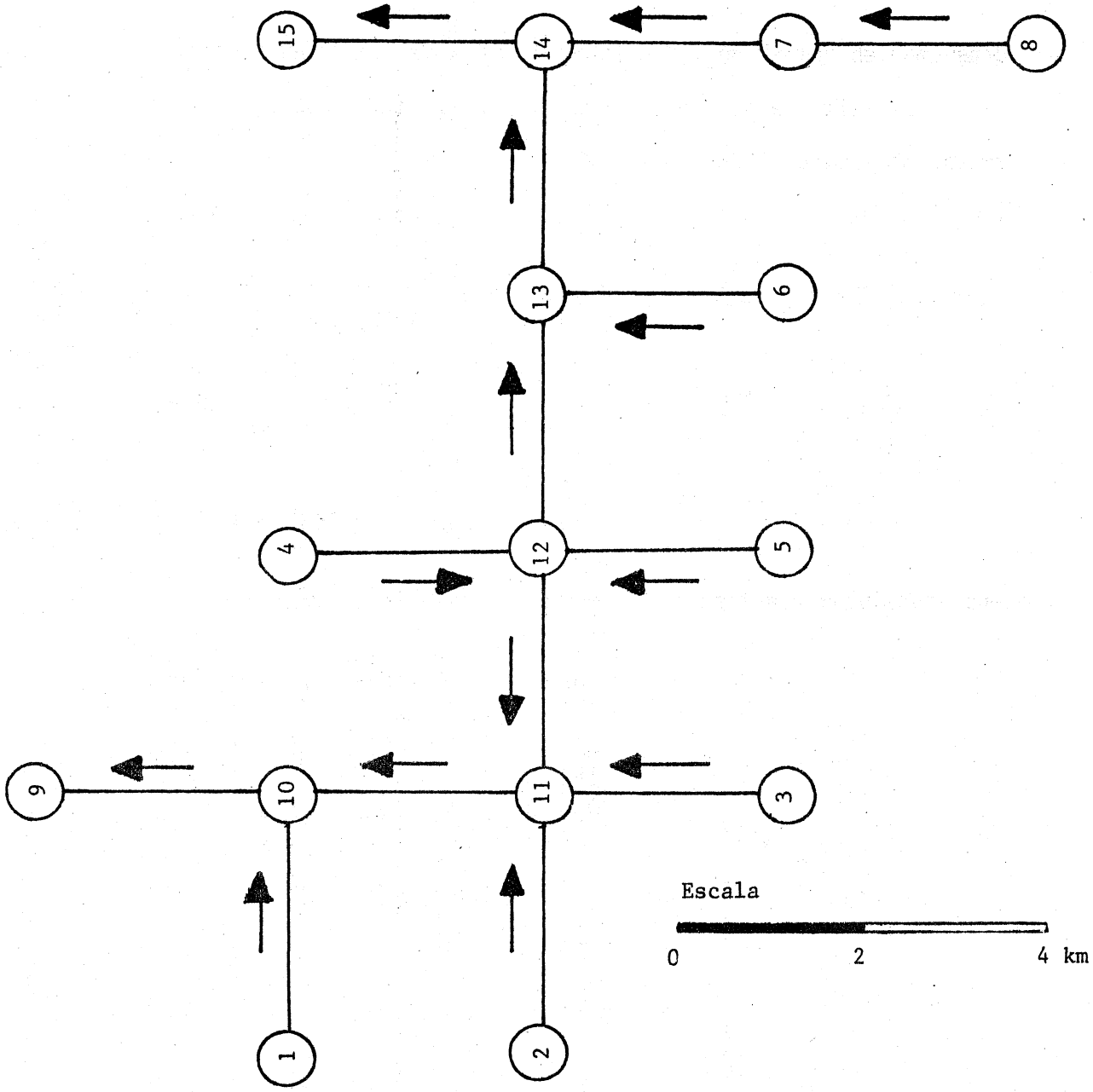


Figura 7. Representación esquemática en términos de nodos y conectores del sistema de recolección en el campo "Las Margaritas".

Las flechas indican el sentido del flujo a través de los conectores.

entregar en el punto A un gasto de 12×10^6 ft^3/d de gas, y en el punto B de 13×10^6 ft^3/d. Además, se conoce la presión en la cabeza de 6 pozos y el gasto en 2 pozos más, tal como se ilustra en la misma figura.

Se desea determinar:

(i) A qué presión se entregará el gas en los puntos A y B.

(ii) Cuál será el gasto Q_i y la presión P_i en cada nodo donde la presión y/o el gasto se desconozcan.

(iii) Cuál será el gasto q_{ij} a través de las líneas.

(iv) El número de iteraciones necesarias para alcanzar la solución.

El diseño de sistemas de distribución en un campo es un problema importante a resolver, ya que permite: (a) analizar el sistema de distribución de gas, (b) cuantificar el efecto que produciría cualquier cambio en el sistema, (c) determinar la capacidad máxima del sistema, y (d) estudiar el comportamiento del yacimiento, considerando la interacción entre los pozos en la superficie.

El sistema de ecuaciones resultante para este caso particular se deriva fácilmente de la ecuación (7)

$$F_i = \sum_{j/(i,j)\varepsilon P} S_{ij}\, q_{ij} + Q_i = 0, \quad i = 1, \ldots, n$$

Substituyendo la ecuación del gasto se tiene

$$F_i = \sum_{j/(i,j)\in P} S_{ij}\ 842.69\ E \left| \frac{P_i^2/Z_i - P_j^2/Z_j}{0.6215\ L_{ij}} \right|^{0.5} d_{ij}^{2.665} + Q_i$$

Asumiendo un factor de eficiencia de 0.80, un factor de compresibilidad constante e igual a 1, y observando que en el sistema $d_{ij} = 2"$ y $L_{ij} = 2$ km $\forall$ i,j, se tiene que la expresión anterior se reduce a

$$F_i = \sum_{j/(i,j)\in P} K\ S_{ij}\ (P_i^2 - P_j^2)^{1/2} + Q_i, \quad \forall\ i = 1, \ldots, n$$

siendo $\sum_{i=1}^{n} Q_i = 0$,

$k = 3835$ y $n = 15$.

En forma explícita el sistema de ecuaciones resultante queda:

para $n = 1$, $k(P_1^2 - P_{10}^2)^{1/2} + Q_1 = F_1$

para $n = 2$, $k(P_2^2 - P_{11}^2)^{1/2} + Q_2 = F_2$

para $n = 3$, $k(P_3^2 - P_{11}^2)^{1/2} + Q_3 = F_3$

para $n = 4$, $k(P_4^2 - P_{12}^2)^{1/2} + Q_4 = F_4$

para $n = 5$, $k(P_5^2 - P_{12}^2)^{1/2} - 3\times10^6 = F_5$

para $n = 6$, $k(P_6^2 - P_{13}^2)^{1/2} - 1.5\times10^6 = F_6$

para $n = 7$, $k(P_7^2 - P_{14}^2)^{1/2} - k(P_7^2 - P_8^2)^{1/2} + Q_7 = F_7$

para $n = 8$, $k(P_8^2 - P_7^2)^{1/2} + Q_8 = F_8$

para $n = 9$, $-k(P_9^2 - P_{10}^2)^{1/2} + 12\times10^6 = F_9$

para n = 10, $-k(P_{10}^2 - P_1^2)^{1/2} - k(P_{10}^2 - P_{11}^2)^{1/2} + k(P_{10}^2 - P_9^2)^{1/2} = F_{10}$

para n = 11, $-k(P_{11}^2 - P_2^2)^{1/2} - k(P_{11}^2 - P_3^2)^{1/2} - k(P_{11}^2 - P_{12}^2)^{1/2} + k(P_{11}^2 - P_{10}^2)^{1/2} = F_{11}$

para n = 12, $k(P_{12}^2 - P_{11}^2)^{1/2} - k(P_{12}^2 - P_5^2)^{1/2} + k(P_{12}^2 - P_{13}^2)^{1/2} - k(P_{12}^2 - P_4^2)^{1/2} = F_{12}$

para n = 13, $-k(P_{13}^2 - P_{12}^2)^{1/2} - k(P_{13}^2 - P_6^2)^{1/2} + k(P_{13}^2 - P_{14}^2)^{1/2} = F_{13}$

para n = 14, $-k(P_{14}^2 - P_{13}^2)^{1/2} - k(P_{14}^2 - P_7^2)^{1/2} + k(P_{14}^2 - P_{15}^2)^{1/2} = F_{14}$

para n = 15, $-k(P_{15}^2 - P_{14}^2)^{1/2} + 13x10^6 = F_{15}$

donde P_1, P_2, P_3, P_4, P_7, y P_8

son presiones conocidas.

Aplique el método de Newton-Raphson para resolver el sistema anterior considerando los siguientes valores iniciales:

$Q_1 = -2.9x10^6 ft^3/d$ $\qquad$ $P_5 = 3990$ psi

$Q_2 = -3.6x10^6 ft^3/d$ $\qquad$ $P_6 = 3700$ psi

$Q_3 = -3.6x10^6 ft^3/d$ $\qquad$ $P_9 = 1000$psi

$Q_4 = -3.1x10^6 ft^3/d$ $\qquad$ $P_{10} = 3100$ psi

$Q_7 = -1.2x10^6 ft^3/d$ $\qquad$ $P_{11} = 3880$ psi

$Q_8 = -6.0x10^6 ft^3/d$ $\qquad$ $P_{12} = 3900$ psi

$P_{13} = 3750$ psi

$P_{14} = 3400$ psi

$P_{15} = 1000$ psi

Stoner ha recomendado, con el propósito de acelerar el proceso de

convergencia, substituir en las primeras iteraciones, el vector

δ_k por el vector $\delta_{\frac{k}{2}}$ o $\delta_k \rightarrow \frac{\delta_k}{2}$, $\forall\ k = 0, 1$

Considere esta recomendación en su programa.

```
      SUBROUTINE DECOMP(NDIM,N,A,COND,IPVT,WORK)                        00000000
C                                                                       00000010
      INTEGER NDIM,N                                                    00000020
      DOUBLE PRECISION A(NDIM,N),COND,WORK(N)                           00000030
      INTEGER IPVT(N)                                                   00000040
C                                                                       00000050
C     DECOMPOSES A DOUBLE PRECISION MATRIX BY GAUSSIAN ELIMINATION      00000060
C     AND ESTIMATES THE CONDITION OF THE MATRIX.                        00000070
C                                                                       00000080
C     USE SOLVE TO COMPUTE SOLUTIONS TO LINEAR SYSTEMS.                 00000090
C                                                                       00000100
C     INPUT..                                                           00000110
C                                                                       00000120
C        NDIM = DECLARED ROW DIMENSION OF THE ARRAY CONTAINING  A.      00000130
C                                                                       00000140
C        N = ORDER OF THE MATRIX.                                       00000150
C                                                                       00000160
C        A = MATRIX TO BE TRIANGULARIZED.                               00000170
C                                                                       00000180
C     OUTPUT..                                                          00000190
C                                                                       00000200
C        A  CONTAINS AN UPPER TRIANGULAR MATRIX  U  AND A PERMUTED      00000210
C          VERSION OF A LOWER TRIANGULAR MATRIX  I-L  SO THAT           00000220
C          (PERMUTATION MATRIX)*A = L*U .                               00000230
C                                                                       00000240
C        COND = AN ESTIMATE OF THE CONDITION OF  A .                    00000250
C           FOR THE LINEAR SYSTEM  A*X = B, CHANGES IN  A  AND  B       00000260
C           MAY CAUSE CHANGES  COND  TIMES AS LARGE IN  X .             00000270
C           IF COND+1 = COND, A IS SINGULAR TO WORKING PRECISION        00000280
C           COND = 1.0D+32  IF EXACT SINGULARITY IS DETECTED.           00000290
C                                                                       00000300
C        IPVT = THE PIVOT VECTOR.                                       00000310
C           IPVT(K) = THE INDEX OF THE K-TH PIVOT ROW                   00000320
C           IPVT(N) = (-1)**(NUMBER OF INTERCHANGES)                    00000330
C                                                                       00000340
C     WORK SPACE..  THE VECTOR  WORK  MUST BE DECLARED AND INCLUDED     00000350
C                   IN THE CALL.  ITS INPUT CONTENTS ARE IGNORED.       00000360
C                   ITS OUTPUT CONTENTS ARE USUALLY UNIMPORTANT.        00000370
C                                                                       00000380
C     THE DETERMINANT OF A CAN BE OBTAINED ON OUTPUT BY                 00000390
C        DET(A) = IPVT(N) * A(1,1) * A(2,2) * ... * A(N,N).             00000400
C                                                                       00000410
      DOUBLE PRECISION EK, T, ANORM, YNORM, ZNORM                       00000420
      INTEGER NM1, I, J, K, KP1, KB, KM1, M                             00000430
      DOUBLE PRECISION DABS, DSIGN                                      00000440
C                                                                       00000450
      IPVT(N) = 1                                                       00000460
      IF (N .EQ. 1) GO TO 80                                            00000470
      NM1 = N - 1                                                       00000480
C                                                                       00000490
C     COMPUTE 1-NORM OF A                                               00000500
C                                                                       00000510
      ANORM = 0.0D0                                                     00000520
      DO 10 J = 1, N                                                    00000530
         T = 0.0D0                                                      00000540
         DO 5 I = 1, N                                                  00000550
            T = T + DABS(A(I,J))                                        00000560
    5    CONTINUE                                                       00000570
         IF (T .GT. ANORM) ANORM = T                                    00000580
   10 CONTINUE                                                          00000590
```

```
C                                                                       00000600
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING                        00000610
C                                                                       00000620
      DO 35 K = 1,NM1                                                   00000630
         KP1= K+1                                                       00000640
C                                                                       00000650
C        FIND PIVOT                                                     00000660
C                                                                       00000670
         M = K                                                          00000680
         DO 15 I = KP1,N                                                00000690
            IF (DABS(A(I,K)) .GT. DABS(A(M,K))) M = I                   00000700
   15    CONTINUE                                                       00000710
         IPVT(K) = M                                                    00000720
         IF (M .NE. K) IPVT(N) = -IPVT(N)                               00000730
         T = A(M,K)                                                     00000740
         A(M,K) = A(K,K)                                                00000750
         A(K,K) = T                                                     00000760
C                                                                       00000770
C        SKIP STEP IF PIVOT IS ZERO                                     00000780
C                                                                       00000790
         IF (T .EQ. 0.0D0) GO TO 35                                     00000800
C                                                                       00000810
C        COMPUTE MULTIPLIERS                                            00000820
C                                                                       00000830
         DO 20 I = KP1,N                                                00000840
            A(I,K) = -A(I,K)/T                                          00000850
   20    CONTINUE                                                       00000860
C                                                                       00000870
C        INTERCHANGE AND ELIMINATE BY COLUMNS                           00000880
C                                                                       00000890
         DO 30 J = KP1,N                                                00000900
            T = A(M,J)                                                  00000910
            A(M,J) = A(K,J)                                             00000920
            A(K,J) = T                                                  00000930
            IF (T .EQ. 0.0D0) GO TO 30                                  00000940
            DO 25 I = KP1,N                                             00000950
               A(I,J) = A(I,J) + A(I,K)*T                               00000960
   25       CONTINUE                                                    00000970
   30    CONTINUE                                                       00000980
   35 CONTINUE                                                          00000990
C                                                                       00001000
C     COND = (1-NORM OF A)*(AN ESTIMATE OF 1-NORM OF A-INVERSE)         00001010
C     ESTIMATE OBTAINED BY ONE STEP OF INVERSE ITERATION FOR THE        00001020
C     SMALL SINGULAR VECTOR.  THIS INVOLVES SOLVING TWO SYSTEMS         00001030
C     OF EQUATIONS, (A-TRANSPOSE)*Y = E  AND  A*Z = Y  WHERE  E         00001040
C     IS A VECTOR OF +1 OR -1 CHOSEN TO CAUSE GROWTH IN Y.              00001050
C     ESTIMATE = (1-NORM OF Z)/(1-NORM OF Y)                            00001060
C                                                                       00001070
C     SOLVE (A-TRANSPOSE)*Y = E                                         00001080
C                                                                       00001090
      DO 50 K = 1, N                                                    00001100
         T = 0.0D0                                                      00001110
         IF (K .EQ. 1) GO TO 45                                         00001120
         KM1 = K-1                                                      00001130
         DO 40 I = 1, KM1                                               00001140
            T = T + A(I,K)*WORK(I)                                      00001150
   40    CONTINUE                                                       00001160
   45    EK = 1.0D0                                                     00001170
         IF (T .LT. 0.0D0) EK = -1.0D0                                  00001180
         IF (A(K,K) .EQ. 0.0D0) GO TO 90                                00001190
```

```
      WORK(K) = -(EK + T)/A(K,K)                                        00001200
   50 CONTINUE                                                          00001210
      DO 60 KB = 1, NM1                                                 00001220
         K = N - KB                                                     00001230
         T = 0.0D0                                                      00001240
         KP1 = K+1                                                      00001250
         DO 55 I = KP1, N                                               00001260
            T = T + A(I,K)*WORK(K)                                      00001270
   55    CONTINUE                                                       00001280
         WORK(K) = T                                                    00001290
         M = IPVT(K)                                                    00001300
         IF (M .EQ. K) GO TO 60                                         00001310
         T = WORK(M)                                                    00001320
         WORK(M) = WORK(K)                                              00001330
         WORK(K) = T                                                    00001340
   60 CONTINUE                                                          00001350
C                                                                       00001360
      YNORM = 0.0D0                                                     00001370
      DO 65 I = 1, N                                                    00001380
         YNORM = YNORM + DABS(WORK(I))                                  00001390
   65 CONTINUE                                                          00001400
C                                                                       00001410
C     SOLVE A*Z = Y                                                     00001420
C                                                                       00001430
      CALL SOLVE(NDIM, N, A, WORK, IPVT)                                00001440
C                                                                       00001450
      ZNORM = 0.0D0                                                     00001460
      DO 70 I = 1, N                                                    00001470
         ZNORM = ZNORM + DABS(WORK(I))                                  00001480
   70 CONTINUE                                                          00001490
C                                                                       00001500
C     ESTIMATE CONDITION                                                00001510
C                                                                       00001520
      COND = ANORM*ZNORM/YNORM                                          00001530
      IF (COND .LT. 1.0D0) COND = 1.0D0                                 00001540
      RETURN                                                            00001550
C                                                                       00001560
C     1-BY-1                                                            00001570
C                                                                       00001580
   80 COND = 1.0D0                                                      00001590
      IF (A(1,1) .NE. 0.0D0) RETURN                                     00001600
C                                                                       00001610
C     EXACT SINGULARITY                                                 00001620
C                                                                       00001630
   90 COND = 1.0D+32                                                    00001640
      RETURN                                                            00001650
      END                                                               00001660
      SUBROUTINE SOLVE(NDIM, N, A, B, IPVT)                             00001670
C                                                                       00001680
      INTEGER NDIM, N, IPVT(N)                                          00001690
      DOUBLE PRECISION A(NDIM,N),B(N)                                   00001700
C                                                                       00001710
C  SOLUTION OF LINEAR SYSTEM, A*X = B .                                 00001720
C  DO NOT USE IF DECOMP HAS DETECTED SINGULARITY.                       00001730
C                                                                       00001740
C  INPUT..                                                              00001750
C                                                                       00001760
C     NDIM = DECLARED ROW DIMENSION OF ARRAY CONTAINING A .             00001770
C                                                                       00001780
C     N = ORDER OF MATRIX.                                              00001790
```

```
C                                                                       00001800
C     A = TRIANGULARIZED MATRIX OBTAINED FROM DECOMP .                  00001810
C                                                                       00001820
C     B = RIGHT HAND SIDE VECTOR.                                       00001830
C                                                                       00001840
C     IPVT = PIVOT VECTOR OBTAINED FROM DECOMP .                        00001850
C                                                                       00001860
C   OUTPUT..                                                            00001870
C                                                                       00001880
C     B = SOLUTION VECTOR, X .                                          00001890
C                                                                       00001900
      INTEGER KB, KM1, NM1, KP1, I, K, M                                00001910
      DOUBLE PRECISION T                                                00001920
C                                                                       00001930
C     FORWARD ELIMINATION                                               00001940
C                                                                       00001950
      IF (N .EQ. 1) GO TO 50                                            00001960
      NM1 = N-1                                                         00001970
      DO 20 K = 1, NM1                                                  00001980
         KP1 = K+1                                                      00001990
         M = IPVT(K)                                                    00002000
         T = B(M)                                                       00002010
         B(M) = B(K)                                                    00002020
         B(K) = T                                                       00002030
         DO 10 I = KP1, N                                               00002040
             B(I) = B(I) + A(I,K)*T                                     00002050
   10    CONTINUE                                                       00002060
   20 CONTINUE                                                          00002070
C                                                                       00002080
C     BACK SUBSTITUTION                                                 00002090
C                                                                       00002100
      DO 40 KB = 1,NM1                                                  00002110
         KM1 = N-KB                                                     00002120
         K = KM1+1                                                      00002130
         B(K) = B(K)/A(K,K)                                             00002140
         T = -B(K)                                                      00002150
         DO 30 I = 1, KM1                                               00002160
             B(I) = B(I) + A(I,K)*T                                     00002170
   30    CONTINUE                                                       00002180
   40 CONTINUE                                                          00002190
   50 B(1) = B(1)/A(1,1)                                                00002200
      RETURN                                                            00002210
      END                                                               00002220
```

CAPITULO III

INTERPOLACION

La interpolación es uno de los problemas que con más frecuencia se enfrenta el ingeniero petrolero en la práctica. En este capítulo se describen algunas técnicas de interpolación.

Considérese un problema en el cual se tiene un conjunto de puntos en R^2 definidos por sus coordenadas

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

y donde

$$x_1 < x_2 < \ldots < x_n$$

Los valores y_i pertenecen a observaciones efectuadas de un cierto fenómeno a las condiciones x_i.

El problema de la interpolación lineal consiste en construir una función f tal que, el valor de f bajo las condiciones x_i sea igual a las observaciones y_i,

$$f(x_i) = y_i, \forall\, i = 1, \ldots, n.$$

Al mismo tiempo, la función f debe tomar "valores razonables" entre los puntos dato cuando se use con propósitos de interpolación. En la práctica, existen tantos criterios para definir "valor razonable" como problemas distintos. Por ejemplo, si las observaciones y_i provienen

de una función matemática "suave", la técnica de los splines cúbicos puede proporcionar buenos resultados. Para valores y_i experimentales y provenientes de observaciones más o menos imprecisas, no será necesario forzar a que la función ajustada pase por los puntos dato. Bastará, en este caso, con el ajuste de una función global para obtener resultados aceptables.

En términos matemáticos, el suavizamiento de una función está relacionado con el valor absoluto de la segunda derivada de la función

$$\left| f''(x) \right|,$$

y la simplicidad está asociada con el grado de la función.

El problema de la interpolación principia pues con la definición de " función razonable".

La mayoría de las funciones f(x) se construyen a partir de otras funciones más elementales. La combinación lineal de monomios (x^k) produce una función polinómica. En general, la forma de las funciones interpoladoras es

$$f(x) = \sum_{i=1}^{m} \lambda_i \, f_i(x)$$

Si las funciones f_i estuvieran definidas como

$$f_i = \text{Sen } ix \quad \text{ó} \quad f_i = \text{Cos } ix,$$

la función interpoladora f(x) representaría un polinomio trigonométrico.

Si f_i estuviese definida como

$$f_i = \frac{\alpha_0 + \alpha_1 x + \ldots + \alpha_i x^i}{\beta_0 + \beta_1 x + \ldots + \beta_n x^n},$$

f(x) representaría una función racional,

Estas son algunas de las funciones interpoladoras que a continuación se presentan.

Interpolación Polinomial.

Histórica y pragmáticamente, las funciones polinomiales constituyen el tipo de funciones más empleadas en procesos de interpolación. Una función polinomial de grado n,

$$p_n(x) = a_0 + a_1 x + \ldots + a_n x^n$$

es fácil de derivar e integrar, y sus coeficientes pueden estimarse sin dificultad. Además, según el teorema de Weierstran*, cualquier función contínua h(x) puede ser aproximada dentro de un cierto intervalo cerrado por una función polinomial única.

La existencia de los coeficientes a_i está asegurada ya que haciendo

$$y_i = p_n(x_i) = \sum_{k=0}^{m} a_k x^k \qquad \forall i=1, \ldots, n$$

*Ralston, A., 1965. A *First Course in Numerical Analysis*. McGraw-Hill.

se obtiene un sistema de ecuaciones lineales cuya matriz A será no lineal toda vez que se cumpla la condición de que

$$x_i \neq x_j, \quad \forall i \neq j .$$

En la práctica, sin embargo, se ha observado que la matriz A es sumamente mal condicionada. Un ejemplo clásico es el de valores x_i regularmente espaciados en el intervalo [0, 1], los cuales al ser sustituidos en las funciones $1, x, x^2, \ldots, x^m$, generan elementos positivos entre 0 y 1, produciéndose con ello una estrecha dependencia entre las columnas o entre los renglones de A.

Un método de interpolación más efectivo, que no presenta el problema de la dependencia, será descrito más adelante (Descomposición del Valor Singular).

Otra alternativa para la construcción de funciones interpoladoras es la de los polinomios de Lagrange.

$$\ell_j(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_{j-1})(x - x_{j+1}) \ldots (x - x_n)}{(x_j - x_0)(x_j - x_1) \ldots (x_j - x_{j-1})(x_j - x_{j+1}) \ldots (x_j - x_n)}$$

$$\ell_j(x) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

La función interpoladora de Lagrange de grado n se define a partir de la combinación lineal

$$f(x) = \sum_{j=1}^{m} y_j \ \ell_j(x)$$

En particular, nótese que el polinomio $y_j \ \ell_j(x)$ adquiere el valor y_j cuando x se sustituye por x_j, y toma el valor cero cuando $x_i \neq x_j$, $\forall i \neq j$.

Splines Cúbicos.

Una técnica de interpolación más sofisticada que las anteriores, es la de "Splines Cúbicos".

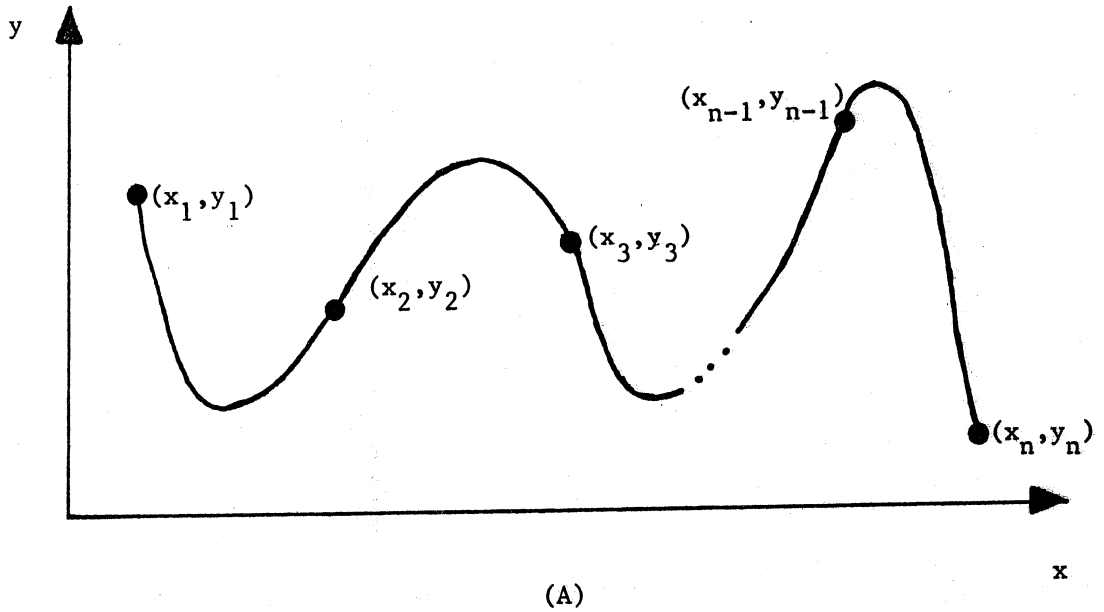
Las funciones cúbicas "spline" constituyen un desarrollo matemático reciente. Estas funciones se caracterizan por ser contínuas y por tener primera y segunda derivadas contínuas.

A diferencia de las técnicas de interpolación que emplean funciones polinómicas en las cuales a un conjunto de N datos se les ajusta un polinomio único de grado N - 1, en el método de las funciones cúbicas "spline" se ajustan N-1 polinomios de tercer grado, un polinomio por cada uno de los N - 1 intervalos definidos. Esta idea se ilustra gráficamente en la figura 8.

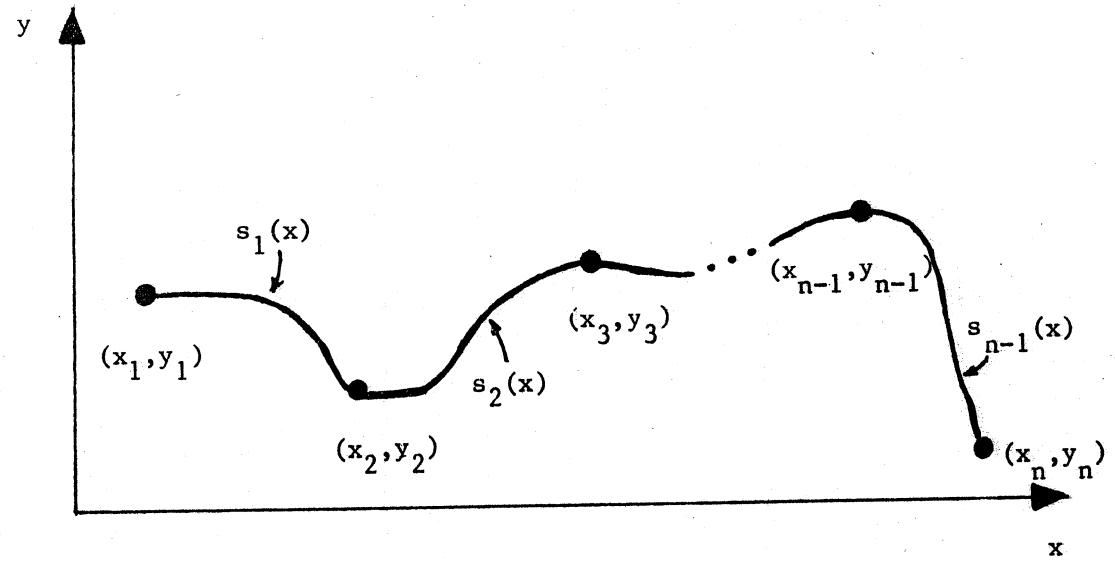
Gran parte de la teoría de los "splines" se inició con el teorema de Holladay.

Sean las abscisas $a = x_0 < x_1 < \ldots < x_N = b$

y las ordenadas $\{y_i\}$ (i = 0, 1, 2, ..., N) dados.



(A)



(B)

Figura 8. (A) Función polinómica única de grado n-1 ajustada a un conjunto de n datos. (B) n-1 funciones cúbicas spline ajustadas en n-1 intervalos.

De todas las funciones $f(x)$ con segunda derivada contínua en el intervalo $[a, b]$ tales que $f(x_i) = y_i$, $i = 0, 1, 2, \ldots, N$, la función spline $s(x)$ con segunda derivada igual a cero en los extremos del intervalo, $s''(a) = s''(b) = 0$, minimiza la integral

$$\int_a^b (f''(x))^2 \, dx$$

Prueba:

$$0 \leq \int_a^b (f''(x) - s''(x))^2 \, dx$$

$$0 \leq \int_a^b (f''(x))^2 \, dx - 2 \int_a^b (f''(x) - s''(x))\, s''(x) \, dx - \int_a^b (s''(x))^2 \, dx$$

$$0 \leq \int_a^b (f''(x))^2 \, dx - \int_a^b (s''(x))^2 \, dx + 2 \sum_{k=1}^{n-1} (f(x) - s(x))\, s'''(x) \Big|_{x_k}^{x_{k+1}}$$

$$- 2(f'(x) - s'(x))\, s''(x) \Big|_a^b$$

En el miembro del lado derecho, el tercer término desaparece debido a la condición de que

$f(x_i) = s(x_i) = y_i$, $(i = 0, 1, 2, \ldots, N)$; el cuarto término desaparece también según la condición de frontera

$$s''(a) = s''(b) = 0$$

Así

$$\int_a^b (s''(x))^2\,dx \leq \int_a^b (f''(x))^2\,dx$$

La función cúbica spline con condición $s''(a) = s''(b) = 0$ llamada también spline natural, es la función que posee la menor curvatura de todas las funciones que pueden interpolar entre puntos dato y cuya integral

$\int_a^b (f''(x))^2\,dx \leq \infty$ existe. En este sentido, la función cúbica spline es la función más suave que puede ajustarse a un conjunto de datos.

Con el objeto de entender mejor la construcción de estas funciones es conveniente contar el número de parámetros que intervienen. En los N - 1 intervalos, existen N - 1 secciones separadas de curvas cúbicas, cada una con cuatro parámetros, haciendo un total de 4N - 4 parámetros a determinar. El hecho de que la función s(x) sea contínua y tenga primera y segunda derivadas contínuas en cada uno de los N - 2 nodos interiores x_i, introduce 3(N - 2) condiciones en s. Luego, el hecho de que $s(x_i)$ sea igual a y_i en cada uno de los nodos impone N condiciones más en s(x), haciendo un total de 4N - 6 condiciones. Para generar un sistema compatible es necesario contar entonces con dos condiciones más, mismas que pueden ser sugeridas por las condiciones frontera, $s''(a) = s''(b) = 0$.

La construcción de una función spline es un proceso simple y numéricamente estable. Considérese el subintervalo (x_i, x_{i+1}) y sea

$$h_i = x_{i+1} - x_i, \quad w = (x - x_i)/h_i, \quad \bar{w} = 1 - w.$$

Ya que x fluctúa sobre este subintervalo, w variará de 0 a 1 y $\bar{w}$ de 1 a 0. Representemos la función spline en este subintervalo por medio de

$$s(x) = w\, y_{i+1} + \bar{w}\, y_i + h_i^2 \left[(w^3 - w)\, \sigma_{i+1} + (\bar{w}^3 - \bar{w})\sigma_i\right]$$

donde σ_i y σ_{i+1} son ciertas constantes por determinar. Los dos primeros términos en esta expresión representan una interpolación lineal, mientras que los términos entre paréntesis rectangulares representan una corrección cúbica, la cual proporcionará la suavidad en la solución. Nótese que el término corrector desaparece en los extremos del subintervalo, de tal manera que

$$s(x_i) = y_i \quad y \quad s(x_{i+1}) = y_{i+1}$$

Según ésto, la función s(x) interpola exactamente los datos, cualesquiera que sean los valores σ_i.

Diferenciando la función s(x) tres veces y usando la regla de la cadena, así como el hecho de que $w' = 1/h_i$ y $\bar{w}' = -1/h_i$, tenemos que

$$s'(x) = (y_{i+1} - y_i)/h_i + h_i \left[(3w^2 - 1)\, \sigma_{i+1} - (3\bar{w}^2 - 1)\, \sigma_i\right]$$

$$s''(x) = 6w\, \sigma_{i+1} + 6\bar{w}\, \sigma_i \ , \quad y$$

$$s'''(x) = 6(\sigma_{i+1} - \sigma_i)/h_i$$

Nótese que s"(x) es una función lineal la cual interpola entre

los valores $6\sigma_i$ y $6\sigma_{i+1}$.

Consecuentemente $\sigma_i = s''(x_i)/6$. Esto explica el significado de σ_i, pero no determina su valor. Nótese también que $s'''(x)$ es constante en cada subintervalo y que la cuarta derivada de $s(x)$ es igual a cero. Esto debe ser cierto, desde luego, ya que $s(x)$ es una función cúbica.

Evaluando $s'(x)$ en los puntos extremo del subintervalo tenemos

$$s'_+(x_i) = @_i - h_i\,(\sigma_{i+1} + 2\sigma_i)$$

$$s'_-(x_{i+1}) = @_i + h_i(2\sigma_{i+1} + \sigma_i)$$

donde

$$@_i = (y_{i+1} - y_i)/h_i$$

En la expresión anterior resulta necesario definir s'_+ y s'_-, ya que la fórmula de $s(x)$ se cumple únicamente en el intervalo $[x_i, x_{i+1}]$ de tal forma que las derivadas en los puntos extremo no están bien definidas. Con el objeto de obtener la continuidad deseada en $s'(x)$ se imponen las condiciones siguientes en los puntos interiores

$$s'_-(x_i) = s'_+(x_i), \qquad i = 2, \ldots, N - 1.$$

Aún cuando el valor de $s'_-(x_i)$ se calcule al considerar el subintervalo $[x_{i-1}, x_i]$ su fórmula puede ser obtenida al reemplazar i por i - 1 en $s'_-(x_{i+1})$, lo cual conduce a

$$@_{i-1} + h_{i-1}\,(2\sigma_i + \sigma_{i-1}) = @_i - h_i\,(\sigma_{i+1} + 2\sigma_i)$$

o bien a

$$h_{i-1}\,\sigma_{i-1} + 2(h_{i-1} + h_i)\,\sigma_i + h_i\,\sigma_{i+1} = \Omega_i - \Omega_{i-1}, \quad i = 2, \ldots, N-1$$

La expresión anterior representa un sistema de N - 2 ecuaciones lineales con N incógnitas, σ_i, i = 1, 2, ..., N. Bajo tales circunstancias, 2 condiciones adicionales deberán ser especificadas para poder obtener una solución única. Considerando un spline natural, el problema queda resuelto, ya que $s''(x_1) = s''(x_n) = 0$, implicará que

σ_1 y σ_n sean ambas idénticas a cero.

Un spline con estas condiciones frontera en las incógnitas σ's define el sistema de N - 2 ecuaciones lineales con N - 2 incógnitas siguiente:

$$\begin{bmatrix} 2(h_1+h_2) & h_2 & & & \\ h_2 & 2(h_2+h_3) & h_3 & & \\ & \ddots & \ddots & \ddots & \\ & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_3 \\ \vdots \\ \sigma_{n-2} \\ \sigma_{n-1} \end{bmatrix} = \begin{bmatrix} \Omega_2 - \Omega_1 \\ \Omega_3 - \Omega_2 \\ \vdots \\ \Omega_{n-2} - \Omega_{n-3} \\ \Omega_{n-1} - \Omega_{n-2} \end{bmatrix}$$

Este sistema puede resolverse fácilmente empleando el algoritmo de Thomas.

En ocasiones, resulta más conveniente calcular los coeficientes actuales del spline cúbico b_i, c_i, y d_i, i = 1, 2, ..., N - 1, para

cada intervalo $[x_i, x_{i+1}]$, siendo

$$s(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad x_i \leq x \leq x_{i+1},$$

a partir de las fórmulas siguientes:

$$b_i = (y_{i+1} - y_i)/h_i - h_i(\sigma_{i+1} + 2\sigma_i)$$

$$c_i = 3\sigma_i$$

$$d_i = (\sigma_{i+1} - \sigma_i)/h_i \quad \text{para} \quad i = 1, 2, \ldots, N-1$$

De esta manera se simplifican manipulaciones en s(x) tales como derivaciones e integraciones. Como comentario final, podemos agregar que éstos son precisamente los coeficientes que calculan la subrutina SPLINE que se enlista al final del capítulo.

Splines en Tres Dimensiones.

La aplicación de las funciones spline al caso de tres dimensiones resulta inmediata. Supóngase el siguiente problema en el cual se tienen M datos en el sentido del eje X por N datos en el sentido del eje Y, cada uno de ellos con un cierto valor correspondiente Z; las coordenadas del punto 1 siendo (X_1, Y_1) y las del punto MxN siendo (X_M, Y_N), tal como se ilustra en la figura 9.

Los MxN puntos definen una región rectangular de interpolación R. El problema es estimar el valor $\hat{Z}$ en cualquier punto (X*, Y*) en la región R.

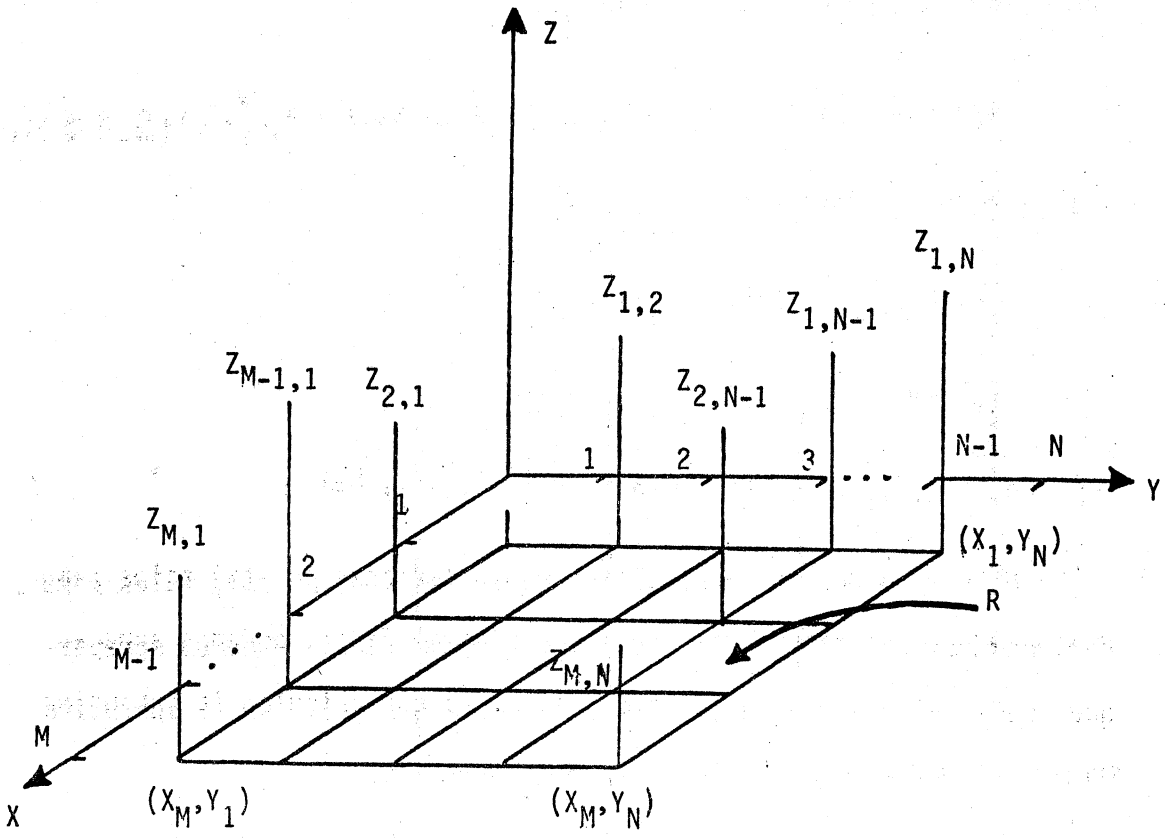


FIGURA 9. Región de aplicación de las funciones cúbicas spline.

El procedimiento es como sigue:

a).- Para cada Y_j, j=1, ..., N seleccionar las M parejas de puntos (X_i, Z_{ij}), i=1, ..., M correspondientes y ajustar por el método descrito en la sección anterior las funciones spline $S_j^{(k)}(x)$, k=1..., M-1 y j=1, ..., N generándose con ello un total de (M-1)x(N) funciones cúbicas.

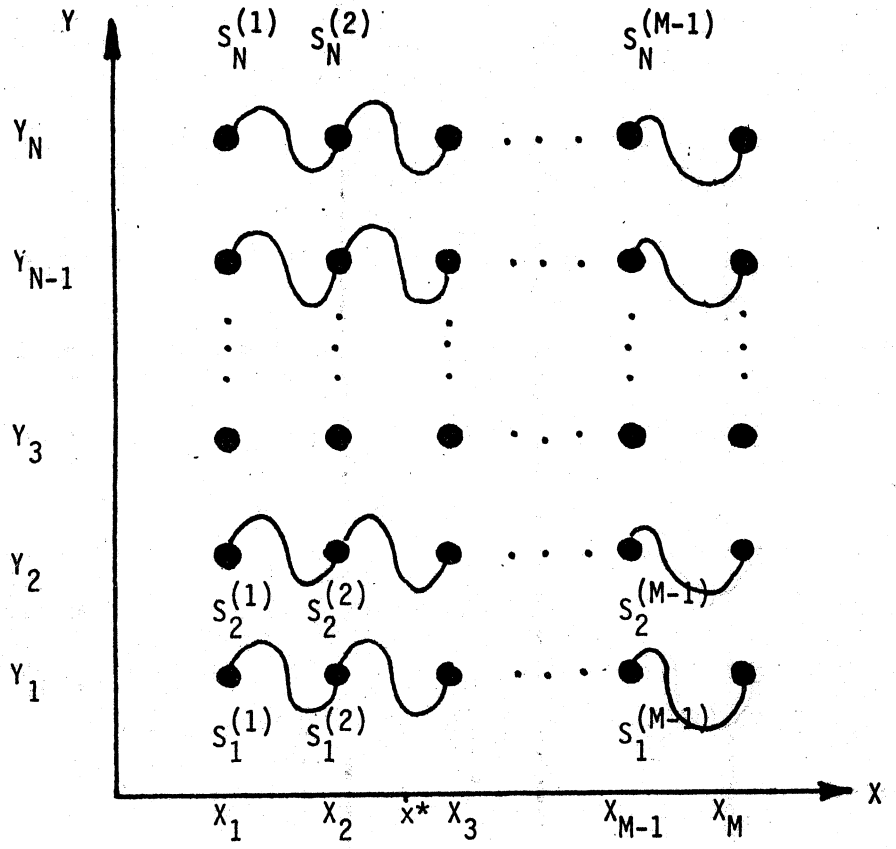


Figura 10. Secuencia de cálculo en la utilización de la técnica "cubic spline" en 3 dimensiones.

b).- Una primera aproximación $\tilde{Z}$ en el punto (X*, Y*) se obtiene seleccionando primeramente aquél conjunto de funciones

$$S_j^{(k)}(x),\ j = 1, \ldots, N$$

que interpolen en el intervalo $[X_k, X_{k+1}]$, y el cual contenga a la absci sa X", es decir, $X_k \leq X^* \leq X_{k+1}$; enseguida, calcular en las funciones seleccionadas

$$S_j^{(k)}(x),\ j=1, \ldots, N \text{ los valores } \tilde{Z}_{x^*,j},\ \tilde{Z}_{x^*,j} = S_j^{(k)}(X^*)$$

c).- Considerando la pareja de valores

$$(Y_j, \tilde{Z}_{x^*,j}),\ j = 1, \ldots N,$$

ajustar las funciones spline correspondientes y calcular finalmente en la función apropiada el valor $\hat{Z}$, esto es

$$\hat{Z} = S^{(\ell)}(Y^*),\quad Y_\ell \leq Y^* \leq Y_{\ell+1},\ \ell = 1, \ldots, N-1$$

Esta técnica proporcionaría el mismo resultado en $\hat{Z}$ si en lugar de seleccionar en el punto (a) las parejas (X_i, Z_{ij}) se hubiesen seleccionado las parejas (Y_j, Z_{ij}), para cada X_i, $i = 1, \ldots, M$.

Ajuste de Datos por Mínimos Cuadrados y Descomposición del Valor Singular.

La de mínimos cuadrados es una de las técnicas de interpolación más empleada en todas las ramas de la ingeniería.

Supongamos que cierto conjunto de datos (x_i, y_i), $i = 1, \ldots, m$ son dados, siendo x la variable independiente y y la variable dependiente. Ambas están relacionadas mediante una función desconocida

$$y = y(x)$$

Por tal motivo la variable y será aproximada mediante una combinación lineal de n funciones básicas ϕ_j,

$$y(x) \approx c_1\,\phi_1(x) + c_2\,\phi_2(x) + \ldots + c_n\,\phi_n(x).$$

A esta combinación lineal se le conoce como modelo matemático lineal. El

problema es seleccionar los n coeficientes $c_1, \ldots, c_n$, de tal forma que el modelo se "ajuste" a los datos de alguna manera preestablecida. El modelo es lineal ya que esa es la forma en que los coeficientes aparecen. Las funciones $\phi_i(x)$ pueden ser no-lineales, sin embargo.

El modelo lineal más común es el polinomial

$$y(x) \simeq c_1 + c_2x + c_3x^2 + \ldots + c_nx^{n-1}$$

donde cada función $\phi_j(x)$ es igual a x^{j-1}.

Para $\phi_j(x) = \text{sen}jx$ el modelo lineal correspondiente sería

$$y(x) = c_1\text{sen}x + c_2\,\text{sen}2x + \ldots + c_n\,\text{sen}\,nx$$

y para $\phi_j(x) = e^{\lambda_j x}$ sería

$$y(x) = c_1\,e^{\lambda_1 x} + c_2\,e^{\lambda_2 x} + \ldots + c_n\,e^{\lambda_n x}$$

En el último caso, si las λ_i tuviesen que ser determinadas, entonces el modelo sería no-lineal.

En adelante, únicamente consideraremos el caso donde el número de puntos dato es mayor o igual a n, el número de coeficientes, $m \geq n$. De todos los métodos utilizados en la evaluación de los coeficientes c_j, el de mínimos cuadrados es el más empleado.

Los coeficientes c_j se determinan de tal forma que el cuadrado de las diferencias entre $\sum_{j=1}^{n} c_j\,\phi_j(x_i)$, y y_i es minimizado, es decir

$$\underset{c_j}{\text{minimizar}} \sum_{i=1}^{m} r_i^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_j \phi_j(x_i) - y_i \right)^2$$

Efectuando las derivadas parciales con respecto a cada coeficiente c_k, e igualando a cero se tiene:

$$\frac{\partial}{\partial c_k} \left[\sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_j \phi_j(x_i) - y_i \right)^2 \right] = 0, \quad \forall\, k = 1, \ldots, n$$

ó

$$\sum_{i=1}^{m} 2\, \phi_k(x_i) \left(\sum_{j=1}^{n} c_j \phi_j(x_i) - y_i \right) = 0, \quad \forall\, k = 1, \ldots, n$$

sistema que puede ser expresado también como:

$$\sum_{i=1}^{m} \phi_k(x_i) \sum_{j=1}^{n} c_j \phi_j(x_i) = \sum_{i=1}^{m} \phi_k(x_i) y_i, \quad \forall\, k = 1, \ldots, n$$

ó

$$\sum_{j=1}^{n} c_j \sum_{i=1}^{m} \phi_k(x_i)\, \phi_j(x_i) = \sum_{i=1}^{m} \phi_k(x_i)\, y_i, \quad \forall\, k = 1, \ldots, n$$

y que en forma matricial puede ser escrito simplemente como $Pc = q$ donde P es una matriz de orden nxn cuyos elementos son

$\{P_{kj}\} = \sum_{i=1}^{m} \phi_k(x_i)\, \phi_j(x_i)$, q es el vector de n términos independientes $q_k = \sum_{i=1}^{m} \phi_k(x_i) y_i$, y c es el vector de coeficientes.

En el caso particular donde la función y(x) es aproximada por la ecuación de una línea recta $y(x) = c_1 + c_2 x$, la matriz P y el vector de términos independientes resultan iguales a

$$P = \begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \qquad q = \begin{bmatrix} \sum y_i \\ \sum y_i x_i \end{bmatrix}$$

En principio, el sistema $Pc = q$ puede resolverse empleando las subrutinas DECOMP y SOLVE. Sin embargo, dado que P es una matriz simétrica, el espacio de memoria puede reducirse a la mitad, y también dado que P es una matriz positiva definida, no es necesario buscar el elemento pivote, ya que los elementos de la diagonal principal serán todos diferentes de cero.

La práctica ha demostrado que la matriz P tiene, en muchas ocasiones, un número condicional (COND) sumamente grande. Esto produce que cualquier pequeño error cometido en los datos se traduzca en un error amplificado en el cálculo de los coeficientes. Igualmente, en el caso extremo de funciones base $\phi_j(x)$ dependientes, la matriz P será singular, y su número condicional podrá considerarse como infinito.

Consecuentemente, cualquier método que evite la generación de números condicionales grandes en la matriz P, puede considerarse como un buen detector de dependencia lineal entre las funciones base. El número condicional COND calculado por la subrutina DECOMP es de ayuda, pero él, por sí sólo, no proporcionará una solución al problema.

A continuación, describiremos una técnica que permite detectar y manejar el problema de la dependencia en las funciones base.

El método más confiable que permite calcular los coeficientes del

problema de mínimos cuadrados está basado en la factorización de una matriz. Dicho método es conocido con el nombre de "descomposición del valor singular" (DVS).

El método se inicia con la generación de la "matriz de diseño" del análisis estadístico de experimentos, o sea con una matriz A de orden mxn cuyos elementos vienen definidos por la expresión

$$a_{ij} = \phi_j(x_i)$$

Si y denota al vector de m elementos $\{y_i\}$, y c al vector de coeficientes, entonces la aproximación del modelo matemático lineal

$$\sum_{j=1}^{n} c_j \phi_j(x_i) \simeq y_i, \quad \forall i = 1, \ldots, m$$

puede escribirse en forma matricial como

$$Ac \simeq y,$$

El método DVS descompone la matriz A en tres matrices Σ, U y V. Σ es una matriz diagonal de orden mxn cuyos elementos no-negativos son los valores singulares de la matriz A. Las matrices U_{mxm} V_{nxn} son dos matrices ortogonales unitarias, las cuales son usadas en la transformación del sistema $Ac \simeq y$ en un sistema equivalente $\Sigma \bar{c} \simeq \bar{y}$. En otras palabras, si la matriz A puede expresarse como $U \Sigma V^t$, entonces la aproximación $Ac \simeq y$ puede substituirse por

$$U \Sigma V^t c \simeq y$$

Como U y V son dos matrices ortogonales, tales que $UU^t = I$ ó $U^{-1} = U^t$, la expresión anterior puede escribirse como

$$\Sigma V_c^t \simeq U_y^t \qquad (1)$$

o bien como $\Sigma \bar{c} \simeq \bar{y}$ donde $\bar{y} = U^t y$ y $\bar{c} = V^t c$

Ahora bien, ¿cómo se construyen las matrices Σ, U y V, tales que Σ sea una matriz diagonal y U y V sean matrices ortogonales?

Los valores singulares de la matriz A no son otros que las raíces cuadradas positivas de los eigenvalores de la matriz AA^t, que por cierto son iguales a los eigenvalores de la matriz A^tA. Los vectores singulares izquierdo y derecho son los eigenvectores de las matrices AA^t y A^tA respectivamente, y ellos constituyen las columnas de las matrices U y V.

Si las funciones base $\phi_j(x)$ fueran linealmente independientes en los puntos dato, entonces los valores singulares serían diferentes de cero.

El uso correcto del método de la descomposición del valor singular debe de considerar una tolerancia ζ la cual refleje la precisión de los datos originales. Cualquier valor singular σ_j mayor que ζ será aceptado y su correspondiente coeficiente $\bar{c}_j$ podrá ser calculado como $\bar{y}_j/\sigma_j$. Por otra parte, cualquier valor singular σ_i menor que ζ será considerado como cero y el correspondiente coeficiente $\bar{c}_j$ se igualará a cero. El cociente $\sigma_{max}/\sigma_{min}$ donde σ_{max} es el mayor valor singular y σ_{min} el menor valor singular, puede considerarse como otra forma alternativa de definir al número condicional de la matriz A

$$COND(A) = \sigma_{max}/\sigma_{min}$$

El método DVS se encuentra totalmente programado. La subrutina SVD (listada al final del capítulo) calcula, dada la matriz A_{mxn}, las matrices U, Σ y V. El siguiente programa ilustra el uso de esta subrutina.

Ejemplo del Uso de la Subrutina SVD.

(I) Creación de la matriz de diseño A.

```
      DØ  20  I  =  1, M
      T (I)      =  abscisa del punto dato i.
      Y (I)      =  ordenada del punto dato i.
      DØ 10 J    =  1, N
      A (I, J)   =  J-esima función base evaluada en el valor T(I)
   10 CØNTINUE
   20 CØNTINUE
```

(II) Llamada a la subrutina

```
      CALL SVD (MDIM,M,N,A,SIGMA,.TRUE.,U,.TRUE.,V,IERR,WØRK)
      IF (IERR.NE.0) WRITE (6,13)
   13 FØRMAT ('ERRØR ENCØNTRADO EN S V D ')
```

(III) Búsqueda del máximo valor singular y detección de valores singulares despreciables. Inicialización del vector de coeficientes.

```
      SIGMA1    =  0.
      DØ  30  J =  1, N
```

```
      IF(SIGMA(J).GT.SIGMA1) SIGMA1  =  SIGMA(J)
      C(J)                           =  0.
   30 CØNTINUE
```

IV) Se proporciona un error de tolerancia relativo RELERR. Si los datos tienen una exactitud de 3 dígitos, entonces RELERR = 10^{-3}, por ejemplo. Calcula la tolerancia en el error absoluto ς

```
      TAU  =  RELERR*SIGMA1
```

Calcula el vector de coeficientes C(J) de acuerdo al desarrollo en la expresión (1).

```
      DØ  60  J  =  1, N
      IF(SIGMA(J).LE.TAU) GØ TØ 60
      S          =  0.
      DØ  40  I  =  1, M
      S          =  S+U(I,J)*Y(I)      ... ȳ  =  U^t y
   40 CØNTINUE
      S          =  S/SIGMA(J)         ... c̄  =  Σ^-1 ȳ
      DØ  50  I  =  1, N
      C(I)       =  C(I)+S*V(I,J)      ... c  =  V c̄
   50 CØNTINUE
   60 CØNTINUE
```

En este programa M representa el número de datos y N el número de funciones base.

Ahora los coeficientes están listos para ser empleados.

```
WRITE(6,_ _ _)(C(J), J = 1, N)
```

V) La raíz cuadrada de la suma del cuadrado de los errores residuales puede obtenerse a partir de la expresión $(Ac-y)^t (Ac-y)$

```
      RSQ      = 0.
      DØ 90 I  = 1, M
      RI       = 0.
      DØ 80 J  = 1, N
      RI       = RI+C(J)*A(I,J)
   80 CONTINUE
      RSQ      = RSQ+(RI-Y(I))**2
   90 CONTINUE
      R        = SQRT(RSQ)
      :
      :
```

Como ilustración del método DVS y del ajuste por mínimos cuadrados veamos el ejemplo siguiente:

En un yacimiento de petróleo se desea inyectar gas nitrógeno como técnica de recuperación secundaria. En el cálculo del gasto y de la presión de inyección es determinante conocer el valor del factor de compresibilidad Z del nitrógeno. La gráfica de la figura 11 proporciona el valor de Z en función de la presión y de la temperatura. Sin embargo, con el objeto de predecir y controlar la operación a través de una computadora, resulta necesario contar con una expresión analítica de Z.

Por tal motivo, se leyeron de la gráfica los valores de Z para todas las temperaturas indicadas y para las presiones de 0 psia a 4000 psia a intervalos de 500 psia. Como expresión analítica se eligió la función cúbica

$$Z = C_1 + C_2 P' + C_3 T' + C_4 P'^2 + C_5 P'T' + C_6 T'^2 + C_7 P'^3 + C_8 P'^2T' + C_9 P'T'^2 + C_{10} T'^3$$

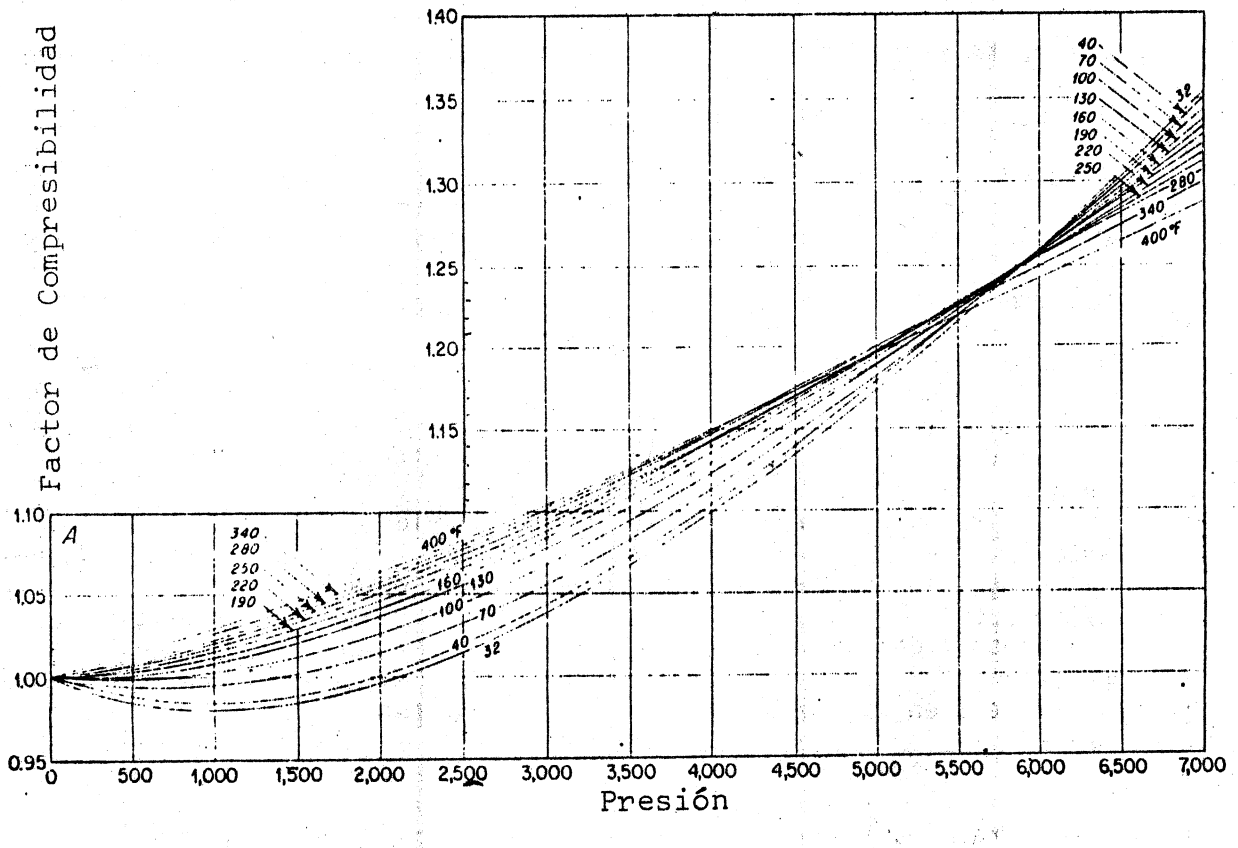


Figura 11. Factor de Compresibilidad "Z" del Nitrógeno.

donde P' y T' adoptarán diferentes valores según las alternativas:

$$\text{Alternativa A} \quad T' = T, \qquad P' = P$$

$$\text{Alternativa B} \quad T' = \frac{T-184}{10}, \qquad P' = \frac{P-2000}{100}$$

$$\text{Alternativa C} \quad T' = \frac{T-184}{100}, \qquad P' = \frac{P-2000}{1000}$$

donde P y T son los valores leídos de la gráfica, y las constantes 184 y 2000 son los valores leídos promedio de la temperatura y de la presión, respectivamente. Los divisores son factores de escala.

Considerando cada una de las tres alternativas, se efectuó el ajuste según mínimos cuadrados y según la técnica D V S. El resumen de los resultados obtenidos se muestra en la tabla 1. En la tabla, ρ es el coeficiente de correlación estadístico e indica la bondad en la predicción. Si $\rho = 1.00$ la predicción es "perfecta"; si $\rho = 0.00$ no hay tal predicción. $\sqrt{\Sigma r_i^2}$ es la raíz cuadrada de la suma de los errores al cuadrado e indica que tan cerca pasa la función ajustada a los puntos dato. Valores menores de $\sqrt{\Sigma r_i^2}$ denotan mayor cercanía. Como puede observarse, M-C permanece inalterable en los valores de ρ y $\sqrt{\Sigma r_i^2}$ ante las tres alternativas, lo cual es obvio ya que por construcción M-C. sólo minimiza las desviaciones y no le afectan los cambios de escala en las variables independientes. D V S es, excepto en la alternativa A, mejor que M-C. Además, D V S es sensible a los cambios de escala.

La figura 12 muestra dos gráficas donde el valor real del factor de compresibilidad Z, en las absisas, ha sido graficado contra el valor calculado Z*, en las ordenadas. La gráfica A corresponde al ajus

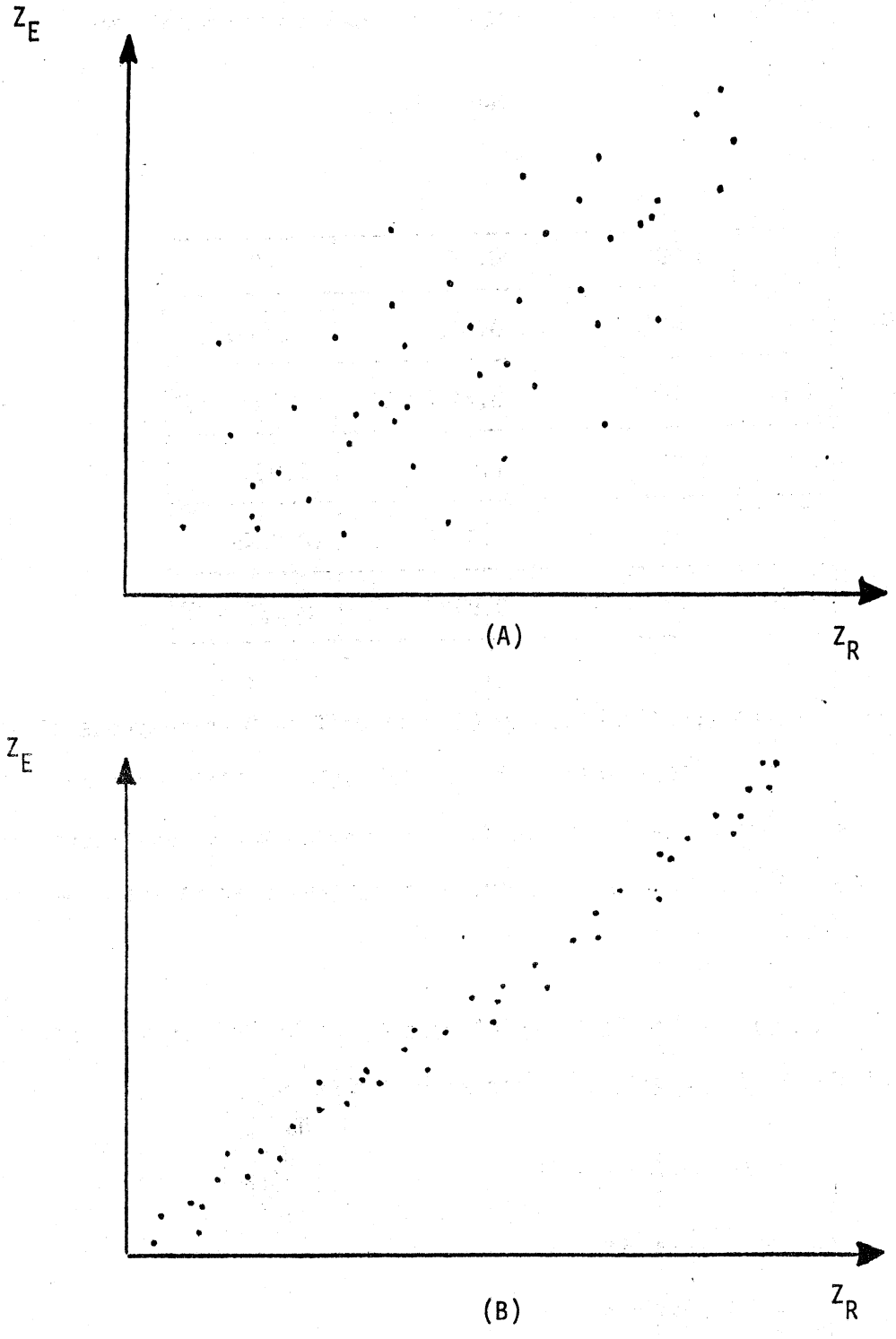


FIGURA 12. (A) Diagrama de correlación entre variable real (abscisa) y variable estimada (ordenada) por mínimos cuadrados. (B) Diagrama de correlación entre variable real (abscisa) y variable estimada (ordenada) por DVS.

Tabla 1. Resumen de resultados del ajuste MC VS DVS

		Alternativa A	Alternativa B	Alternativa C
M.C.	ρ	0.48	0.48	0.48
	$\sqrt{\Sigma ri^2}$	0.4361	0.4361	0.4361
	COND	0.4591×10^{23}	0.7448×10^{8}	0.1462×10^{3}
D.V.S.	ρ	0.79	1.00	1.00
	$\sqrt{\Sigma ri^2}$	6.8109	0.0349	0.0349
	COND	0.2789×10^{11}	0.8425×10^{4}	0.1220×10^{2}

te por M-C.(alternativas A, B y C) y la gráfica B corresponde al ajuste según D V S (alternativa C.). En la tabla 1, COND representa el número de condición de la matriz del sistema. En todas las alternativas el método D V S produce un número de condición siempre mucho menor al calculado por M-C.

Los valores de los coeficientes C_i de la función ajustada por la técnica D V S y bajo la alternativa C fueron:

$$C_1 = 0.10437360 \times 10^{1}$$

$$C_2 = 0.35185506 \times 10^{-1}$$

$$C_3 = 0.17852805 \times 10^{-1}$$

$$C_4 = 0.70853098 \times 10^{-2}$$

$$C_5 = 0.32667433 \times 10^{-2}$$

$C_6 = -0.54567901 \times 10^{-2}$

$C_7 = -0.34567901 \times 10^{-3}$

$C_8 = -0.29738655 \times 10^{-3}$

$C_9 = -0.79286510 \times 10^{-2}$

$C_{10} = 0.12825284 \times 10^{-2}$

La función ajustada proporciona un ajuste confiable para valores de T y P dentro del rango $32 \leq T \leq 400°F$ y $0 \leq P \leq 4000$ psia.

Kriging - Una Nueva Filosofía Dentro de los Procesos de Interpolación.

En contraste con los métodos bi-dimensionales de interpolación, interpolar en tres dimensiones resulta sumamente costoso en tiempo de cómputo y los requisitos exigidos a la información pueden, en muchas ocasiones, no ser satisfechos. Considérese el siguiente problema donde se desea obtener un plano configurado de las permeabilidades calculadas a través de pruebas de presión efectuadas en un cierto número de pozos (Figura 13). Como suele ocurrir en estos casos, lo primero que se observa es que la información no está distribuída regularmente en el espacio. Esto impediría la aplicación del método de las funciones bi-cúbicas o del método de mínimos cuadrados descrito en el Apéndice B. Otra característica importante de la información es que ésta se encuentra agrupada en ciertas porciones del área de estudio formando lo que se conoce como "nubes de información". El ajuste de una superficie polinomial produciría, bajo estas circunstancias, resultados incoherentes ya que la información más aislada estaría ejerciendo fuerte influencia sobre los resultados de

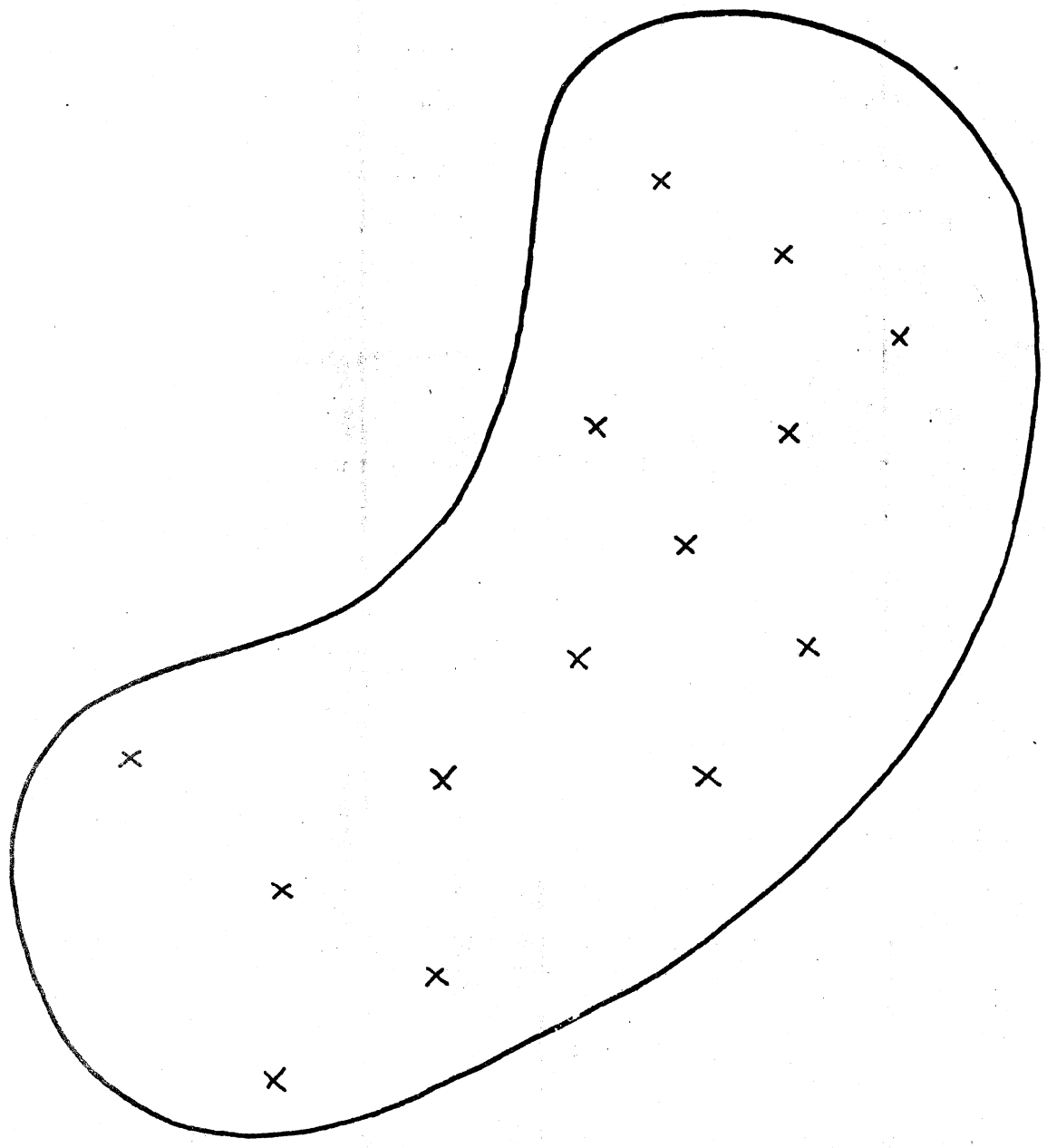


FIGURA 13. Representación gráfica de un campo petrolífero donde localizaciones de pozos perforados han sido indicadas con cruces. A través de pruebas de presión, permeabilidades han sido calculadas en cada pozo.

los coeficientes del polinomio.

Por tales motivos, se han creado otros métodos de interpolación, tales como el de ponderación con respecto al inverso de la distancia, ponderación con respecto al inverso del cuadrado de la distancia, etc., los cuales empleados conjuntamente con la técnica de búsqueda octal pueden producir resultados "aceptables". Todos estos métodos, sin embargo, no pueden evitar el error inherente de todo proceso de interpolación.

Existe un método que por diseño minimiza precisamente el error del proceso de interpolación y el cual se conoce como método Kriging de estimación. Tratar de explicar este método en toda su extensión tomaría por sí sólo un curso completo. Aquí nos limitaremos a resumir los principios básicos.

La piedra angular del método Kriging es una función denominada semi-variograma la cual expresa el grado de correlación espacial existente entre la información. Una vez estimada la función semi-variograma, es posible expresar el error, o mejor dicho, el valor promedio del error en términos del semi-variograma. Lo que resta es simplemente una aplicación del método de los multiplicadores de Lagrange sobre la expresión del error.

Si lo que se busca es estimar el valor de la variable desconocida Z empleando un estimador Z' definido como

$$Z' = \sum_{i=1}^{n} \lambda_i Z_i$$

donde los n valores Z_i son datos conocidos, entonces el método Kriging

proporcionará los n valores λ_i tales que el valor esperado del cuadrado de la diferencia entre Z y Z', esto es, $E[(Z-Z')^2]$ es el mínimo.

EJERCICIO # 4

Escriba un programa para la solución del siguiente problema. Un gas natural conteniendo 0.7% de nitrógeno fue sometido a un análisis de laboratorio donde el factor de compresibilidad Z fue medido. La tabla siguiente muestra los resultados obtenidos.

Presión	Temperatura			
(psia)	32°F	100°F	190°F	280°F
1.4	0.6885	0.8213	0.9097	0.9557
1.6	0.6593	0.8044	0.9009	0.9516
1.8	0.6433	0.7896	0.8943	0.9486
2.0	0.6369	0.7805	0.8899	0.9472
3.0	0.6981	0.7901	0.8932	0.9571
4.0	0.8103	0.8600	0.9356	0.9890
5.0	0.9333	0.9516	1.005	1.0384

Empleando el algoritmo de "bicubic-splines" descrito, interpole los valores de Z considerando valores de presión de 1.4 psia a 5.0 psia a intervalos de 0.1 psia, y considerando valores de temperatura de 32°F a 100°F a intervalos de 2°F y de 100°F a 280°F a intervalos de 10°F.

Comente la exactitud en la interpolación del método.

EJERCICIO # 5

En yacimientos productores de aceite cuando éste fluye por la tubería de producción del pozo se libera gas produciendo un flujo en dos fases. Poettmann y Carpenter han derivado una expresión analítica que permite calcular la caída de presión en tuberías verticales con flujo multifásico

$$\frac{\Delta p}{\Delta h} = \frac{1}{144}\left[\rho + \frac{f(qM)^2}{7.413 \times 10^{10} \rho\, d^5}\right]$$

donde

ρ es la densidad de la mezcla gas-aceite,

qM es el gasto de aceite por masa de la mezcla,

f es el factor de pérdidas de energía el cual depende del gasto qM,

d es el diámetro interno de la tubería, y

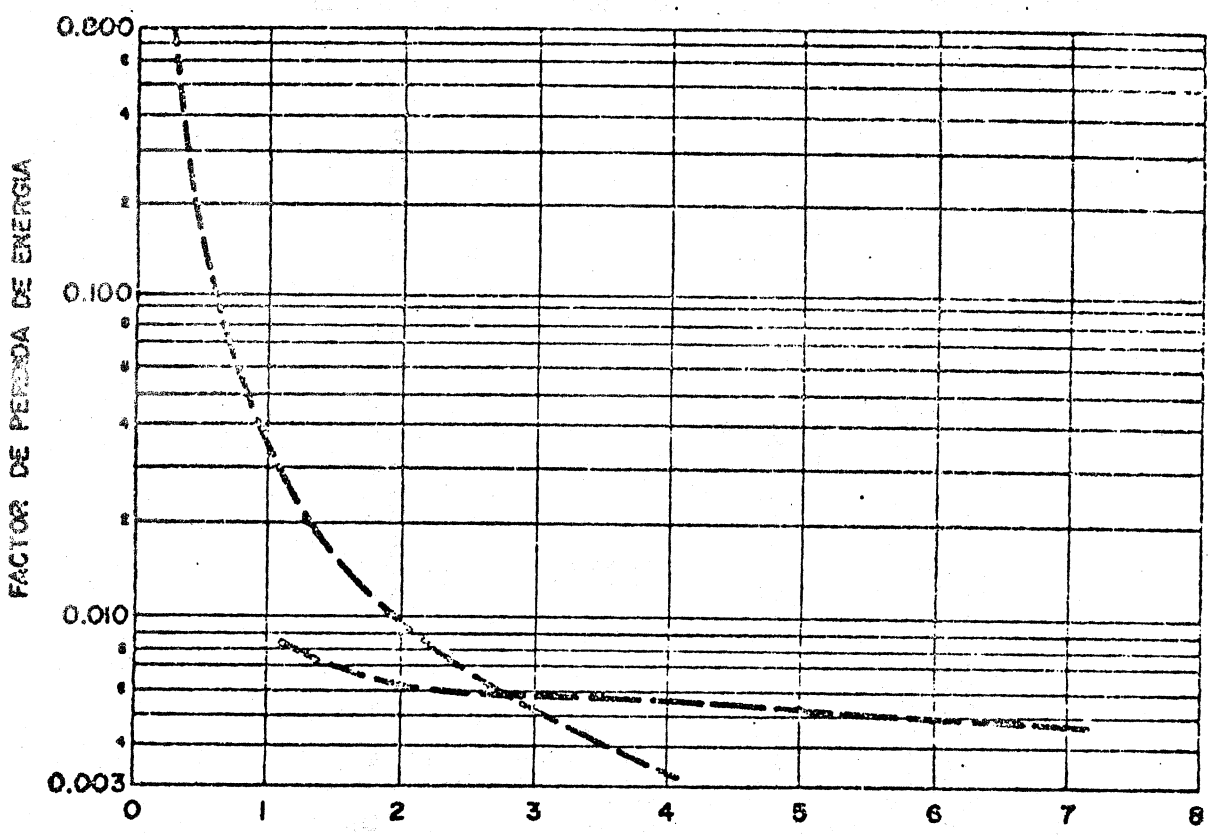
$\frac{\Delta p}{\Delta h}$ es el gradiente de presión.

Se desea encontrar, dado un cierto diámetro de tubería d, el gasto qM que producirá la mínima caída de presión en la tubería.

Del análisis sabemos que derivando $\Delta p/\Delta h$ con respecto al gasto qM e igualando a cero, obtendremos las condiciones buscadas en el gasto que producirán la caída mínima de presión. Sin embargo, el factor de pérdidas de energía f depende también del gasto qM, por lo que resulta necesario generar una función que exprese f en términos de qM. Dicha

función deberá ser substituída por f antes de proceder a la derivación.

La curva de abajo, obtenida experimentalmente por Baxendell, expresa el factor de pérdida de energía en términos del gasto qM/d. Empleando la técnica de mínimos cuadrados o la de descomposición del valor singular, ajuste una función a la curva mostrada. Substituya la función ajustada en la ecuación de Poettmann y Carpenter y encuentre una expresión del gasto qM en términos del diámetro d, para la cual la caída de presión en la tubería sea mínima.



```
      SUBROUTINE SPLINE (N, X, Y, B, C, D)                                  00000000
      INTEGER N                                                             00000010
      DOUBLE PRECISION X(N), Y(N), B(N), C(N), D(N)                         00000020
C                                                                           00000030
C  THE COEFFICIENTS B(I), C(I), AND D(I), I=1,2,...,N ARE COMPUTED          00000040
C  FOR A CUBIC INTERPOLATING SPLINE                                         00000050
C                                                                           00000060
C    S(X) = Y(I) + B(I)*(X-X(I)) + C(I)*(X-X(I))**2 + D(I)*(X-X(I))**3     00000070
C                                                                           00000080
C    FOR  X(I) .LE. X .LE. X(I+1)                                           00000090
C                                                                           00000100
C  INPUT..                                                                  00000110
C                                                                           00000120
C    N = THE NUMBER OF DATA POINTS OR KNOTS (N.GE.2)                        00000130
C    X = THE ABSCISSAS OF THE KNOTS IN STRICTLY INCREASING ORDER            00000140
C    Y = THE ORDINATES OF THE KNOTS                                         00000150
C                                                                           00000160
C  OUTPUT..                                                                 00000170
C                                                                           00000180
C    B, C, D  = ARRAYS OF SPLINE COEFFICIENTS AS DEFINED ABOVE.             00000190
C                                                                           00000200
C  USING  P  TO DENOTE DIFFERENTIATION,                                     00000210
C                                                                           00000220
C    Y(I) = S(X(I))                                                         00000230
C    B(I) = SP(X(I))                                                        00000240
C    C(I) = SPP(X(I))/2                                                     00000250
C    D(I) = SPPP(X(I))/6  (DERIVATIVE FROM THE RIGHT)                       00000260
C                                                                           00000270
C  THE ACCOMPANYING FUNCTION SUBPROGRAM  SEVAL  CAN BE USED                 00000280
C  TO EVALUATE THE SPLINE.                                                  00000290
C                                                                           00000300
C                                                                           00000310
      INTEGER NM1, IB, I                                                    00000320
      DOUBLE PRECISION T                                                    00000330
C                                                                           00000340
      NM1 = N-1                                                             00000350
      IF ( N .LT. 2 ) RETURN                                                00000360
      IF ( N .LT. 3 ) GO TO 50                                              00000370
C                                                                           00000380
C  SET UP TRIDIAGONAL SYSTEM                                                00000390
C                                                                           00000400
C  B = DIAGONAL, D = OFFDIAGONAL, C = RIGHT HAND SIDE.                      00000410
C                                                                           00000420
      D(1) = X(2) - X(1)                                                    00000430
      C(2) = (Y(2) - Y(1))/D(1)                                             00000440
      DO 10 I = 2, NM1                                                      00000450
         D(I) = X(I+1) - X(I)                                               00000460
         B(I) = 2.*(D(I-1) + D(I))                                          00000470
         C(I+1) = (Y(I+1) - Y(I))/D(I)                                      00000480
         C(I) = C(I+1) - C(I)                                               00000490
   10 CONTINUE                                                              00000500
C                                                                           00000510
C  END CONDITIONS.  THIRD DERIVATIVES AT  X(1)  AND  X(N)                   00000520
C  OBTAINED FROM DIVIDED DIFFERENCES                                        00000530
C                                                                           00000540
      B(1) = -D(1)                                                          00000550
      B(N) = -D(N-1)                                                        00000560
      C(1) = 0.                                                             00000570
      C(N) = 0.                                                             00000580
      IF ( N .EQ. 3 ) GO TO 15                                              00000590
      C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))                            00000600
      C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))                  00000610
      C(1) = C(1)*D(1)**2/(X(4)-X(1))                                       00000620
      C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))                                  00000630
C                                                                           00000640
C  FORWARD ELIMINATION                                                      00000650
C                                                                           00000660
   15 DO 20 I = 2, N                                                        00000670
         T = D(I-1)/B(I-1)                                                  00000680
         B(I) = B(I) - T*D(I-1)                                             00000690
         C(I) = C(I) - T*C(I-1)                                             00000700
   20 CONTINUE                                                              00000710
```

```
C                                                                       00000720
C  BACK SUBSTITUTION                                                    00000730
C                                                                       00000740
      C(N) = C(N)/B(N)                                                  00000750
      DO 30 IB = 1, NM1                                                 00000760
         I = N-IB                                                       00000770
         C(I) = (C(I) - D(I)*C(I+1))/B(I)                               00000780
   30 CONTINUE                                                          00000790
C                                                                       00000800
C  C(I) IS NOW THE SIGMA(I) OF THE TEXT                                 00000810
C                                                                       00000820
C  COMPUTE POLYNOMIAL COEFFICIENTS                                      00000830
C                                                                       00000840
      B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))         00000850
      DO 40 I = 1, NM1                                                  00000860
         B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))          00000870
         D(I) = (C(I+1) - C(I))/D(I)                                    00000880
         C(I) = 3.*C(I)                                                 00000890
   40 CONTINUE                                                          00000900
      C(N) = 3.*C(N)                                                    00000910
      D(N) = D(N-1)                                                     00000920
      RETURN                                                            00000930
C                                                                       00000940
   50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))                                    00000950
      C(1) = 0.                                                         00000960
      D(1) = 0.                                                         00000970
      RETURN                                                            00000980
      END                                                               00000990
```

```
      DOUBLE PRECISION FUNCTION SEVAL(N, U, X, Y, B, C, D)              00001000
      INTEGER N                                                         00001010
      DOUBLE PRECISION  U, X(N), Y(N), B(N), C(N), D(N)                 00001020
C                                                                       00001030
C  THIS SUBROUTINE EVALUATES THE CUBIC SPLINE FUNCTION                  00001040
C                                                                       00001050
C    SEVAL = Y(I) + B(I)*(U-X(I)) + C(I)*(U-X(I))**2 + D(I)*(U-X(I))**3 00001060
C                                                                       00001070
C    WHERE  X(I) .LT. U .LT. X(I+1), USING HORNER'S RULE                00001080
C                                                                       00001090
C  IF  U .LT. X(1) THEN  I = 1  IS USED.                                00001100
C  IF  U .GE. X(N) THEN  I = N  IS USED.                                00001110
C                                                                       00001120
C  INPUT..                                                              00001130
C                                                                       00001140
C    N = THE NUMBER OF DATA POINTS                                      00001150
C    U = THE ABSCISSA AT WHICH THE SPLINE IS TO BE EVALUATED            00001160
C    X,Y = THE ARRAYS OF DATA ABSCISSAS AND ORDINATES                   00001170
C    B,C,D = ARRAYS OF SPLINE COEFFICIENTS COMPUTED BY SPLINE           00001180
C                                                                       00001190
C  IF  U  IS NOT IN THE SAME INTERVAL AS THE PREVIOUS CALL, THEN A      00001200
C  BINARY SEARCH IS PERFORMED TO DETERMINE THE PROPER INTERVAL.         00001210
C                                                                       00001220
      INTEGER I, J, K                                                   00001230
      DOUBLE PRECISION DX                                               00001240
      DATA I/1/                                                         00001250
      IF ( I .GE. N ) I = 1                                             00001260
      IF ( U .LT. X(I) ) GO TO 10                                       00001270
      IF ( U .LE. X(I+1) ) GO TO 30                                     00001280
C                                                                       00001290
C  BINARY SEARCH                                                        00001300
C                                                                       00001310
   10 I = 1                                                             00001320
      J = N+1                                                           00001330
   20 K = (I+J)/2                                                       00001340
      IF ( U .LT. X(K) ) J = K                                          00001350
      IF ( U .GE. X(K) ) I = K                                          00001360
      IF ( J .GT. I+1 ) GO TO 20                                        00001370
C                                                                       00001380
C  EVALUATE SPLINE                                                      00001390
C                                                                       00001400
   30 DX = U - X(I)                                                     00001410
      SEVAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))                    00001420
      RETURN                                                            00001430
      END                                                               00001440
```

```
C                                                                       32440001
C     ------------------------------------------------------------------32440002
C                                                                       32440003
      SUBROUTINE SVD(NM,M,N,A,W,MATU,U,MATV,V,IERR,RV1)                 32440004
C                                                                       32440005
      INTEGER I,J,K,L,M,N,II,II,KK,K1,LL,L1,MN,NM,ITS,IERR             32440006
      REAL*8 A(NM,N),W(N),U(NM,N),V(NM,N),RV1(N)                        32440007
      REAL*8 C,F,G,H,S,X,Y,Z,EPS,SCALE,MACHEP                           32440008
      REAL*8 DSQRT,DMAX1,DABS,DSIGN                                     32440009
      LOGICAL MATU,MATV                                                 32440010
C                                                                       32440011
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE SVD,      32440012
C     NUM. MATH. 14, 403-420(1970) BY GOLUB AND REINSCH.                32440013
C     HANDBOOK FOR AUTO. COMP., VOL II-LINEAR ALGEBRA, 134-151(1971).   32440014
C                                                                       32440015
C     THIS SUBROUTINE DETERMINES THE SINGULAR VALUE DECOMPOSITION       32440016
C          T                                                            32440017
C     A=USV  OF A REAL M BY N RECTANGULAR MATRIX.  HOUSEHOLDER          32440018
C     BIDIAGONALIZATION AND A VARIANT OF THE QR ALGORITHM ARE USED.     32440019
C                                                                       32440020
C     ON INPUT:                                                         32440021
C                                                                       32440022
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL         32440023
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM          32440024
C          DIMENSION STATEMENT.  NOTE THAT NM MUST BE AT LEAST          32440025
C          AS LARGE AS THE MAXIMUM OF M AND N;                          32440026
C                                                                       32440027
C        M IS THE NUMBER OF ROWS OF A (AND U);                          32440028
C                                                                       32440029
C        N IS THE NUMBER OF COLUMNS OF A (AND U) AND THE ORDER OF V;    32440030
C                                                                       32440031
C        A CONTAINS THE RECTANGULAR INPUT MATRIX TO BE DECOMPOSED;      32440032
C                                                                       32440033
C        MATU SHOULD BE SET TO .TRUE. IF THE U MATRIX IN THE            32440034
C          DECOMPOSITION IS DESIRED, AND TO .FALSE. OTHERWISE;          32440035
C                                                                       32440036
C        MATV SHOULD BE SET TO .TRUE. IF THE V MATRIX IN THE            32440037
C          DECOMPOSITION IS DESIRED, AND TO .FALSE. OTHERWISE.          32440038
C                                                                       32440039
C     ON OUTPUT:                                                        32440040
C                                                                       32440041
C        A IS UNALTERED (UNLESS OVERWRITTEN BY U OR V);                 32440042
C                                                                       32440043
C        W CONTAINS THE N (NON-NEGATIVE) SINGULAR VALUES OF A (THE      32440044
C          DIAGONAL ELEMENTS OF S).  THEY ARE UNORDERED.  IF AN         32440045
C          ERROR EXIT IS MADE, THE SINGULAR VALUES SHOULD BE CORRECT    32440046
C          FOR INDICES IERR+1,IERR+2,...,N;                             32440047
C                                                                       32440048
C        U CONTAINS THE MATRIX U (ORTHOGONAL COLUMN VECTORS) OF THE     32440049
C          DECOMPOSITION IF MATU HAS BEEN SET TO .TRUE.  OTHERWISE      32440050
C          U IS USED AS A TEMPORARY ARRAY.  U MAY COINCIDE WITH A.      32440051
C          IF AN ERROR EXIT IS MADE, THE COLUMNS OF U CORRESPONDING     32440052
C          TO INDICES OF CORRECT SINGULAR VALUES SHOULD BE CORRECT;     32440053
C                                                                       32440054
C        V CONTAINS THE MATRIX V (ORTHOGONAL) OF THE DECOMPOSITION IF   32440055
C          MATV HAS BEEN SET TO .TRUE.  OTHERWISE V IS NOT REFERENCED.  32440056
C          V MAY ALSO COINCIDE WITH A IF U IS NOT NEEDED.  IF AN ERROR  32440057
C          EXIT IS MADE, THE COLUMNS OF V CORRESPONDING TO INDICES OF   32440058
C          CORRECT SINGULAR VALUES SHOULD BE CORRECT;                   32440059
C                                                                       32440060
```

```
C        IERR IS SET TO                                                 32440061
C          ZERO       FOR NORMAL RETURN,                               32440062
C          K          IF THE K-TH SINGULAR VALUE HAS NOT BEEN          32440063
C                     DETERMINED AFTER 30 ITERATIONS;                  32440064
C                                                                      32440065
C        RV1 IS A TEMPORARY STORAGE ARRAY.                             32440066
C                                                                      32440067
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,       32440068
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY        32440069
C                                                                      32440070
C     ------------------------------------------------------------------32440071
C                                                                      32440072
C     :::::::::: MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING    32440073
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.  32440074
C                MACHEP = 16.0D0**(-13) FOR LONG FORM ARITHMETIC       32440075
C                ON S360 ::::::::::                                    32440076
      DATA MACHEP/Z3410000000000000/                                   32440077
C                                                                      32440078
      IERR = 0                                                         32440079
C                                                                      32440080
      DO 100 I = 1, M                                                  32440081
C                                                                      32440082
         DO 100 J = 1, N                                               32440083
            U(I,J) = A(I,J)                                            32440084
  100 CONTINUE                                                         32440085
C     :::::::::: HOUSEHOLDER REDUCTION TO BIDIAGONAL FORM ::::::::::   32440086
      G = 0.0D0                                                        32440087
      SCALE = 0.0D0                                                    32440088
      X = 0.0D0                                                        32440089
C                                                                      32440090
      DO 300 I = 1, N                                                  32440091
         L = I + 1                                                     32440092
         RV1(I) = SCALE * G                                            32440093
         G = 0.0D0                                                     32440094
         S = 0.0D0                                                     32440095
         SCALE = 0.0D0                                                 32440096
         IF (I .GT. M) GO TO 210                                       32440097
C                                                                      32440098
         DO 120 K = I, M                                               32440099
  120    SCALE = SCALE + DABS(U(K,I))                                  32440100
C                                                                      32440101
         IF (SCALE .EQ. 0.0D0) GO TO 210                               32440102
C                                                                      32440103
         DO 130 K = I, M                                               32440104
            U(K,I) = U(K,I) / SCALE                                    32440105
            S = S + U(K,I)**2                                          32440106
  130    CONTINUE                                                      32440107
C                                                                      32440108
         F = U(I,I)                                                    32440109
         G = -DSIGN(DSQRT(S),F)                                        32440110
         H = F * G - S                                                 32440111
         U(I,I) = F - G                                                32440112
         IF (I .EQ. N) GO TO 190                                       32440113
C                                                                      32440114
         DO 150 J = L, N                                               32440115
            S = 0.0D0                                                  32440116
C                                                                      32440117
            DO 140 K = I, M                                            32440118
  140       S = S + U(K,I) * U(K,J)                                    32440119
C                                                                      32440120
```

```
                F = S / H                                                    32440121
C                                                                            32440122
                DO 150 K = I, M                                              32440123
                   U(K,J) = U(K,J) + F * U(K,I)                              32440124
  150           CONTINUE                                                     32440125
C                                                                            32440126
  190           DO 200 K = I, M                                              32440127
  200           U(K,I) = SCALE * U(K,I)                                      32440128
C                                                                            32440129
  210           W(I) = SCALE * G                                             32440130
                G = 0.0D0                                                    32440131
                S = 0.0D0                                                    32440132
                SCALE = 0.0D0                                                32440133
                IF (I .GT. M .OR. I .EQ. N) GO TO 290                        32440134
C                                                                            32440135
                DO 220 K = L, N                                              32440136
  220           SCALE = SCALE + DABS(U(I,K))                                 32440137
C                                                                            32440138
                IF (SCALE .EQ. 0.0D0) GO TO 290                              32440139
C                                                                            32440140
                DO 230 K = L, N                                              32440141
                   U(I,K) = U(I,K) / SCALE                                   32440142
                   S = S + U(I,K)**2                                         32440143
  230           CONTINUE                                                     32440144
C                                                                            32440145
                F = U(I,L)                                                   32440146
                G = -DSIGN(DSQRT(S),F)                                       32440147
                H = F * G - S                                                32440148
                U(I,L) = F - G                                               32440149
C                                                                            32440150
                DO 240 K = L, N                                              32440151
  240           RV1(K) = U(I,K) / H                                          32440152
C                                                                            32440153
                IF (I .EQ. M) GO TO 270                                      32440154
C                                                                            32440155
                DO 260 J = L, M                                              32440156
                   S = 0.0D0                                                 32440157
C                                                                            32440158
                   DO 250 K = L, N                                           32440159
  250              S = S + U(J,K) * U(I,K)                                   32440160
C                                                                            32440161
                   DO 260 K = L, N                                           32440162
                      U(J,K) = U(J,K) + S * RV1(K)                           32440163
  260           CONTINUE                                                     32440164
C                                                                            32440165
  270           DO 280 K = L, N                                              32440166
  280           U(I,K) = SCALE * U(I,K)                                      32440167
C                                                                            32440168
  290           X = DMAX1(X,DABS(W(I))+DABS(RV1(I)))                         32440169
  300 CONTINUE                                                               32440170
C     ::::::::::: ACCUMULATION OF RIGHT-HAND TRANSFORMATIONS :::::::::::     32440171
      IF (.NOT. MATV) GO TO 410                                              32440172
C     ::::::::::: FOR I=N STEP -1 UNTIL 1 DO -- :::::::::::                  32440173
      DO 400 II = 1, N                                                       32440174
         I = N + 1 - II                                                      32440175
         IF (I .EQ. N) GO TO 390                                             32440176
         IF (G .EQ. 0.0D0) GO TO 360                                         32440177
C                                                                            32440178
         DO 320 J = L, N                                                     32440179
C     ::::::::::: DOUBLE DIVISION AVOIDS POSSIBLE UNDERFLOW :::::::::::     32440180
```

```
  320     V(J,I) = (U(I,J) / U(I,L)) / G                                  32440181
C                                                                         32440182
        DO 350 J = L, N                                                   32440183
           S = 0.0D0                                                      32440184
C                                                                         32440185
           DO 340 K = L, N                                                32440186
  340      S = S + U(I,K) * V(K,J)                                        32440187
C                                                                         32440188
           DO 350 K = L, N                                                32440189
              V(K,J) = V(K,J) + S * V(K,I)                                32440190
  350   CONTINUE                                                          32440191
C                                                                         32440192
  360   DO 380 J = L, N                                                   32440193
           V(I,J) = 0.0D0                                                 32440194
           V(J,I) = 0.0D0                                                 32440195
  380   CONTINUE                                                          32440196
C                                                                         32440197
  390   V(I,I) = 1.0D0                                                    32440198
        G = RV1(I)                                                        32440199
        L = I                                                             32440200
  400 CONTINUE                                                            32440201
C     :::::::::: ACCUMULATION OF LEFT-HAND TRANSFORMATIONS ::::::::::     32440202
  410 IF (.NOT. MATU) GO TO 510                                           32440203
C     ::::::::::FOR I=MIN(M,N) STEP -1 UNTIL 1 DO -- ::::::::::         32440204
      MN = N                                                              32440205
      IF (M .LT. N) MN = M                                                32440206
C                                                                         32440207
      DO 500 II = 1, MN                                                   32440208
         I = MN + 1 - II                                                  32440209
         L = I + 1                                                        32440210
         G = W(I)                                                         32440211
         IF (I .EQ. N) GO TO 430                                          32440212
C                                                                         32440213
         DO 420 J = L, N                                                  32440214
  420    U(I,J) = 0.0D0                                                   32440215
C                                                                         32440216
  430    IF (G .EQ. 0.0D0) GO TO 475                                      32440217
         IF (I .EQ. MN) GO TO 460                                         32440218
C                                                                         32440219
         DO 450 J = L, N                                                  32440220
            S = 0.0D0                                                     32440221
C                                                                         32440222
            DO 440 K = L, M                                               32440223
  440       S = S + U(K,I) * U(K,J)                                       32440224
C     :::::::::: DOUBLE DIVISION AVOIDS POSSIBLE UNDERFLOW ::::::::::    32440225
            F = (S / U(I,I)) / G                                          32440226
C                                                                         32440227
            DO 450 K = I, M                                               32440228
               U(K,J) = U(K,J) + F * U(K,I)                               32440229
  450   CONTINUE                                                          32440230
C                                                                         32440231
  460   DO 470 J = I, M                                                   32440232
  470   U(J,I) = U(J,I) / G                                               32440233
C                                                                         32440234
        GO TO 490                                                         32440235
C                                                                         32440236
  475   DO 480 J = I, M                                                   32440237
  480   U(J,I) = 0.0D0                                                    32440238
C                                                                         32440239
  490   U(I,I) = U(I,I) + 1.0D0                                           32440240
```

```
  500 CONTINUE                                                          32440241
C     ********** DIAGONALIZATION OF THE BIDIAGONAL FORM **********     32440242
  510 EPS = MACHEP * X                                                  32440243
C     ********** FOR K=N STEP -1 UNTIL 1 DO -- **********              32440244
      DO 700 KK = 1, N                                                  32440245
         K1 = N - KK                                                    32440246
         K = K1 + 1                                                     32440247
         ITS = 0                                                        32440248
C     ********** TEST FOR SPLITTING.                                   32440249
C                FOR L=K STEP -1 UNTIL 1 DO -- **********              32440250
  520    DO 530 LL = 1, K                                               32440251
            L1 = K - LL                                                 32440252
            L = L1 + 1                                                  32440253
            IF (DABS(RV1(L)) .LE. EPS) GO TO 565                        32440254
C     ********** RV1(1) IS ALWAYS ZERO, SO THERE IS NO EXIT             32440255
C                THROUGH THE BOTTOM OF THE LOOP **********             32440256
            IF (DABS(W(L1)) .LE. EPS) GO TO 540                         32440257
  530    CONTINUE                                                       32440258
C     ********** CANCELLATION OF RV1(L) IF L GREATER THAN 1 **********  32440259
  540    C = 0.0D0                                                      32440260
         S = 1.0D0                                                      32440261
C                                                                       32440262
         DO 560 I = L, K                                                32440263
            F = S * RV1(I)                                              32440264
            RV1(I) = C * RV1(I)                                         32440265
            IF (DABS(F) .LE. EPS) GO TO 565                             32440266
            G = W(I)                                                    32440267
            H = DSQRT(F*F+G*G)                                          32440268
            W(I) = H                                                    32440269
            C = G / H                                                   32440270
            S = -F / H                                                  32440271
            IF (.NOT. MATU) GO TO 560                                   32440272
C                                                                       32440273
            DO 550 J = 1, M                                             32440274
               Y = U(J,L1)                                              32440275
               Z = U(J,I)                                               32440276
               U(J,L1) = Y * C + Z * S                                  32440277
               U(J,I) = -Y * S + Z * C                                  32440278
  550       CONTINUE                                                    32440279
C                                                                       32440280
  560    CONTINUE                                                       32440281
C     ********** TEST FOR CONVERGENCE **********                       32440282
  565    Z = W(K)                                                       32440283
         IF (L .EQ. K) GO TO 650                                        32440284
C     ********** SHIFT FROM BOTTOM 2 BY 2 MINOR **********             32440285
         IF (ITS .EQ. 30) GO TO 1000                                    32440286
         ITS = ITS + 1                                                  32440287
         X = W(L)                                                       32440288
         Y = W(K1)                                                      32440289
         G = RV1(K1)                                                    32440290
         H = RV1(K)                                                     32440291
         F = ((Y - Z) * (Y + Z) + (G - H) * (G + H)) / (2.0D0 * H * Y)  32440292
         G = DSQRT(F*F+1.0D0)                                           32440293
         F = ((X - Z) * (X + Z) + H * (Y / (F + DSIGN(G,F)) - H)) / X   32440294
C     ********** NEXT QR TRANSFORMATION **********                     32440295
         C = 1.0D0                                                      32440296
         S = 1.0D0                                                      32440297
C                                                                       32440298
         DO 600 I1 = L, K1                                              32440299
            I = I1 + 1                                                  32440300
```

```
                G = RV1(I)                                                      32440301
                Y = W(I)                                                        32440302
                H = S * G                                                       32440303
                G = C * G                                                       32440304
                Z = DSQRT(F*F+H*H)                                              32440305
                RV1(I1) = Z                                                     32440306
                C = F / Z                                                       32440307
                S = H / Z                                                       32440308
                F = X * C + G * S                                               32440309
                G = -X * S + G * C                                              32440310
                H = Y * S                                                       32440311
                Y = Y * C                                                       32440312
                IF (.NOT. MATV) GO TO 575                                       32440313
C                                                                               32440314
                DO 570 J = 1, N                                                 32440315
                   X = V(J,I1)                                                  32440316
                   Z = V(J,I)                                                   32440317
                   V(J,I1) = X * C + Z * S                                      32440318
                   V(J,I) = -X * S + Z * C                                      32440319
  570           CONTINUE                                                        32440320
C                                                                               32440321
  575           Z = DSQRT(F*F+H*H)                                              32440322
                W(I1) = Z                                                       32440323
C     :::::::::: ROTATION CAN BE ARBITRARY IF Z IS ZERO ::::::::::              32440324
                IF (Z .EQ. 0.0D0) GO TO 580                                     32440325
                C = F / Z                                                       32440326
                S = H / Z                                                       32440327
  580           F = C * G + S * Y                                               32440328
                X = -S * G + C * Y                                              32440329
                IF (.NOT. MATU) GO TO 600                                       32440330
C                                                                               32440331
                DO 590 J = 1, M                                                 32440332
                   Y = U(J,I1)                                                  32440333
                   Z = U(J,I)                                                   32440334
                   U(J,I1) = Y * C + Z * S                                      32440335
                   U(J,I) = -Y * S + Z * C                                      32440336
  590           CONTINUE                                                        32440337
C                                                                               32440338
  600       CONTINUE                                                            32440339
C                                                                               32440340
            RV1(L) = 0.0D0                                                      32440341
            RV1(K) = F                                                          32440342
            W(K) = X                                                            32440343
            GO TO 520                                                           32440344
C     :::::::::: CONVERGENCE ::::::::::                                         32440345
  650       IF (Z .GE. 0.0D0) GO TO 700                                         32440346
C     :::::::::: W(K) IS MADE NON-NEGATIVE ::::::::::                           32440347
            W(K) = -Z                                                           32440348
            IF (.NOT. MATV) GO TO 700                                           32440349
C                                                                               32440350
            DO 690 J = 1, N                                                     32440351
  690       V(J,K) = -V(J,K)                                                    32440352
C                                                                               32440353
  700 CONTINUE                                                                  32440354
C                                                                               32440355
      GO TO 1001                                                                32440356
C     :::::::::: SET ERROR -- NO CONVERGENCE TO A                               32440357
C                 SINGULAR VALUE AFTER 30 ITERATIONS ::::::::::                 32440358
 1000 IERR = K                                                                  32440359
 1001 RETURN                                                                    32440360
C     :::::::::: LAST CARD OF SVD ::::::::::                                    32440361
      END                                                                       32440362
```

CAPITULO IV

INTEGRACION NUMERICA

Existen diversos métodos numéricos que permiten efectuar la integración y/o la diferenciación de funciones. El método más apropiado para cada problema particular dependerá principalmente de la cantidad de información disponible sobre la función. Los problemas básicos que trataremos en este capítulo serán los de las funciones reales de una sola variable x, definida sobre un intervalo $[a, b]$. Estas funciones pueden dividirse en 4 categorías:

I) La función f(x) está bien definida y puede ser evaluada en cualquier valor real de la variable x sobre el intervalo $[a, b]$.

II) Los valores $f(x_i)$ están disponibles únicamente en ciertos puntos x_i del intervalo $[a, b]$.

III) La definición de la función puede ser extendida analíticamente al campo de los complejos.

IV) Una fórmula explícita de la función f(x) se encuentra disponible en una representación apropiada para su manipulación simbólica.

En todas aquellas funciones que caen dentro de las dos primeras categorías, la diferenciación numérica es básicamente más difícil que la

integración. Esto es debido a que la diferenciación tiende a amplificar cualquier error presente en la información y la integración tienda a suavizar o disminuir tal error. Así mismo, en aquellos casos donde los valores de la función f(x) son conocidos, o pueden ser calculados con cierta exactitud, y donde se requieren únicamente las primeras derivadas de la función, entonces los métodos basados en funciones splines o funciones polinómicas proporcionan, en general, resultados satisfactorios. Si además, derivadas de alto orden son deseadas, o si los valores de la función presentan "ruido", entonces los resultados pueden ser inexactos.

Dentro de la tercera categoría, podemos considerar aquellas funciones donde expresiones trigonométricas u otras funciones elementales forman parte de las funciones a integrar y/o diferenciar. Finalmente, si la extensión en el dominio de los complejos C es factible de hacerse, entonces la integración en C puede producir resultados satisfactorios.

En el lenguaje de los métodos numéricos, el término "regla" o "cuadratura" es empleado en la definición de aquellos algoritmos cuyas metodologías sirven para calcular aproximadamente ciertas integrales definidas. En este capítulo estudiaremos exclusivamente aquellas reglas o cuadraturas aplicables al tipo de funciones descritas en las categorías I y II.

Regla del Rectángulo y del Trapezoide.

Sea $[a, b]$ un intervalo finito para la variable x, el cual se ha

dividido en n subintervalos o páneles $[x_i, x_{i+1}]$, $i = 1, \ldots, n$.

Sean $x_1 = a$, $x_{n+1} = b$ donde

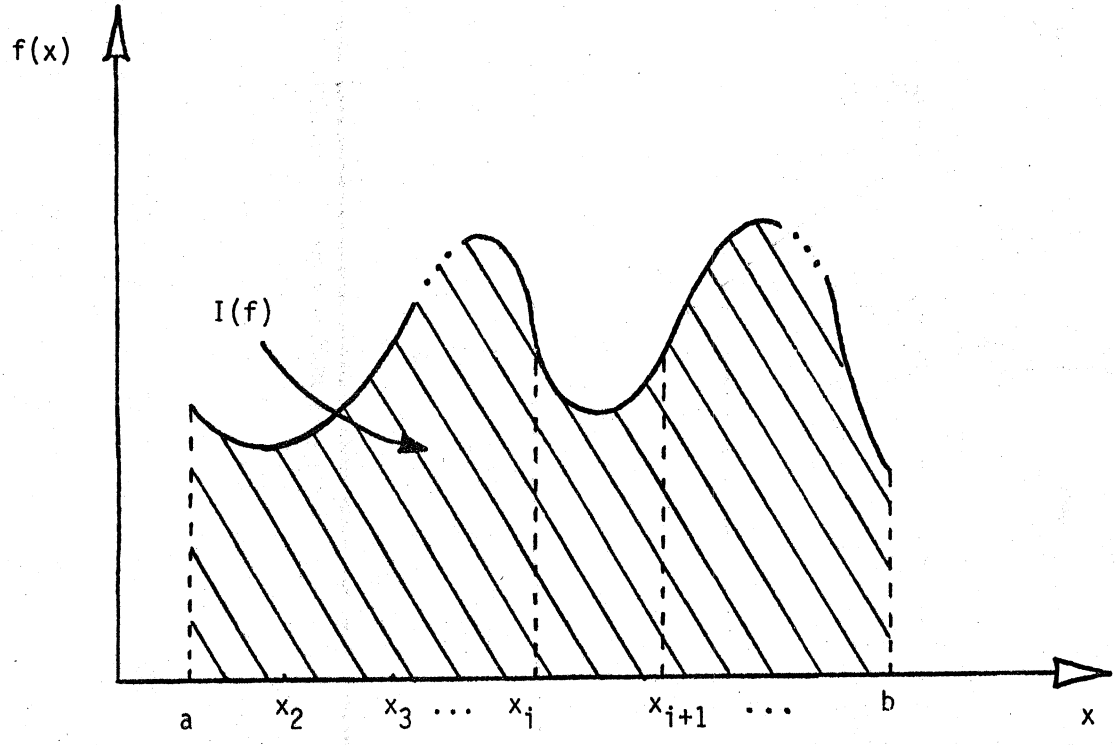
$$x_1 < x_2 < x_3 < x_4 < \ldots < x_{n+1},$$

siendo $h_i = x_{i+1} - x_i$, el ancho de los páneles, y sea $f(x)$ una función definida en el mismo intervalo $[a, b]$.

Una aproximación a la integral $I(f) = \int_a^b f(x)\,dx$ es deseada.

Tal aproximación podría derivarse a través de la suma de las integrales sobre páneles h_i, es decir

$$I(f) = \sum_{i=1}^{n} I_i, \quad \text{donde} \quad I_i = I_i(f) = \int_{x_i}^{x_{i+1}} f(x)\,dx.$$



Una regla de cuadratura simple es una fórmula que permite aproximar las integrales I_i. Una regla de cuadratura compuesta es una fórmula que aproxima la integral I(f) mediante una suma de reglas de cuadratura simples aplicadas a cada integral I_i.

Dos de las reglas de cuadratura simple más usadas son la regla del rectángulo y la del trapezoide. La regla del rectángulo utiliza en su estimación los valores de la función f(x) en los puntos medios de los páneles,

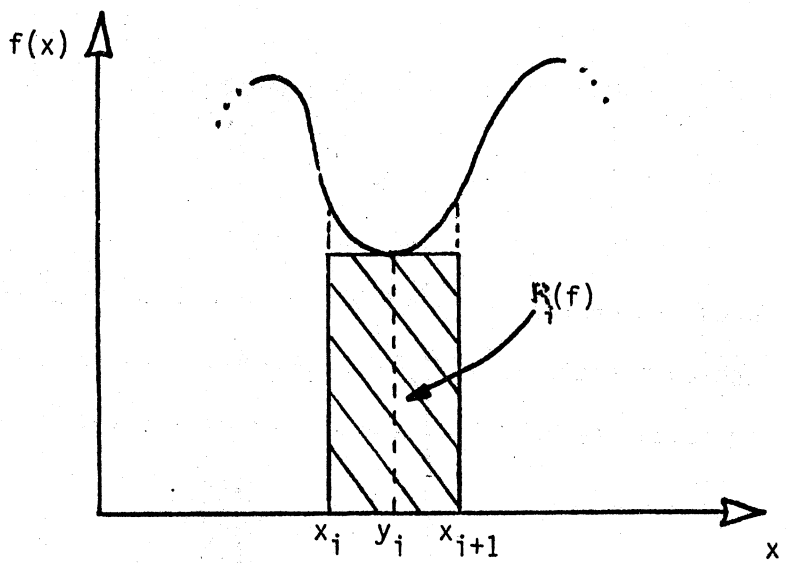
$$y_i = \frac{x_i + x_{i+1}}{2}, \quad i = 1, \ldots, n,$$

aproximando cada integral I_i mediante el área de un rectángulo de base h_i y de altura $f(y_i)$, es decir

$$I_i \simeq h_i \, f(y_i)$$

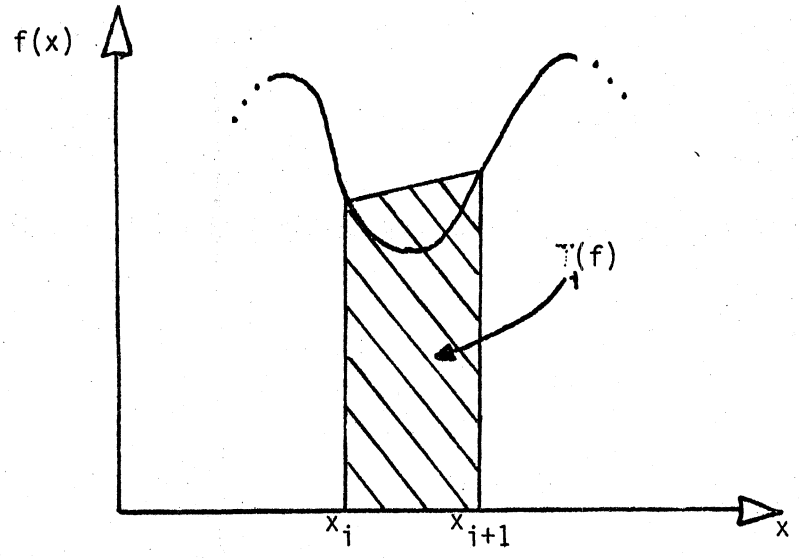
Según lo anterior, la regla compuesta del rectángulo resulta

$$R(f) = \sum_{i=1}^{n} h_i \, f(y_i)$$



La regla del trapezoide emplea los valores de la función en los extremos del pánel y aproxima cada integral a través del área de un trapezoide cuya base es h_i y cuya altura varía linealmente desde $f(x_i)$ a $f(x_{i+1})$, es decir

$$I_i(f) \simeq h_i \left(\frac{f(x_i) + f(x_{i+1})}{2}\right)$$



La regla compuesta del trapezoide resulta entonces

$$T(f) = \sum_{i=1}^{n} h_i \frac{f(x_i) + f(x_{i+1})}{2}$$

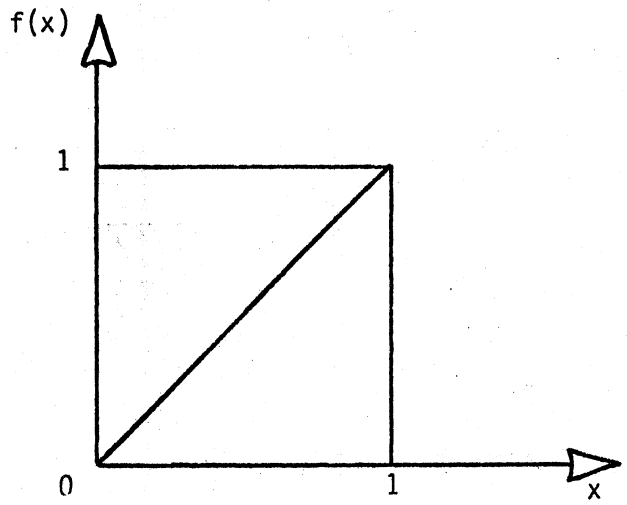
Puede mostrarse que si f(x) es una función contínua (o simplemente integrable según Riemann) en el intervalo $[a, b]$, y si $h = \max_i h_i$, entonces ambas reglas convergen al resultado exacto a medida que el ancho de

los subintervalos tiende a cero;

$$\lim_{h \to 0} R(f) = I(f) \quad y \quad \lim_{h \to 0} T(f) = I(f).$$

La pregunta obligada es ahora: ¿Qué tan rápido convergen estas reglas?. La regla del rectángulo está basada en la aproximación de una constante en cada subintervalo, mientras que la regla del trapezoide está basada en una interpolación lineal. Intuitivamente, uno podría esperar mayor exactitud en la regla trapezoidal. Sin embargo, observemos que pasa en el caso de la función

$f(x) = x$ definida sobre el intervalo $[0, 1]$.



La regla trapezoidal obviamente no da error, ya que la interpolación lineal concuerda con la función f(x). Sin embargo, la regla rectangular da un resultado también sin error a pesar del hecho de que la función interpoladora en el punto $x = \frac{1}{2}$ no concuerda con la función f(x). El error promedio es, en ambos casos, cero.

Es este ejemplo típico, o simplemente es suerte de la cuadratura del rectángulo? ¿Qué regla es en general más exacta?

Para dar respuesta a estas interrogantes, considere una función $f(x)$ y su expresión en series de Taylor con respecto al punto y_i localizado en el centro del panel $[x_i, x_{i+1}]$, esto es

$$f(x) = f(y_i) + (x-y_i)\, f'(y_i) + \frac{1}{2!}(x-y_i)^2 f''(y_i) + \frac{1}{3!}(x-y_i)^3 f'''(y_i)$$

$$+ \frac{1}{4!}(x-y_i)^4 f^{iv}(y_i) + \ldots$$

La primer suposición consiste en considerar que los términos " ... " son menos significativos que los términos mostrados explícitamente.

Observemos lo siguiente:

en la integral $\int_{x_i}^{x_{i+1}} f(x)\, dx = \int_{x_i}^{x_{i+1}} [f(y_i) + (x-y_i)\, f'(y_i) + \ldots]\, dx$

se obtiene que las potencias impares al ser integradas dancero:

$$\int_{x_i}^{x_{i+1}} (x-y_i)^p\, dx = \begin{cases} h_i, & \text{si} \quad p = 0 \\ 0, & \text{si} \quad p = 1 \\ \frac{h_i^3}{12}, & \text{si} \quad p = 2 \\ 0, & \text{si} \quad p = 3 \\ \frac{h_i^5}{80}, & \text{si} \quad p = 4 \end{cases}$$

Por lo tanto, la integral I_i puede ser expresada como

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \underbrace{h_i\, f(y_i)}_{R(f)} + \frac{1}{24}\, h_i^3\, f''(y_i) + \frac{1}{1920}\, h_i^5\, f'^{v}(y_i) + \ldots$$

donde, para valores de h_i pequeños, el error en la aproximación por la cuadratura del rectángulo (R) será $\frac{1}{24}\, h_i^3\, f'''(y_i)$, más algunos términos menos significativos.

Volviendo a la serie de Taylor y substituyendo en ella el valor $x=x_i$ primeramente, y posteriormente el valor $x = x_{i+1}$, tenemos:

$$f(x_i) = f(y_i) - \frac{1}{2} h_i\, f'(y_i) + \frac{1}{8} h_i^2\, f''(y_i) - \frac{1}{48} h_i^3\, f'''(y_i) + \frac{1}{384} h_i^4\, f'^{v}(y_i) + \ldots$$

y

$$f(x_{i+1}) = f(y_i) + \frac{1}{2} h_i f'(y_i) + \frac{1}{8} h_i^2\, f''(y_i) + \frac{1}{48} h_i^3\, f'''(y_i) + \frac{1}{384} h_i^4\, f'^{v}(y_i) + \ldots$$

de donde sumando las dos expresiones anteriores se obtiene que

$$\frac{f(x_i) + f(x_{i+1})}{2} = f(y_i) + \frac{1}{8} h_i^2\, f''(y_i) + \frac{1}{384} h_i^4\, f'^{v}(y_i) + \ldots$$

Combinando esto con la expresión anterior,

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = h_i\, f(y_i) + \frac{1}{24} h_i^3\, f''(y_i) + \frac{1}{1920} h_i^5\, f'^{v}(y_i) + \ldots$$

y considerando que

$$h_i\, f(y_i) = h_i \left(\frac{f(x_i) + f(x_{i+1})}{2}\right) - \frac{1}{8} h_i^3\, f''(y_i) - \frac{1}{384} h_i^5\, f'^{v}(y_i) + \ldots$$

se obtiene que

$$\int_{x_i}^{x_{i+1}} f(x)dx = \underbrace{h_i\left(\frac{f(x_i) + f(x_{i+1})}{2}\right)}_{T(f)} - \frac{1}{12} h_i^3 f''(y_i) - \frac{1}{480} h_i^5 f'^{v}(y_i) + \ldots$$

Para valores pequeños de h_i, el error en la aproximación de la regla trapezoidal (T) en un panel es:

$$-\frac{1}{12} h_i^3 f''(y_i),$$

más otros términos menos significativos. El error total de cada regla será la suma de los errores en todos los paneles, es decir, haciendo

$$E = \frac{1}{24} \sum_{i=1}^{n} h_i^3 f''(y_i)$$

$$F = \frac{1}{1920} \sum_{i=1}^{n} h_i^5 f'^{v}(y_i)$$

el error total resulta:

$$I(f) = R(f) + E + F + \ldots \qquad \text{(Rectángulo)}$$

$$I(f) = T(f) - 2E - 4F + \ldots \qquad \text{(Trapezoide)}$$

Para valores de $h_i < 1$ entonces $h_i^5 << h_i^3$ y por lo tanto $F << E$, provisto que f'^{v} no se comporte ruidosamente.

Conclusiones

a) R es cerca de 2 veces más exacta que T.

b) La diferencia entre los valores de R(f) y T(f) puede emplearse en la estimación del error en la integración

$$R(f) - T(f) = -3E - 5F \simeq -3E$$

c) Duplicando el número de páneles $h_i \to \frac{h_i}{2}$, se cuadruplica la exactitud ya que, por una parte, E se reduce por un factor de $\frac{1}{8}$, y por otra, el número de páneles se incrementa en 2, lo cual hace que el término total se vea reducido en $\frac{1}{4}$. Estrictamente hablando, el factor no es $\frac{1}{4}$, ya que la función f" no es generalmente constante y los términos de mayor orden ejercen alguna influencia. Desde un punto de vista práctico, en ambos casos, R y T, al doblar el número de páneles, cuadruplican su exactitud.

Esta técnica de doblar repetidamente el número de páneles y de calcular mediante R y T el error, puede programarse para producir un método, el cual automáticamente pueda determinar el número de páneles necesarios, de tal forma que la integral aproximada sea computada dentro de cierta tolerancia de error prescrita. Esta técnica será empleada en reglas de cuadratura más sofisticadas.

Cuadratura de Simpson.

En la sección anterior hemos definido las cuadraturas R y T

$$R(f) = \sum_{i=1}^{n} h_i \, f(y_i) \;, \quad y_i = \frac{x_i + x_{i+1}}{2}$$

$$T(f) = \sum_{i=1}^{n} \frac{h_i}{2} \left[f(x_{i+1}) + f(x_i) \right]$$

y además hemos probado que los errores de estas cuadraturas están dados por:

$$I(f) - R(f) = E + F + \ldots .$$

$$I(f) - T(f) = -2E - 4F + \ldots$$

donde

$$E = \frac{1}{24} \sum_{i=1}^{n} h_i^3 f''(y_i) \qquad y \qquad F = \frac{1}{1920} \sum_{i=1}^{n} h_i^3 f'^{v}(y_i).$$

Combinando estas dos cuadraturas podemos producir otra cuadratura donde el error no contenga los términos E.

Sea $S(f) = \frac{2}{3} R(f) + \frac{1}{3} T(f)$

ó $S(f) = \sum_{i=1}^{n} \frac{1}{6} h_i \left[f(x_i) + 4f(y_i) + f(x_{i+1})\right]$

Esta cuadratura recibe el nombre de Simpson y puede derivarse, como en los dos casos anteriores, integrando pieza a pieza una función, parabólica en este caso, la cual interpole los datos.

El error puede obtenerse directamente de las fórmulas de los errores en R y T.

$$I(f) - S(f) = \frac{2}{3} \left[I(f) - R(f)\right] + \frac{1}{3} \left[I(f) - T(f)\right]$$

$$= \frac{2}{3} (E + F + \ldots) + \frac{1}{3} (-2E - 4F + \ldots)$$

$$= (\frac{2}{3} - \frac{2}{3}) E + (\frac{2}{3} - \frac{4}{3}) F + \ldots$$

$$= -\frac{2}{3} F + \ldots,$$

$$= -\frac{1}{2880} \sum_{i=1}^{n} h_i^5 f'^{v}(y_i) + \ldots$$

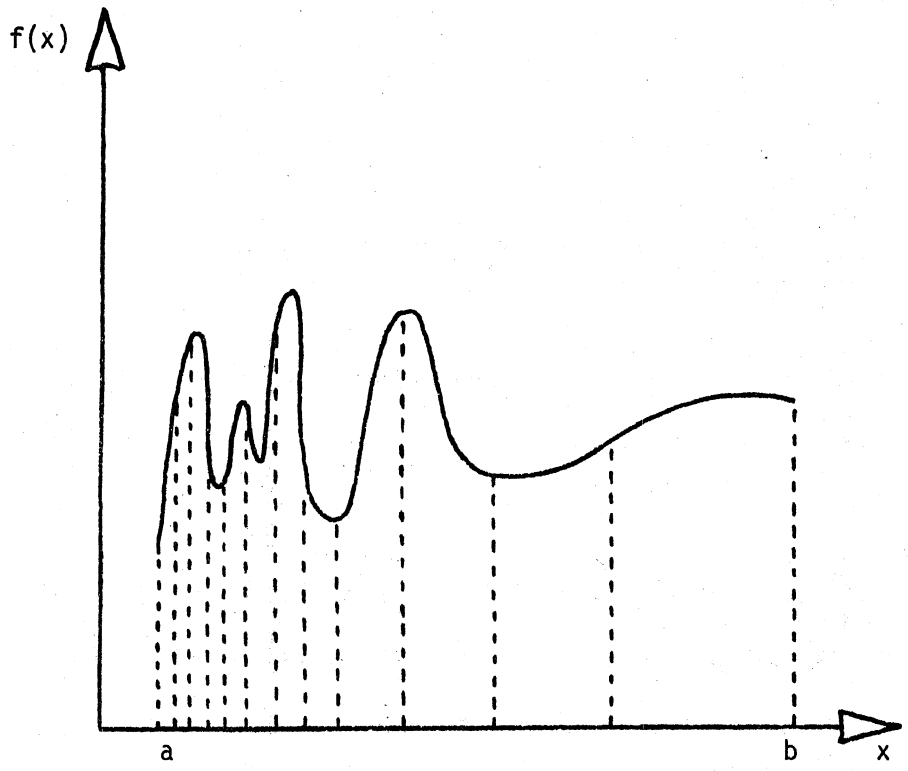
La cuadratura S(f) está basada en una interpolación de segundo grado y su error contiene derivadas de cuarto orden, por lo tanto, S(x) será una cuadratura exacta para funciónes cúbicas.

Cuando la longitud del intervalo es dividida, $h_i \to \frac{h_i}{2}$, el término h_i^5 decrece por un factor de $\frac{1}{32}$, al mismo tiempo el número de sumandos se duplica siendo la reducción del error total, un factor aproximadamente igual a $\frac{1}{16}$.

Esta técnica de combinar dos aproximaciones con errores similares para obtener una aproximación más exacta puede ser continuada. Por ejemplo, los valores de S(f) pueden combinarse con las cuadraturas R o T, para obtener errores del orden de h_i^7 y $f^{v'}(y_i)$. La regla general que describe este procedimiento se conoce con el nombre de cuadratura de Romberg.

Regla de Cuadratura Adaptada.

Esta regla es un algoritmo numérico el cual, empleando una o dos cuadraturas básicas, determina automáticamente el tamaño del subintervalo de tal manera que ciertos requisitos de exactitud sean satisfechos. El principio básico de esta regla consiste en seleccionar un tamaño relativamente grande de intervalos donde la función a integrar tenga un



comportamiento suave, y un intervalo pequeño donde la función cambie abrup tamente.

Esta regla requiere de la siguiente información:

(i) un intervalo finito $[a, b]$,

(ii) una función a integrar $f(x)$ definida en el intervalo $[a, b]$,

(iii) y un error de tolerancia ξ.

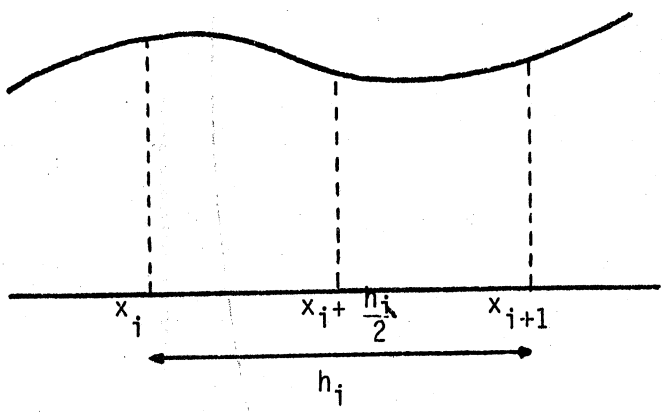
El algoritmo de cuadratura adaptada calculará una cantidad Q tal que

$$|Q - \int_a^b f(x)dx| \leq \xi$$

El intervalo de integración se divide en n subintervalos $[x_i, x_{i+1}]$, el número de subintervalos dependerá de ξ y de la función $f(x)$.

Sea, como antes, $h_i = x_{i+1} - x_i$, $(x_1 = a \quad y \quad x_{n+1} = b)$. El esquema o regla de la cuadratura adaptada aplica dos cuadraturas diferentes a cada subintervalo. Sean los dos resultados de estas cuadraturas P_i y Q_i. Por ejemplo, un esquema basado en la cuadratura de Simpson emplearía la fórmula de 2- páneles siguiente:

$$P_i = \frac{h_i}{6} \left[f(x_i) + 4f(x_i + \frac{h_i}{2}) + f(x_i + h_i)\right]$$



y un esquema de Simpson con 4-páneles emplearía

$$Q_i = \frac{h_i}{12} \left[f(x_i) + 4f(x_i + \frac{h_i}{4}) + 2f(x_i + \frac{h_i}{2}) + 4f(x_i + 3\frac{h_i}{4}) + f(x_i + h_i)\right],$$

P_i y Q_i son dos aproximaciones distintas de la integral

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = I_i .$$

El principio de la cuadratura adaptada consiste en comparar estas dos aproximaciones y obtener de la comparación una medida de la exactitud de cada aproximación.

Si el resultado es aceptable una de las dos se toma como el valor de la integral, de lo contrario se subdivide el intervalo en dos o más partes y el proceso se repite en los subintervalos más pequeños.

Es importante notar que la evaluación de la función f(x) se simplifica, ya que como puede observarse los términos P_i y Q_i tienen ciertas partes comunes. Q_i contiene únicamente dos términos que no están en P_i. En lo que resta, deberá entenderse que Q_i se obtiene aplicando la regla de P_i dos veces, una vez para cada mitad del intervalo. Esta idea facilitará el análisis de la regla adaptada.

Consideremos por un momento que la regla de P_i da la respuesta exacta cuando la función integrante es un polinomio de grado p-1, o equivalentemente, $f^{(p)}(x) = 0$. Expandiendo por series de Taylor con respecto al punto central de un subintervalo, puede demostrarse que

$$I_i - P_i = ch_i^{p+1} f^{(p)} (x_i + \frac{h_i}{2}) + \ldots$$

(ver esta misma expresión en la cuadratura del rectángulo pg. 98).

Como hemos supuesto que Q_i es la suma de dos P_i's calculados en subintervalos de longitud $\frac{h_i}{2}$, podemos concluir que

$$I_i - Q_i = c\left(\frac{h_i}{2}\right)^{p+1} \left[f^{(p)}\left(x_i + \frac{h_i}{4}\right) + f^{(p)}\left(x_i + 3\frac{h_i}{4}\right) \right] + \ldots$$

Pero dado que

$$f^{(p)}\left(x_i + \frac{h_i}{4}\right) + f^{(p)}\left(x_i + 3\frac{h_i}{4}\right) = 2f^{(p)}\left(x_i + \frac{h_i}{2}\right) + \ldots$$

los dos errores estarán relacionados por

$$I_i - Q_i = \frac{2}{2^{p+1}}(I_i - P_i) + \ldots = \frac{1}{2^p}(I_i - P_i) + \ldots$$

Esto indica que al partir el subintervalo, el error decrecerá cerca de 2^p veces.

Rearreglando términos:

$$I_i\left(1 - \frac{1}{2^p}\right) = Q_i - \frac{1}{2^p} P_i + \ldots$$

$$I_i(2^p - 1) = 2^p Q_i - P_i + \ldots$$

$$I_i - \frac{1}{2^p - 1} Q_i = \frac{2^p}{2^p - 1} Q_i - \frac{P_i}{2^p - 1} - \frac{1}{2^p - 1} Q_i + \ldots$$

$$I_i - \frac{Q_i}{2^p - 1} = Q_i - \frac{P_i}{2^p - 1} + \ldots$$

$$\frac{1}{2^p - 1}(P_i - Q_i) = Q_i - I_i + \ldots$$

Finalmente, el error en la expresión más exacta Q_i es cerca de $\frac{1}{2^p - 1}$ veces la diferencia entre las dos aproximaciones.

El objetivo esencial será pues, dividir cada subintervalo hasta que

la desigualdad

$$\frac{1}{2^p - 1} |P_i - Q_i| \leq \frac{h_i}{b - a} \xi \quad \text{sea satisfecha.}$$

ξ es la tolerancia requerida. La expresión $\frac{h_i}{b - a} \xi$ resultará evidente a continuación. Si todo el intervalo [a, b] fuera cubierto por n subintervalos, entonces el resultado sería:

$$Q = \sum_{i=1}^{N} Q_i$$

Tomando en cuenta lo anterior y despreciando los términos de alto orden tenemos:

$$|Q - \int_a^b f(x)dx| = |\sum_{i=1}^{n} Q_i - I_i|$$

$$\leq \sum_{i=1}^{n} |Q_i - I_i|$$

$$\leq \frac{1}{2^p - 1} \sum_{i=1}^{n} |P_i - Q_i|$$

$$\leq \frac{1}{2^p - 1} \frac{2^p - 1}{b - a} \xi \sum_{i=1}^{n} h_i = \xi$$

lo cual coincide con lo planteado arriba.

No se debe pasar por alto que en este análisis se requiere de la suposición de que $f^{(P)}(x)$ sea una función contínua y proporcional a $h_i^{p+1} f^{(p)}(x)$.

Hasta aquí y por simplicidad hemos empleado el criterio de error absoluto,

es decir

$$|Q - \int f| < \xi$$

En ocasiones el criterio de error relativo, el cual es independiente de la escala de f, es empleado

$$\frac{|Q - \int f|}{|\int f|} < \xi$$

Esto complica el análisis ya que el denominador $|\int f|$ puede ser cero. Además una buena aproximación del denominador solo se obtendría hasta el final del cálculo.

Otro criterio podría ser

$$\frac{|Q - \int f|}{\int |f|} < \xi$$

en cuyo caso el denominador no sería cero, al menos que la función f(x) fuese idéntica a cero y en cuyo caso el análisis sería igualmente complicado.

De cualquier manera el usuario de la cuadratura adaptada debe estar siempre conciente del tipo de error que se está empleando en la subrutina paquete que utilice para aproximar integrales.

Al final del capítulo se enlista la subrutina QUANC8 la cual aproxima integrales por el método de la cuadratura adaptada.

Cuadratura Spline.

La técnica de interpolación de las funciones cúbicas "spline" puede emplearse en la obtención de resultados interesantes en problemas de integración.

Para el intervalo $x_i \leq x \leq x_{i+1}$, sea

$$s(x) = f_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

la función spline que interpola f(x) en el intervalo $[x_i, x_{i+1}]$.

Si $a = x_1$ y $b = x_{n+1}$, la integral $\int_a^b f(x)dx$ puede ser aproximada por la integral $\int_a^b s(x)dx$,

$$\int_a^b s(x)dx = \sum_{i=1}^{n} (h_i f_i + \frac{1}{2} h_i^2 b_i + \frac{1}{3} h_i^3 c_i + \frac{1}{4} h_i^4 d_i)$$

donde $h_i = x_{i+1} - x_i$

Los coeficientes b_i, c_i y d_i pueden obtenerse por medio de la técnica cubic-spline ya descrita. (Dado que existen n+1 nodos, la subrutina SPLINE será llamada con N = n + 1).

La función s(x) puede representarse también como (ver capítulo III):

$$s(x) = wf_{i+1} + \bar{w}f_i + h_i^2 [(w^3 - w)\sigma_{i+1} + (\bar{w}^3 - \bar{w})\sigma_i]$$

donde

$$w = \frac{x - x_i}{h_i} = 1 - \bar{w}\ , \quad \sigma_i = \frac{s''(x_i)}{6} = \frac{c_i}{3}$$

y donde las incógnitas σ_i son calculadas a partir de un sistema de ecuaciones tridiagonal.

Para obtener la fórmula de la cuadratura observemos que:

$$\int_{x_i}^{x_{i+1}} s(x)dx = h_i \int_0^1 s(w)dw$$

Además, $\int_0^1 wdw = \frac{1}{2}$, y $\int_0^1 (w^3 - w)dw = \frac{-1}{4}$

Entonces, substituyendo en la primera integral se obtiene que

$$\int_{x_i}^{x_{i+1}} s(x)dx = h_i \left(\frac{f_i + f_{i+1}}{2} \right) - h_i^3 \left(\frac{\sigma_i + \sigma_{i+1}}{4} \right)$$

En otras palabras, la fórmula de la cuadratura spline es la misma que la fórmula de la cuadratura trapezoidal, más un "término de corrección", el cual contiene a la variable σ_i. Una fórmula más conveniente para el cálculo en el intervalo total sería

$$\int_a^b s(x)dx = \sum_{i=1}^{n} h_i \frac{f_i + f_{i+1}}{2} - h_i^3 \frac{c_i + c_{i+1}}{12}\ .$$

En este caso, únicamente los datos (x, f) y el arreglo de segundas derivadas c_i, calculado por SPLINE, serían requeridos.

Con respecto a la exactitud podemos observar que

$$h_i^3 \frac{c_i + c_{i+1}}{12} = h_i^3 \frac{s''(x_i) + s''(x_{i+1})}{24}$$

$$\simeq \frac{h_i^3}{12} f''(y_i) \quad \text{donde} \quad y_i = \frac{x_i + x_{i+1}}{2}$$

Por lo tanto e interesantemente, el término de corrección provisto por la cuadratura SPLINE aproxima al término error de la cuadratura trapezoidal. En la literatura no se han reportado resultados prácticos donde se haya aplicado la cuadratura SPLINE; sin embargo, por los resultados mostrados aquí, ésta parece ser de gran utilidad.

Por último, debemos observar que la cuadratura spline no presenta la forma convencional de otras cuadraturas, es decir,

$$\int_a^b f(x)dx \simeq \sum_{i=1}^{n+1} \alpha_i f(x_i).$$

En la cuadratura spline cada coeficiente α_i dependerá de los valores $f(x_i)$ en forma más complicada.

Aproximación de Integrales por el Método de Monte Carlo.

Supongamos que se desea integrar la función g(x) en el intervalo $a \leq x \leq b$,

$$I = \int_a^b g(x)dx .$$

La metodología a emplear consiste en seleccionar una función de densidad arbitraria $P_V(x)$ de una variable aleatoria V definida en el intervalo $[a, b]$, es decir

$$P_V(x) \geq 0 \ , \quad \forall \ x \in [a, b]$$

$$\int_a^b P_V(x)dx = 1$$

Posteriormente, además de la variable aleatoria V, se requiere definir la siguiente función aleatoria

$$H = \frac{g(v)}{P_V(v)}$$

Ahora bien, la definición de valor esperado $E[f_V(v)] = \int_a^b f(v)P_V(v)dv$ puede aplicarse directamente en la función H, esto es

$$E[H] = \int_a^b \frac{g(v)}{P_V(v)} P_V(v)dv = I$$

En vez de una variable aleatoria H, consideremos ahora n variables aleatorias independientes e idénticamente distribuidas

$H_1, H_2, H_3, \ldots, H_n$ tales que

$$E[H_i] = I, \quad \forall \ i = 1, \ldots, n$$

$$\text{Var}[H_i] = \text{Var}[H], \quad \forall \ i = 1, \ldots, n$$

Si formamos una nueva variable aleatoria H*, definida como

$$H^* = H_1 + \ldots + H_n, \text{ entonces}$$

$$E[H^*] = nI$$

$$\text{Var}[H^*] = n\text{Var}[H]$$

Según el teorema del Límite Central, la distribución de la nueva variable aleatoria H* tenderá a la distribución normal, acercándose a $N(\mu,\sigma)$ a medida que $n \to \infty$, es decir,

$$Pr\{|H^* - nI| < 3\sqrt{\text{Var } H^*}\} \simeq 0.997$$

ó $$Pr\{|\sum_{i=1}^{n} H_i - nI| < 3\sqrt{n\text{Var } H}\} \simeq 0.997.$$

Dividiendo entre n

$$Pr\{|\frac{1}{n}\sum_{i=1}^{n} H_i - I| < 3\sqrt{\frac{\text{Var } H}{n}}\} \simeq 0.997,$$

lo que también se interpreta como una variable aleatoria $\frac{1}{n}\sum_{i=1}^{n} H_i$ cercana al valor I con probabilidad 0.997 y con un error menor a $3\sqrt{\frac{\text{Var } H}{n}}$.

A medida que n crece se observa que

$$\frac{1}{n}\sum_{i=1}^{n} H_i = \frac{1}{n}\sum_{i=1}^{n} \frac{g(v_i)}{P_V(v_i)} \simeq I.$$

Ahora bien, según hemos visto en el cálculo aproximado de I, pue-

de emplearse la distribución de cualquier variable aleatoria V. En cualquier caso, el estimador será insesgado ya que

$$E[H] = E\left[\frac{g(v)}{P_V(v)}\right] = I$$

y su varianza será Var[H]. La pregunta que ahora se presenta es, ¿cómo reducir la varianza del error?, esto es,

$$\min \text{Var}[H] = \min(E[H^2] - I^2) = \min\left(\int_a^b \frac{g^2(x)}{P_V(x)}\,dx - I^2\right).$$

A continuación mostraremos que Var[H] se minimiza si se elige una función de densidad $P_V(x)$ proporcional a $|g(x)|$.

Haciendo $u = \dfrac{g(x)}{\sqrt{P_V(x)}}$ y $v = \sqrt{P_V(x)}$ en la desigualdad de Schwarz

$$\left|\int_a^b |u(x)\, v(x)|\,dx\right|^2 \leq \int_a^b u^2(x)dx \int_a^b v^2(x)dx$$

se obtiene:

$$\left[\int_a^b |g(x)|dx\right]^2 \leq \int_a^b \frac{g^2(x)}{P_V(x)}\,dx \int_a^b P_V(x)dx = \int_a^b \frac{g^2(x)}{P_V(x)}\,dx \qquad (1)$$

y de acuerdo a la definición de la varianza se tiene entonces que

$$\text{Var}[H] \geq \left[\int_a^b |g(x)|dx\right]^2 - I^2.$$

Seleccionando $P_V(x)$ proporcional a $|g(x)|$, es decir

$$P_V(x) = \frac{|g(x)|}{\int_a^b |g(x)|dx}$$

obtendremos que la expresión (1) se convierte en la identidad, ya que

$$\int_a^b \frac{g^2(x)dx}{\frac{|g(x)|}{\int_a^b |g(x)|dx}} = \int_a^b \frac{|g(x)|}{|g(x)|}\,|g(x)|dx \cdot \int_a^b |g(x)|dx = \left[\int_a^b |g(x)|dx\right]^2,$$

y por consiguiente $\mathrm{Var}[H]$ se convierte en igualdad.

En la práctica no se seleccionan funciones $P_V(x)$ complejas ya que ello haría sumamente laborioso el método Monte Carlo. Por otro lado, la función $g(x)$ puede ser una guía para la selección de la función $P_V(x)$.

Así mismo, las integrales de forma

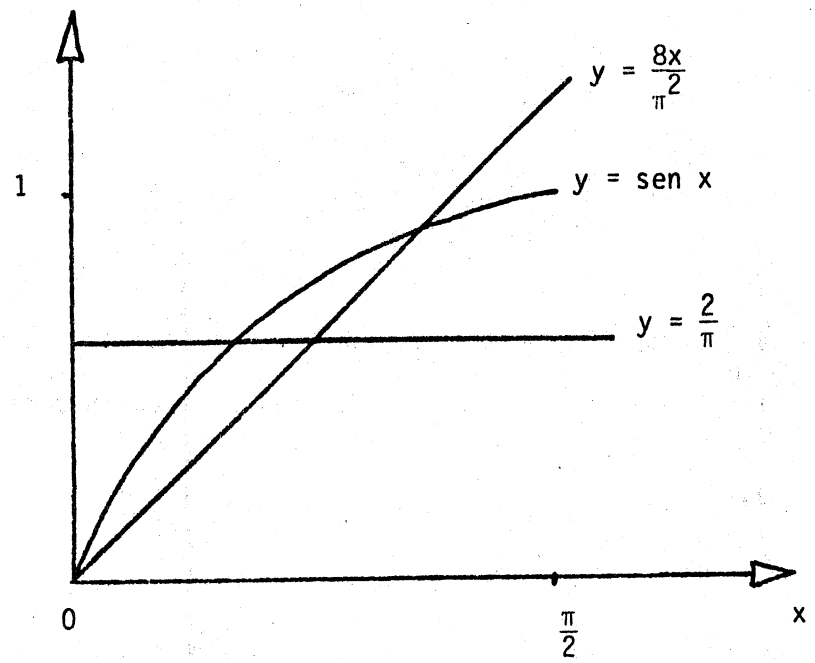
$$I = \int_a^b g(x)dx$$

no se resuelven por el método Monte Carlo, sino por métodos de cuadratura. En cambio, en el caso de integrales múltiples la situación cambia, el método de la cuadratura se vuelve sumamente complejo, mientras que Monte Carlo permanece prácticamente inalterable.

Ejemplo de aplicación:

Sea la integral $I = \int_0^{\frac{\pi}{2}} \operatorname{sen} x\, dx$

cuya solución sabemos es 1.



Con propósito de ilustración considérense las dos funciones de densidad siguientes:

$$\text{(I) } y = P_V(x) = \frac{2}{\pi} \qquad y \qquad \text{(II) } P_V(x) = \frac{8x}{\pi^2}$$

La función de densidad (I) pertenece a una variable aleatoria con distribución uniforme definida en el intervalo $\left[0, \frac{\pi}{2}\right]$. Intuitivamente se observa que la función de densidad (II) satisface mejor las recomendaciones de proporcionalidad, por lo tanto, de ella se esperarán los mejores resultados. Para cada caso se tiene:

$$\text{(a) } P_V(x) = \frac{2}{\pi}$$

$$v = v_{min} + U\,[v_{max} - v_{min}]$$

ó

$$v = \frac{U\pi}{2}\,, \quad \text{ya que } v_{max} = \frac{\pi}{2} \text{ y } v_{min} = 0.$$

Según lo anterior, $I \simeq \frac{1}{n} \sum_{i=1}^{n} \frac{g(v_i)}{P_V(v_i)} = \frac{\pi}{2n} \sum_{i=1}^{n} \text{sen } v_i$.

Generando 10 valores aleatorios $U_1, \ldots, U_{10}$ se producen, según la fórmula $v = \frac{U\pi}{2}$, otros 10, $v_1, \ldots, v_{10}$ y el valor de la integral resulta $I = 0.952$

(b) $P_V(x) = \frac{8x}{\pi^2}$

$$\int_0^v \frac{8x}{\pi^2}\,dx = U$$

$$v = \frac{\pi}{2}\sqrt{U}$$

$$I \simeq \frac{\pi^2}{8n} \sum_{i=1}^{n} \frac{\text{sen } v_i}{v_i}$$

para $n = 10$, $I \simeq 1.016$

La medida del orden del error estaría dada por $3\sqrt{\frac{Var[H]}{n}}$, pero un resultado más práctico sería $0.675\sqrt{\frac{Var[H]}{n}}$, entonces para cada uno de los casos vistos se tendría el error siguiente:

$$Var\,[H]_I = 0.256$$

$$\text{Var } |H|_{II} = 0.016$$

Integración Numérica a Través de Polinomios Ortogonales.

Polinomios Ortogonales.

Definición: Dos funciones, $g_m(x)$ y $g_n(x)$, obtenidas a partir de una familia de funciones $\{g_k(x)\}$, se dicen ser ortogonales con respecto a la función $w(x)$ en el intervalo $[a, b]$, si ellas satisfacen que:

$$\int_a^b w(x)\, g_m(x)\, g_n(x)\, dx = 0, \quad \forall m \neq n \quad y$$

$$\int_a^b w(x)\, g_m^2(x)\, dx = c(m) \neq 0 \quad \forall\, m.$$

Como ejemplo de estas funciones pueden citarse las funciones seno y coseno (sen Rx, cos Rx).

La propiedad de ortogonalidad se interpreta, en términos de espacios vectoriales, como la perpendicularidad existente entre dos vectores, en un espacio n-dimensional (n muy grande) (Figura 14). Los elementos o vectores de este espacio vectorial no son otros que los formados con la familia $\{g_k(x)\}$.

La familia de polinomios $\{x^k\}$, k = 0, 1, 2,..., define un espacio vectorial cuyos elementos no son ortogonales. Otras familias, tales como las de los polinomios de Legendre, Laguerre, Chebyshev y Hermite, además de definir un espacio vectorial, sus elementos son ortogonales.

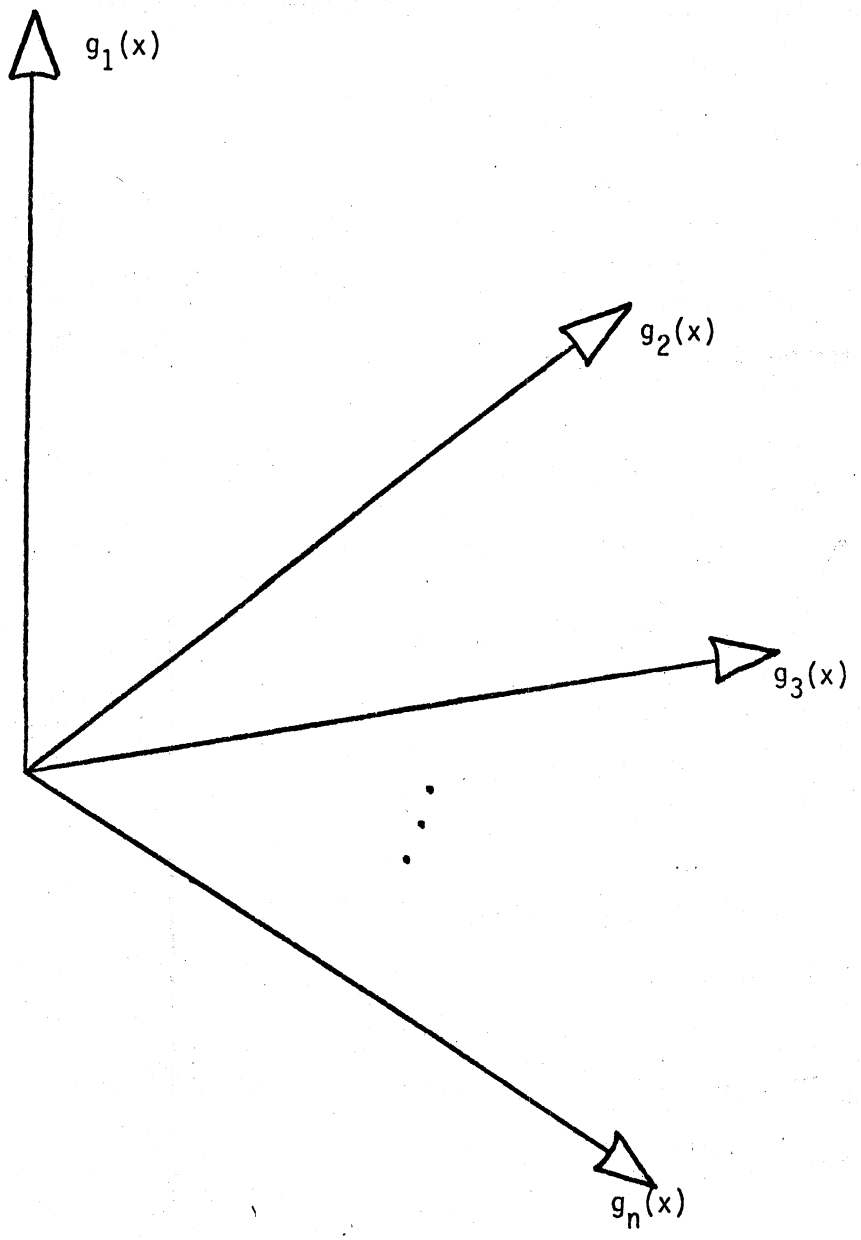


Figura 14.- Familia de funciones ortogonales representada por medio de un espacio vectorial n-dimensional. Cada vector en el espacio representa una función.

(i) Polinomios de Legendre $\{P_n(x)\}$.

Los polinomios de Legendre satisfacen la propiedad de ortogonalidad en el intervalo $[-1, 1]$ con función peso $w(x) = 1$, es decir:

$$\int_{-1}^{1} P_m(x)\, P_n(x)\, dx = \begin{cases} 0, & m \neq n \\ c(m), & m = n \end{cases}$$

Los tres primeros polinomios de Legendre se definen como:

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_2(x) = \frac{1}{2}(3x^2 - 1)$$

Los polinomios de Legendre subsecuentes pueden obtenerse a partir de la fórmula de recurrencia siguiente:

$$P_n(x) = \frac{2n-1}{n} x\, P_{n-1}(x) - \frac{n-1}{n} P_{n-2}(x)$$

(ii) Polinomios de Laguerre $\{L_n(x)\}$

Los polinomios de Laguerre satisfacen la propiedad de ortogonalidad en el intervalo $[0, \infty]$ con función peso $w(x) = e^{-x}$, o sea que:

$$\int_{0}^{\infty} e^{-x} L_m(x)\, L_n(x)\, dx = \begin{cases} 0, & m \neq n \\ c(m), & m = n \end{cases}$$

Los tres primeros polinomios de Laguerre son iguales a:

$$L_0(x) = 1$$

$$L_1(x) = 1 - x$$

$$L_2(x) = x^2 - 4x + 2$$

La fórmula de recurrencia de los polinomios de Laguerre es:

$$L_n(x) = (2n - x - 1)\, L_{n-1}(x) - (n - 1)^2\, L_{n-2}(x).$$

(iii) Polinomios de Chebyshev $\{T_n(x)\}$.

Los polinomios de Chebyshev son ortogonales en el intervalo $[-1, 1]$ con función peso $w(x) = (1 - x^2)^{-\frac{1}{2}}$, dando

$$\int_{-1}^{1} (1 - x^2)^{-\frac{1}{2}}\, Tm(x)\, Tn(x)dx = \begin{cases} 0, & m \neq n \\ c(m), & m = n \end{cases}$$

Los tres primeros polinomios de Chebyshev se definen así:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

y la fórmula de recurrencia es:

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$$

iv) Polinomios de Hermite $\{H_n(x)\}$.

Los polinomios de Hermite satisfacen la propiedad de ortogonalidad

en el intervalo $(-\infty, \infty)$ con función peso $w(x)=e^{-x^2}$, esto es:

$$\int_{-\infty}^{\infty} e^{-x^2} H_m(x) H_n(x) dx = \begin{cases} 0, & m \neq n \\ c(m), & m = n \end{cases}$$

Los tres primeros polinomios de Hermite son:

$$H_0(x) = 1$$

$$H_1(x) = 2x$$

$$H_2(x) = 4x^2 - 2$$

con fórmula de recurrencia $H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x)$

Cada uno de los polinomios definidos arriba es un polinomio con coeficientes reales, de grado n en la variable x y con n raíces reales distintas, las cuales caen dentro del intervalo de integración de cada polinomio. Como ejemplo, todas las raíces del polinomio de Legrendre $P_n(x)$ caen dentro del intervalo $[-1, 1]$.

Una propiedad importante de los polinomios ortogonales es la de generar el espacio vectorial de todos los otros polinomios, ortogonales o no.

Sea $p_n(x)$ un polinomio arbitrario de grado n

$$p_n(x) = \sum_{i=0}^{n} \alpha_i x^i$$

El polinomio $p_n(x)$ puede representarse como la combinación lineal de cualquiera de las familias de polinomios ortogonales mencionadas arriba:

$$p_n(x) = \sum_{i=0}^{n} \beta_i \, Z_i(x)$$

siendo $Z_i(x)$ el polinomio de grado i de una de las familias de polinomios ortogonales.

Ejemplo:

El polinomio de cuarto grado

$p_4(x) = \sum_{i=0}^{4} \alpha_i \, x^i$ puede expresarse de manera única en términos

de los polinomios de Legendre, es decir:

$$p_4(x) = \sum_{i=0}^{4} \beta_i \, P_i(x) \quad \text{ó}$$

$$p_4(x) = \beta_0 + \beta_1 x + \beta_2(\tfrac{3}{2} x^2 - \tfrac{1}{2}) + \beta_3(\tfrac{5}{2} x^3 - \tfrac{3}{2} x) + \beta_4(\tfrac{35}{8} x^4 - \tfrac{15}{4} x^2 + \tfrac{3}{8})$$

Agrupando términos según el grado de la variable x

$$p_4(x) = (\beta_0 - \frac{\beta_2}{2} + \frac{3}{8} \beta_4) + (\beta_1 - \frac{3}{2} \beta_2)x + (\frac{3}{2} \beta_2 - \frac{15}{4} \beta_4)x^2 + \frac{5}{2} \beta_3 x^3 + \frac{35}{8} \beta_4 x^4$$

e igualando con los coeficientes α se deduce que:

$$\beta_4 = \frac{8}{35} \alpha_4$$

$$\beta_3 = \frac{2}{5} \alpha_3$$

$$\beta_2 = \frac{2}{3} (\alpha_2 + \frac{6}{7} \alpha_4)$$

$$\beta_1 = \alpha_1 + \frac{3}{5} \alpha_3$$

y $\beta_0 = \alpha_0 + \frac{1}{3}\alpha_2 + \frac{1}{5}\alpha_4$

Es fácil de verificar que el polinomio

$$p_4(x) = x^4 + 3x^3 - 2x^2 + 2x - 1$$

puede ser expresado también como:

$$p_4(x) = -\frac{22}{15}P_0(x) + \frac{19}{5}P_1(x) - \frac{16}{21}P_2(x) + \frac{6}{5}P_3(x) + \frac{8}{35}P_4(x)$$

Cuadraturas Gaussianas.

En general, el método de las cuadraturas gaussianas propone aproximar la integral

$$\int_a^b w(x)\,F(x)dx$$

por medio de una suma de n términos

$$\sum_{i=1}^{n} W_i F(T_i),$$

siendo los T_i y W_i, ciertos nodos y ponderadores, respectivamente. La función w(x), en la integral, no es otra que la función peso asociada a alguna familia de polinomios ortogonales. Los nodos T_i son las raíces del polinomio ortogonal de grado n perteneciente a la misma familia anterior y el cual es empleado como herramienta de solución del problema. Cada elemento ponderador W_i es la solución de la integral efectuada sobre un po

linomio de grado i (de la misma familia anterior), en el intervalo [a, b].

En otras palabras, muchas de las técnicas de integración numérica, tales como Simpson, Romberg, etc. (donde w(x) = 1) emplean puntos equidistantes x_i, lo cual puede ser conveniente pero no necesario. En los métodos Gaussianos no se seleccionan puntos equidistantes, sino mas bien se emplean puntos (nodos) que son las raíces de un polinomio ortogonal de grado n definido sobre el intervalo [a, b] y con función peso, la función w(x).

La razón de los nodos y ponderadores resultará más clara con lo que se explique a continuación.

(i)* Cuadratura Gauss-Legendre.

En el método de la cuadratura de Legendre, la herramienta de solución a la integral

$$\int_a^b f(x)dx, \quad \text{son los polinomios de Legendre.}$$

El método consiste en aproximar la función f(x) por medio de un polinomio cualquiera de grado n que pase por n puntos x_i y que sea igual a $f(x_i)(\Psi_i)$, más una cierta función error $R_n(x)$, la cual contenga el error en la aproximación

$$f(x) = p_n(x) + R_n(x) \qquad (1)$$

La función $p_n(x)$ puede ser expresada en términos de polinomios Langragianos (ver Capítulo III) como:

*V.I.Krilov, Approximate Calculation of Integrals, Macmillan, New York, 1962.

$$p_n(x) = \sum_{i=0}^{n} L_i(x)\, f(x_i) \qquad (2)$$

donde cada función $L_i(x)$ es el polinomio de grado n igual a:

$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j}\right)$, y la función error, según el método de Lagrange de interpolación, es

$$R_n(x) = \prod_{i=0}^{n} (x - x_i)\, \frac{f^{(n+1)}(\xi)}{(n+1)!}, \quad a < \xi < b \qquad (3)$$

Substituyendo las expresiones (2) y (3) en (1) se tiene:

$$f(x) = \sum_{i=0}^{n} L_i(x)\, f(x_i) + \left[\prod_{i=0}^{n} (x - x_i)\right] \frac{f^{(n+1)}(\xi)}{(n+1)!}, \quad a < \xi < b \qquad (4)$$

Siendo el intervalo de ortogonalidad de los polinomios de Legendre $[-1, 1]$, es necesario, antes de integrar la función f(x), efectuar un cambio de variable a fin de modificar el intervalo de integración de $[a, b]$ a $[-1, 1]$.

Haciendo

$z = \frac{2x - (a+b)}{b - a}$, la función f(x) se convierte en una nueva función en z,

$$f(x) = F(z) = f\left(\frac{z(b - a) + (a + b)}{2}\right)$$

quedando la expresión (4) definida entonces como:

$$F(z) = \sum_{i=0}^{n} L_i(z)\, F(z_i) + \left[\prod_{i=0}^{n} (z - z_i)\right] \frac{F^{(n+1)}(\xi)}{(n+1)!}, \quad -1 < \xi < 1 \tag{5}$$

donde $L_i(z) = \prod_{\substack{j=0 \\ j\neq 1}}^{n} \left(\frac{z-z_j}{z_i-z_j}\right)$

Si la función f(x) (ó F(z)) fuera un polinomio de grado 2n+1, entonces, el término $\sum_{i=0}^{n} L_i(z)\, F(z_i)$ sería un polinomio de, a lo más, grado n, el término $\prod_{i=0}^{n} (z - z_i)$ sería un polinomio de grado n+1, y el término $\frac{F^{(n+1)}(\xi)}{(n+1)!}$ sería un polinomio de grado n.

Considerando éste último término igual a un polinomio de grado n, $q_n(z)$, e integrando entre -1 y 1, la expresión (5) se convierte en:

$$\int_{-1}^{1} F(z)dz = \int_{-1}^{1} \sum_{i=0}^{n} L_i(z)\, F(z_i)dz + \int_{-1}^{1} \left[\prod_{i=0}^{n} (z - z_i)\right] q_n(z)dz.$$

En conclusión, la integral $\int_{-1}^{1} F(z)dz$ puede aproximarse por la expresión

$$\sum_{i=0}^{n} W_i\, F(z_i) \tag{6}$$

con un error dado por la expresión

$$\text{Error} = \int_{-1}^{1} \prod_{i=0}^{n} (z - z_i)\, q_n(z)dz,$$

siendo

$$W_i = \int_{-1}^{1} L_i(z)dz = \int_{-1}^{1} \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{z - z_j}{z_i - z_j}\right)dz . \qquad (7)$$

El objetivo fundamental será seleccionar aquellos valores z_i que anulen (o minimicen) el error en la aproximación. La ortogonalidad de los polinomios de Legendre puede emplearse con este propósito.

Ahora bien, expandiendo los polinomios $q_n(z)$ y $\prod_{i=1}^{n} (z - z_i)$ en términos de polinomios de Legendre se tiene:

$$q_n(z) = \sum_{j=0}^{n} c_j P_j(z) \quad y \qquad (8)$$

$$\prod_{i=0}^{n} (z - z_i) = \sum_{i=0}^{n+1} b_i P_i(z) . \qquad (9)$$

El producto $q_n(z) \prod_{i=0}^{n} (z - z_i)$ resulta entonces igual a:

$\sum_{i=0}^{n+1} \sum_{j=0}^{n} b_i c_j P_i(z) P_j(z)$, mismo que al ser integrado entre $[-1, 1]$ se reduce a:

$$\text{Error} = \sum_{i=0}^{n} b_i c_i \int_{-1}^{1} [P_i(z)]^2 dz$$

Para que el error sea igual a cero será menester igualar los n+1 coeficientes b_i a cero. Por otro lado, en la expresaión (9), si todos los b_i ($i = 0, \ldots, n$) son iguales a cero, exceptuando el último, se ob

tiene que:

$$\prod_{i=0}^{n} (z - z_i) = b_{n+1}\, P_{n+1}(z). \qquad (10)$$

En esta última expresión el polinomio de la izquierda tiene como coeficientes del término de mayor orden, la unidad. Por lo tanto, si la expresión (10) se cumple, b_{n+1} deberá ser igual al inverso del coeficiente del término de mayor orden del polinomio $P_{n+1}(z)$. Mas aún, si ambos polinomios son iguales, ellos deberán tener las mismas raíces, o lo que es lo mismo, las raíces de $P_{n+1}(z)$ deberán de ser las mismas que las raíces de $\prod_{i=0}^{n} (z - z_i)$, que son precisamente, todas las z_i.

Habiendo definido los puntos z_i como las raíces del polinomio de Legendre de grado n+1, $P_{n+1}(z)$, los pesos W_i en la aproximación integral (expresión (6)) se definen de manera inmediata a través de la expresión (7).

Todo este desarrollo se ha basado en la suposición inicial de una función f(x) polinómica de grado menor o igual a 2n+1. Si éste no fuese el caso, se incurriría en un error en la aproximación igual a:

$$\text{Error}_n = \frac{2^{2n+3} \left[(n+1)!\right]^4}{(2n+3)\left[(2n+2)!\right]^3} F^{(2n+2)}(\xi), \quad -1 < \xi < 1 .$$

(ii) Cuadratura Gauss-Laguerre.

El desarrollo del método de la cuadratura Gauss-Laguerre, es similar al anterior. Los nodos z_i son, en este caso, las raíces del polino-

mio de Laguerre de grado n+1, $L_{n+1}(z)$, y los pesos asociados a cada nodo están dados por la expresión

$$W_i = \int_0^\infty e^{-z} L_i(z)dz = \int_0^\infty e^{-z} \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{z - z_j}{z_i - z_j}\right)dz.$$

El error en la aproximación integral de funciones no-polinómicas está dado por:

$$E_n = \frac{[(n+1)!]^2}{(2n+2)!} F^{(2n+2)}(\xi), \quad 0 < \xi < \infty$$

(iii) Cuadratura Gauss-Chebyshev.

Análogamente, los nodos z_i son las raíces del polinomio $T_{n+1}(z)$, los pesos W_i están dados por la expresión

$$W_i = \int_{-1}^{1} \frac{1}{\sqrt{1 - z^2}} L_i(z)\, dz ,$$

y el error para funciones no polinómicas es igual a:

$$E_n = \frac{2\pi}{2^{2n+2}\,(2n+2)!} F^{(2n+2)}(\xi), \quad -1 < \xi < 1 .$$

(iv) Cuadratura Gauss-Hermite

En este caso, z_i serán las raíces de $H_{n+1}(z)$, los pesos W_i estarán definidos por

$$W_i = \int_{-\infty}^{\infty} e^{-z^2} L_i(z)\, dz$$

y el error en funciones no-polinómicas será

$$E_n = \frac{(n+1)!\,\sqrt{\pi}}{2^{n+1}\,(2n+2)!}\; F^{(2n+2)}(\xi), \qquad -\infty < \xi < \infty \ .$$

Conclusiones.

(I) Cualquier función f(x) que exista en el intervalo de integración puede ser integrada empleando alguna de las cuatro cuadraturas anteriores.

(II) Si la función peso w(x) no coincide con la función peso de las cuadraturas Gaussianas, entonces es recomendable usar la siguiente transformación:

$$\int_a^b f(x)dx = \int_a^b w(x)\,\frac{f(x)}{w(x)}\,dx = \int_a^b w(x)\,g(x)dx$$

donde w(x) sería la función peso apropiada.

(III) Si el intervalo de integración no coincide con el intervalo de la cuadratura Gaussiana, entonces la siguiente transformación puede emplearse:

$$z = \frac{2x - (a+b)}{b - a}, \quad \text{intervalo} \quad (-1,\ 1).$$

La subrutina GAUSSQ listada al final del capítulo calcula, según sea el tipo de cuadratura a emplear, los valores de los nodos T_i y ponderadores W_i para diferentes valores de N, el número de términos en la aproximación.

Ejercicio # 6

Distribución de Velocidades Adimensionales.

Gill y Scher* derivaron una expresión de la distribución de velocidades de un fluído en una tubería circular modificando la ecuación de Prandtl.

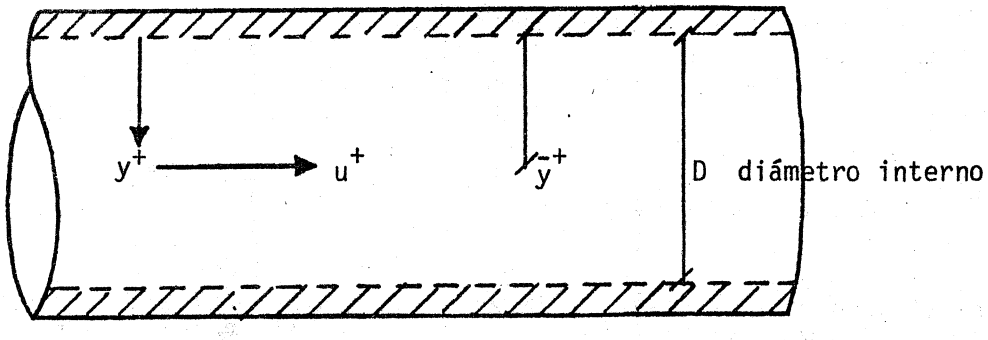
La velocidad adimensional u^+ en la dirección x está dada en función de una distancia adimensional y^+, medida a partir de la cara interna de la tubería.

$$u^+ = \int_0^{y+} \frac{-1 + \sqrt{1 + 4cd}}{2c}\, dy^+ = \int_0^{y+} \frac{2d}{1 + \sqrt{1 + 4cd}}\, dy^+$$

donde

$$c = (0.36)^2\ (y+)^2\ (1 - e^{-\phi(y^+/\bar{y}^+)})^2$$

$$d = 1 - \frac{y+}{\bar{y}^+}$$



* W.N. Gill and M.Scher, "A Modification of the Momentum Transport Hypothesis", A.I.Ch.E. Journal, 7, 61-65 (1961).

$\bar{y}^+$ es la distancia adimensional correspondiente al eje de la tubería y está dada por

$$\bar{y}^+ = \frac{N_{Re}}{2}\sqrt{\frac{\bar{f}}{2}}$$

ϕ es una función empírica de $\bar{y}^+$

$$\phi = \frac{\bar{y}^+ - 60}{22}$$

N_{Re} - número de Reynolds y

$\bar{f}$ - factor de fricción de Fanning.

Ambos, N_{Re} y $\bar{f}$ son parámetros adimensionales que dependen de las propiedades físicas del fluído, la rugosidad de la tubería y la velocidad del fluido, principalmente.

N_{Re} está dado por la fórmula

$$N_{Re} = \frac{vD\rho}{\mu}$$

donde

v - velocidad media del fluido
D - diámetro interno
ρ - densidad
μ - viscosidad

} en unidades compatibles

Para el caso de tuberías circulares no muy rugosas donde N_{Re} fluctúa entre 2000 y 5000, $\bar{f}$ puede representarse adecuadamente por la ecuación de

Blasius

$$\overline{f} = 0.079\, N_{Re}^{-0.25}$$

Escriba un programa general que efectúe el cálculo de la integral

$$\int_a^b f(x)\, dx$$

empleando la cuadratura de Chebyshev y de Legendre con n puntos de expansión. Substituya apropiadamente f(x) y los intervalos (a, b) en la expresión de la velocidad adimensional u^+.

Compare los resultados obtenidos de la integral por las dos cuadraturas para valores del número de Reynolds de

$$N_{Re} = 5000,\ 10000,\ 25000 \text{ y } 50000$$

y para distancias adimensionales de

$$y^+ = 1,\ 2,\ 5,\ 10,\ 20,50,\ 100,\ 200,\ 500 \text{ y } 1000$$

empleando

2, 4, 6, 10 y 15 puntos de expansión.

Grafique en papel semilogarítmico los resultados de $\log\ y^+$ .vs.u^+ obtenidos cuando el número de partes en la expansión es 15. Comente los resultados.

Ejercicio # 7

Considere la siguiente integral:

$$I = \int_{0.01}^{1.0} \frac{dx}{e^x - 1}$$

a).- Evalúese analíticamente.

b).- Aproxime I empleando la cuadratura de Simpson con N paneles igualmente espaciados. Dada la singularidad en el punto x=0, cerca del intervalo de integración .01, es probable encontrar dificultades con la exactitud del método. Corra un programa con N = 10, 20 y 40.

c).- Reescriba I como:

$$I = \int_{0.01}^{1.0} \left(\frac{1}{e^x - 1} - \frac{1}{x}\right) dx + \int_{0.01}^{1.0} \frac{dx}{x}$$

Evalúe la segunda integral analíticamente y la primera usando la cuadratura de Simpson, entonces sume los dos valores para obtener I. Use, otra vez, N = 10, 20 y 40. ¿Obtuvo resultados más exactos que en el inciso (b)? ¿Porqué?.

d).- Repita los incisos (b) y (c) usando la subrutina QUANC8 en lugar de la regla de Simpson. Use una tolerancia de 10^{-5} para ambos, el error absoluto y el error relativo, Imprima el número de funciones evaluadas requeridas. ¿Es QUANC8 más exacta que la regla de Simpson?

```
      SUBROUTINE QUANC8(FUN,A,B,ABSERR,RELERR,RESULT,ERREST,NOFUN,FLAG) 00000000
C                                                                       00000010
      DOUBLE PRECISION FUN, A, B, ABSERR, RELERR, RESULT, ERREST, FLAG  00000020
      INTEGER NOFUN                                                     00000030
C                                                                       00000040
C   ESTIMATE THE INTEGRAL OF FUN(X) FROM A TO B                         00000050
C   TO A USER PROVIDED TOLERANCE.                                       00000060
C   AN AUTOMATIC ADAPTIVE ROUTINE BASED ON                              00000070
C   THE 8-PANEL NEWTON-COTES RULE.                                      00000080
C                                                                       00000090
C   INPUT ..                                                            00000100
C                                                                       00000110
C   FUN      THE NAME OF THE INTEGRAND FUNCTION SUBPROGRAM FUN(X).      00000120
C   A        THE LOWER LIMIT OF INTEGRATION.                            00000130
C   B        THE UPPER LIMIT OF INTEGRATION.(B MAY BE LESS THAN A.)     00000140
C   RELERR   A RELATIVE ERROR TOLERANCE. (SHOULD BE NON-NEGATIVE)       00000150
C   ABSERR   AN ABSOLUTE ERROR TOLERANCE. (SHOULD BE NON-NEGATIVE)      00000160
C                                                                       00000170
C   OUTPUT ..                                                           00000180
C                                                                       00000190
C   RESULT   AN APPROXIMATION TO THE INTEGRAL HOPEFULLY SATISFYING THE  00000200
C            LEAST STRINGENT OF THE TWO ERROR TOLERANCES.               00000210
C   ERREST   AN ESTIMATE OF THE MAGNITUDE OF THE ACTUAL ERROR.          00000220
C   NOFUN    THE NUMBER OF FUNCTION VALUES USED IN CALCULATION OF RESULT.00000230
C   FLAG     A RELIABILITY INDICATOR.  IF FLAG IS ZERO, THEN RESULT     00000240
C            PROBABLY SATISFIES THE ERROR TOLERANCE.  IF FLAG IS        00000250
C            XXX.YYY , THEN  XXX = THE NUMBER OF INTERVALS WHICH HAVE   00000260
C            NOT CONVERGED AND  0.YYY = THE FRACTION OF THE INTERVAL    00000270
C            LEFT TO DO WHEN THE LIMIT ON  NOFUN  WAS APPROACHED.       00000280
C                                                                       00000290
      DOUBLE PRECISION W0,W1,W2,W3,W4,AREA,X0,F0,STONE,STEP,COR11,TEMP  00000300
      DOUBLE PRECISION QPREV,QNOW,QDIFF,QLEFT,ESTERR,TOLERR             00000310
      DOUBLE PRECISION QRIGHT(31),F(16),X(16),FSAVE(8,30),XSAVE(8,30)   00000320
      DOUBLE PRECISION DABS,DMAX1                                       00000330
      INTEGER LEVMIN,LEVMAX,LEVOUT,NOMAX,NOFIN,LEV,NIM,I,J              00000340
C                                                                       00000350
C   ***   STAGE 1 ***   GENERAL INITIALIZATION                          00000360
C   SET CONSTANTS.                                                      00000370
C                                                                       00000380
      LEVMIN = 1                                                        00000390
      LEVMAX = 30                                                       00000400
      LEVOUT = 6                                                        00000410
      NOMAX = 5000                                                      00000420
      NOFIN = NOMAX - 8*(LEVMAX-LEVOUT+2**(LEVOUT+1))                   00000430
C                                                                       00000440
C   TROUBLE WHEN NOFUN REACHES NOFIN                                    00000450
C                                                                       00000460
      W0 =   3956.0D0 / 14175.0D0                                       00000470
      W1 =  23552.0D0 / 14175.0D0                                       00000480
      W2 =  -3712.0D0 / 14175.0D0                                       00000490
      W3 =  41984.0D0 / 14175.0D0                                       00000500
      W4 = -18160.0D0 / 14175.0D0                                       00000510
```

```
C
C   INITIALIZE RUNNING SUMS TO ZERO.                                    00000520
C                                                                       00000530
      FLAG = 0.0D0                                                      00000540
      RESULT = 0.0D0                                                    00000550
      CORR   = 0.0D0                                                    00000560
      ERREST = 0.0D0                                                    00000570
      AREA   = 0.0D0                                                    00000580
      NOFUN = 0                                                         00000590
      IF (A .EQ. B) RETURN                                              00000600
C                                                                       00000610
C   ***   STAGE 2 ***   INITIALIZATION FOR FIRST INTERVAL               00000620
C                                                                       00000630
      LEV = 0                                                           00000640
      NIM = 1                                                           00000650
      X0 = A                                                            00000660
      X(16) = B                                                         00000670
      QPREV  = 0.0D0                                                    00000680
      F0 = FUN(X0)                                                      00000690
      STONE = (B - A) / 16.0D0                                          00000700
      X(8)  =  (X0  + X(16)) / 2.0D0                                    00000710
      X(4)  =  (X0  + X(8))  / 2.0D0                                    00000720
      X(12) =  (X(8)  + X(16)) / 2.0D0                                  00000730
      X(2)  =  (X0  + X(4))  / 2.0D0                                    00000740
      X(6)  =  (X(4)  + X(8))  / 2.0D0                                  00000750
      X(10) =  (X(8)  + X(12)) / 2.0D0                                  00000760
      X(14) =  (X(12) + X(16)) / 2.0D0                                  00000770
      DO 25 J = 2, 16, 2                                                00000780
         F(J) = FUN(X(J))                                               00000790
   25 CONTINUE                                                          00000800
      NOFUN = 9                                                         00000810
C                                                                       00000820
C   ***   STAGE 3 ***   CENTRAL CALCULATION                            00000830
C   REQUIRES QPREV,X0,X2,X4,...,X16,F0,F2,F4,...,F16.                   00000840
C   CALCULATES X1,X3,...X15, F1,F3,...F15,QLEFT,QRIGHT,QNOW,QDIFF,AREA. 00000850
C                                                                       00000860
   30 X(1) = (X0 + X(2)) / 2.0D0                                        00000870
      F(1) = FUN(X(1))                                                  00000880
      DO 35 J = 3, 15, 2                                                00000890
         X(J) = (X(J-1) + X(J+1)) / 2.0D0                               00000900
         F(J) = FUN(X(J))                                               00000910
   35 CONTINUE                                                          00000920
      NOFUN = NOFUN + 8                                                 00000930
      STEP = (X(16) - X0) / 16.0D0                                      00000940
      QLEFT  =  (W0*(F0 + F(8))  + W1*(F(1)+F(7))  + W2*(F(2)+F(6))     00000950
     1  + W3*(F(3)+F(5))  +  W4*F(4)) * STEP                           00000960
      QRIGHT(LEV+1)=(W0*(F(8)+F(16))+W1*(F(9)+F(15))+W2*(F(10)+F(14))   00000970
     1  + W3*(F(11)+F(13)) + W4*F(12)) * STEP                          00000980
      QNOW = QLEFT + QRIGHT(LEV+1)                                      00000990
      QDIFF = QNOW - QPREV                                              00001000
      AREA = AREA + QDIFF                                               00001010
C                                                                       00001020
C   ***   STAGE 4 *** INTERVAL CONVERGENCE TEST                         00001030
C                                                                       00001040
      ESTERR = DABS(QDIFF) / 1023.0D0                                   00001050
      TOLERR = DMAX1(ABSERR,RELERR*DABS(AREA)) * (STEP/STONE)           00001060
      IF (LEV .LT. LEVMIN) GO TO 50                                     00001070
      IF (LEV .GE. LEVMAX) GO TO 62                                     00001080
      IF (NOFUN .GT. NOFIN) GO TO 60                                    00001090
      IF (ESTERR .LE. TOLERR) GO TO 70                                  00001100
                                                                        00001110
```

```
C                                                                       00001120
C   ***   STAGE 5   ***   NO CONVERGENCE                               00001130
C   LOCATE NEXT INTERVAL.                                               00001140
C                                                                       00001150
   50 NIM = 2*NIM                                                       00001160
      LEV = LEV+1                                                       00001170
C                                                                       00001180
C   STORE RIGHT HAND ELEMENTS FOR FUTURE USE.                           00001190
C                                                                       00001200
      DO 52 I = 1, 8                                                    00001210
         FSAVE(I,LEV) = F(I+8)                                          00001220
         XSAVE(I,LEV) = X(I+8)                                          00001230
   52 CONTINUE                                                          00001240
C                                                                       00001250
C   ASSEMBLE LEFT HAND ELEMENTS FOR IMMEDIATE USE.                      00001260
C                                                                       00001270
      QPREV = QLEFT                                                     00001280
      DO 55 I = 1, 8                                                    00001290
         J = -I                                                         00001300
         F(2*J+18) = F(J+9)                                             00001310
         X(2*J+18) = X(J+9)                                             00001320
   55 CONTINUE                                                          00001330
      GO TO 30                                                          00001340
C                                                                       00001350
C   ***   STAGE 6   ***   TROUBLE SECTION                              00001360
C   NUMBER OF FUNCTION VALUES IS ABOUT TO EXCEED LIMIT.                 00001370
C                                                                       00001380
   60 NOFIN = 2*NOFIN                                                   00001390
      LEVMAX = LEVOUT                                                   00001400
      FLAG = FLAG + (B - X0) / (B - A)                                  00001410
      GO TO 70                                                          00001420
C                                                                       00001430
C   CURRENT LEVEL IS LEVMAX.                                            00001440
C                                                                       00001450
   62 FLAG = FLAG + 1.0D0                                               00001460
C                                                                       00001470
C   ***   STAGE 7   ***   INTERVAL CONVERGED                           00001480
C   ADD CONTRIBUTIONS INTO RUNNING SUMS.                                00001490
C                                                                       00001500
   70 RESULT = RESULT + QNOW                                            00001510
      ERREST = ERREST + ESTERR                                          00001520
      COR11  = COR11  + QDIFF / 1023.0D0                                00001530
C                                                                       00001540
C   LOCATE NEXT INTERVAL.                                               00001550
C                                                                       00001560
   72 IF (NIM .EQ. 2*(NIM/2)) GO TO 75                                  00001570
      NIM = NIM/2                                                       00001580
      LEV = LEV-1                                                       00001590
      GO TO 72                                                          00001600
   75 NIM = NIM + 1                                                     00001610
      IF (LEV .LE. 0) GO TO 80                                          00001620
C                                                                       00001630
C   ASSEMBLE ELEMENTS REQUIRED FOR THE NEXT INTERVAL.                   00001640
C                                                                       00001650
      QPREV = QRIGHT(LEV)                                               00001660
      X0 = X(16)                                                        00001670
      F0 = F(16)                                                        00001680
      DO 78 I = 1, 8                                                    00001690
         F(2*I) = FSAVE(I,LEV)                                          00001700
         X(2*I) = XSAVE(I,LEV)                                          00001710
   78 CONTINUE                                                          00001720
      GO TO 30                                                          00001730
C                                                                       00001740
C   ***   STAGE 8   ***   FINALIZE AND RETURN                          00001750
C                                                                       00001760
   80 RESULT = RESULT + COR11                                           00001770
C                                                                       00001780
C   MAKE SURE ERREST NOT LESS THAN ROUNDOFF LEVEL.                      00001790
C                                                                       00001800
      IF (ERREST .EQ. 0.0D0) RETURN                                     00001810
   82 TEMP = DABS(RESULT) + ERREST                                      00001820
      IF (TEMP .NE. DABS(RESULT)) RETURN                                00001830
      ERREST = 2.0D0*ERREST                                             00001840
      GO TO 82                                                          00001850
      END                                                               00001860
```

```
      C     ********** START OF GAUSSQ **********
      C          LAST UPDATED:        FEBRUARY, 1976
            SUBROUTINE GAUSSQ(KIND, N, ALPHA, BETA, KPTS, ENDPTS, B, T, W)
      C
      C        THIS SET OF ROUTINES COMPUTES THE NODES T(J) AND WEIGHTS
      C     W(J) FOR GAUSSIAN-TYPE QUADRATURE RULES WITH PRE-ASSIGNED
      C     NODES.  THESE ARE USED WHEN ONE WISHES TO APPROXIMATE
      C
      C                INTEGRAL (FROM A TO B)  F(X) W(X) DX
      C
      C                                     N
      C     BY                            SUM W  F(T )
      C                                   J=1  J    J
      C
      C     (NOTE W(X) AND W(J) HAVE NO CONNECTION WITH EACH OTHER.)
      C     HERE W(X) IS ONE OF SIX POSSIBLE NON-NEGATIVE WEIGHT
      C     FUNCTIONS (LISTED BELOW), AND F(X) IS THE
      C     FUNCTION TO BE INTEGRATED.  GAUSSIAN QUADRATURE IS PARTICULARLY
      C     USEFUL ON INFINITE INTERVALS (WITH APPROPRIATE WEIGHT
      C     FUNCTIONS), SINCE THEN OTHER TECHNIQUES OFTEN FAIL.
      C
      C        ASSOCIATED WITH EACH WEIGHT FUNCTION W(X) IS A SET OF
      C     ORTHOGONAL POLYNOMIALS.  THE NODES T(J) ARE JUST THE ZEROES
      C     OF THE PROPER N-TH DEGREE POLYNOMIAL.
      C
      C   INPUT PARAMETERS (ALL REAL NUMBERS ARE IN DOUBLE PRECISION)
      C
      C     KIND      AN INTEGER BETWEEN 1 AND 6 GIVING THE TYPE OF
      C               QUADRATURE RULE:
      C
      C     KIND = 1:  LEGENDRE QUADRATURE, W(X) = 1 ON (-1, 1)
      C     KIND = 2:  CHEBYSHEV QUADRATURE OF THE FIRST KIND
      C                W(X) = 1/SQRT(1 - X*X) ON (-1, +1)
      C     KIND = 3:  CHEBYSHEV QUADRATURE OF THE SECOND KIND
      C                W(X) = SQRT(1 - X*X) ON (-1, 1)
      C     KIND = 4:  HERMITE QUADRATURE, W(X) = EXP(-X*X) ON
      C                (-INFINITY, +INFINITY)
      C     KIND = 5:  JACOBI QUADRATURE, W(X) = (1-X)**ALPHA * (1+X)**
      C                BETA ON (-1, 1), ALPHA, BETA .GT. -1.
      C                NOTE: KIND=2 AND 3 ARE A SPECIAL CASE OF THIS.
      C     KIND = 6:  GENERALIZED LAGUERRE QUADRATURE, W(X) = EXP(-X)*
      C                X**ALPHA ON (0, +INFINITY), ALPHA .GT. -1
      C
      C     N         THE NUMBER OF POINTS USED FOR THE QUADRATURE RULE
      C     ALPHA     REAL PARAMETER USED ONLY FOR GAUSS-JACOBI AND GAUSS-
      C               LAGUERRE QUADRATURE (OTHERWISE USE 0.D0).
      C     BETA      REAL PARAMETER USED ONLY FOR GAUSS-JACOBI QUADRATURE--
      C               (OTHERWISE USE 0.D0)
      C     KPTS      (INTEGER) NORMALLY 0, UNLESS THE LEFT OR RIGHT END-
      C               POINT (OR BOTH) OF THE INTERVAL IS REQUIRED TO BE A
      C               NODE (THIS IS CALLED GAUSS-RADAU OR GAUSS-LOBATTO
      C               QUADRATURE).  THEN KPTS IS THE NUMBER OF FIXED
      C               ENDPOINTS (1 OR 2).
      C     ENDPTS    REAL ARRAY OF LENGTH 2.  CONTAINS THE VALUES OF
      C               ANY FIXED ENDPOINTS, IF KPTS = 1 OR 2.
      C     B         REAL SCRATCH ARRAY OF LENGTH N
      C
      C   OUTPUT PARAMETERS (BOTH DOUBLE PRECISION ARRAYS OF LENGTH N)
      C
      C     T         WILL CONTAIN THE DESIRED NODES.
```

```
C        W        WILL CONTAIN THE DESIRED WEIGHTS W(J).
C
C     SUBROUTINES REQUIRED
C
C        SOLVE, CLASS, AND IMTQL2 ARE PROVIDED.  UNDERFLOW MAY SOMETIMES
C        OCCUR, BUT IT IS HARMLESS IF THE UNDERFLOW INTERRUPTS ARE
C        TURNED OFF.  TO DO THIS, THE FIRST CALL OF THE MAIN PROGRAM
C        SHOULD BE
C                  CALL TRAPS (0, 0, 10000, 0, 0)     IN WATFIV
C        OR
C                  CALL INIT                          IN FORTRAN G OR H.
C
C     ACCURACY
C
C        THE ROUTINE WAS TESTED UP TO N = 512 FOR LEGENDRE QUADRATURE,
C        UP TO N = 136 FOR HERMITE, UP TO N = 68 FOR LAGUERRE, AND UP
C        TO N = 10 OR 20 IN OTHER CASES.  IN ALL BUT TWO INSTANCES,
C        COMPARISON WITH TABLES IN REF. 3 SHOWED 12 OR MORE SIGNIFICANT
C        DIGITS OF ACCURACY.  THE TWO EXCEPTIONS WERE THE WEIGHTS FOR
C        HERMITE AND LAGUERRE QUADRATURE, WHERE UNDERFLOW CAUSED SOME
C        VERY SMALL WEIGHTS TO BE SET TO ZERO.  THIS IS, OF COURSE,
C        COMPLETELY HARMLESS.
C
C     METHOD
C
C           THE COEFFICIENTS OF THE THREE-TERM RECURRENCE RELATION
C        FOR THE CORRESPONDING SET OF ORTHOGONAL POLYNOMIALS ARE
C        USED TO FORM A SYMMETRIC TRIDIAGONAL MATRIX, WHOSE
C        EIGENVALUES (DETERMINED BY THE IMPLICIT QL-METHOD WITH
C        SHIFTS) ARE JUST THE DESIRED NODES.  THE FIRST COMPONENTS OF
C        THE ORTHONORMALIZED EIGENVECTORS, WHEN PROPERLY SCALED,
C        YIELD THE WEIGHTS.  THIS TECHNIQUE IS MUCH FASTER THAN USING A
C        ROOT-FINDER TO LOCATE THE ZEROES OF THE ORTHOGONAL POLYNOMIAL.
C        FOR FURTHER DETAILS, SEE REF. 1.  REF. 2 CONTAINS DETAILS OF
C        GAUSS-RADAU AND GAUSS-LOBATTO QUADRATURE ONLY.
C
C     REFERENCES
C
C        1.  GOLUB, G. H., AND WELSCH, J. H., "CALCULATION OF GAUSSIAN
C            QUADRATURE RULES," MATHEMATICS OF COMPUTATION 23 (APRIL,
C            1969), PP. 221-230.
C        2.  GOLUB, G. H., "SOME MODIFIED MATRIX EIGENVALUE PROBLEMS,"
C            SIAM REVIEW 15 (APRIL, 1973), PP. 318-334 (SECTION 7).
C        3.  STROUD AND SECREST, GAUSSIAN QUADRATURE FORMULAS, PRENTICE-
C            HALL, ENGLEWOOD CLIFFS, N.J., 1966.
C
C     ....................................................................
C
      DOUBLE PRECISION B(N), T(N), W(N), ENDPTS(2), MUZERO, T1,
     X GAM, SOLVE, DSQRT, ALPHA, BETA
C
      CALL CLASS (KIND, N, ALPHA, BETA, B, T, MUZERO)
C
C           THE MATRIX OF COEFFICIENTS IS ASSUMED TO BE SYMMETRIC.
C           THE ARRAY T CONTAINS THE DIAGONAL ELEMENTS, THE ARRAY
C           B THE OFF-DIAGONAL ELEMENTS.
C           MAKE APPROPRIATE CHANGES IN THE LOWER RIGHT 2 BY 2
C           SUBMATRIX.
C
      IF (KPTS.EQ.0)  GO TO 100
```

```
      IF (KPTS.EQ.2)  GO TO  50
C
C           IF KPTS=1, ONLY T(N) MUST BE CHANGED
C
      T(N) = SOLVE(ENDPTS(1), N, T, B)*B(N-1)**2 + ENDPTS(1)
      GO TO 100
C
C           IF KPTS=2, T(N) AND B(N-1) MUST BE RECOMPUTED
C
   50 GAM = SOLVE(ENDPTS(1), N, T, B)
      T1 = ((ENDPTS(1) - ENDPTS(2))/(SOLVE(ENDPTS(2), N, T, B) - GAM))
      B(N-1) = DSQRT(T1)
      T(N) = ENDPTS(1) + GAM*T1
C
C           NOTE THAT THE INDICES OF THE ELEMENTS OF B RUN FROM 1 TO N-1
C           AND THUS THE VALUE OF B(N) IS ARBITRARY.
C           NOW COMPUTE THE EIGENVALUES OF THE SYMMETRIC TRIDIAGONAL
C           MATRIX, WHICH HAS BEEN MODIFIED AS NECESSARY.
C           THE METHOD USED IS A QL-TYPE METHOD WITH ORIGIN SHIFTING
C
  100 W(1) = 1.0D0
      DO 105 I = 2, N
  105    W(I) = 0.0D0
C
      CALL IMTQL2 (N, T, B, W, IERR)
      DO 110 I = 1, N
  110    W(I) = MUZERO * W(I) * W(1)
C
      RETURN
      END
C
C
C
      DOUBLE PRECISION FUNCTION SOLVE(SHIFT, N, A, B)
C
C       THIS PROCEDURE PERFORMS ELIMINATION TO SOLVE FOR THE
C       N-TH COMPONENT OF THE SOLUTION DELTA TO THE EQUATION
C
C             (JN - SHIFT*IDENTITY) * DELTA  = EN,
C
C       WHERE EN IS THE VECTOR OF ALL ZEROES EXCEPT FOR 1 IN
C       THE N-TH POSITION.
C
C       THE MATRIX JN IS SYMMETRIC TRIDIAGONAL, WITH DIAGONAL
C       ELEMENTS A(I), OFF-DIAGONAL ELEMENTS B(I).  THIS EQUATION
C       MUST BE SOLVED TO OBTAIN THE APPROPRIATE CHANGES IN THE LOWER
C       2 BY 2 SUBMATRIX OF COEFFICIENTS FOR ORTHOGONAL POLYNOMIALS.
C
C
      DOUBLE PRECISION SHIFT, A(N), B(N), ALPHA
C
      ALPHA = A(1) - SHIFT
      NM1 = N - 1
      DO 10 I = 2, NM1
   10    ALPHA = A(I) - SHIFT - B(I-1)**2/ALPHA
      SOLVE = 1.0D0/ALPHA
      RETURN
      END
C
C
```

```
      C
            SUBROUTINE CLASS(KIND, N, ALPHA, BETA, B, A, MUZERO)
      C
      C           THIS PROCEDURE SUPPLIES THE COEFFICIENTS A(J), B(J) OF THE
      C         RECURRENCE RELATION
      C
      C                B P (X) = (X - A ) P    (X) - B   P   (X)
      C                 J J              J   J-1       J-1 J-2
      C
      C         FOR THE VARIOUS CLASSICAL (NORMALIZED) ORTHOGONAL POLYNOMIALS,
      C         AND THE ZERO-TH MOMENT
      C
      C              MUZERO = INTEGRAL W(X) DX
      C
      C         OF THE GIVEN POLYNOMIAL'S WEIGHT FUNCTION W(X).  SINCE THE
      C         POLYNOMIALS ARE ORTHONORMALIZED, THE TRIDIAGONAL MATRIX IS
      C         GUARANTEED TO BE SYMMETRIC.
      C
      C           THE INPUT PARAMETER ALPHA IS USED ONLY FOR LAGUERRE AND
      C         JACOBI POLYNOMIALS, AND THE PARAMETER BETA IS USED ONLY FOR
      C         JACOBI POLYNOMIALS.  THE LAGUERRE AND JACOBI POLYNOMIALS
      C         REQUIRE THE GAMMA FUNCTION.
      C
      C     ..................................................................
      C
            DOUBLE PRECISION A(N), B(N), MUZERO, ALPHA, BETA
            DOUBLE PRECISION ABI, A2B2, DGAMMA, PI, DSQRT, AB
            DATA PI / 3.14159265358979 3D0/
      C
            NM1 = N - 1
            GO TO (10, 20, 30, 40, 50, 60), KIND
      C
      C                    KIND = 1:  LEGENDRE POLYNOMIALS P(X)
      C                    ON (-1, +1), W(X) = 1.
      C
         10 MUZERO = 2.0D0
            DO 11 I = 1, NM1
               A(I) = 0.0D0
               ABI = I
         11    B(I) = ABI/DSQRT(4*ABI*ABI - 1.0D0)
            A(N) = 0.0D0
            RETURN
      C
      C                    KIND = 2:  CHEBYSHEV POLYNOMIALS OF THE FIRST KIND T(X)
      C                    ON (-1, +1), W(X) = 1 / SQRT(1 - X*X)
      C
         20 MUZERO = PI
            DO 21 I = 1, NM1
               A(I) = 0.0D0
         21    B(I) = 0.5D0
            B(1) = DSQRT(0.5D0)
            A(N) = 0.0D0
            RETURN
      C
      C                    KIND = 3:  CHEBYSHEV POLYNOMIALS OF THE SECOND KIND U(X)
      C                    ON (-1, +1), W(X) = SQRT(1 - X*X)
      C
         30 MUZERO = PI/2.0D0
            DO 31 I = 1, NM1
               A(I) = 0.0D0
```

```
      31    B(I) = 0.5D0
            A(N) = 0.0D0
            RETURN
C
C                 KIND = 4:  HERMITE POLYNOMIALS H(X) ON (-INFINITY,
C                 +INFINITY), W(X) = EXP(-X**2)
C
      40 MUZERO = DSQRT(PI)
            DO 41 I = 1, NM1
               A(I) = 0.0D0
      41       B(I) = DSQRT(I/2.0D0)
            A(N) = 0.0D0
            RETURN
C
C                 KIND = 5:  JACOBI POLYNOMIALS P(ALPHA, BETA)(X) ON
C                 (-1, +1), W(X) = (1-X)**ALPHA + (1+X)**BETA, ALPHA AND
C                 BETA GREATER THAN -1
C
      50 AB = ALPHA + BETA
            ABI = 2.0D0 + AB
            MUZERO = 2.0D0 ** (AB + 1.0D0) * DGAMMA(ALPHA + 1.0D0) * DGAMMA(
           X BETA + 1.0D0) / DGAMMA(ABI)
            A(1) = (BETA - ALPHA)/ABI
            B(1) = DSQRT(4.0D0*(1.0D0 + ALPHA)*(1.0D0 + BETA)/((ABI + 1.0D0)*
           1  ABI*ABI))
            A2B2 = BETA*BETA - ALPHA*ALPHA
            DO 51 I = 2, NM1
               ABI = 2.0D0*I + AB
               A(I) = A2B2/((ABI - 2.0D0)*ABI)
      51       B(I) = DSQRT (4.0D0*I*(I + ALPHA)*(I + BETA)*(I + AB)/
           1    ((ABI*ABI - 1)*ABI*ABI))
            ABI = 2.0D0*N + AB
            A(N) = A2B2/((ABI - 2.0D0)*ABI)
            RETURN
C
C                 KIND = 6:  LAGUERRE POLYNOMIALS L(ALPHA)(X) ON
C                 (0, +INFINITY), W(X) = EXP(-X) * X**ALPHA, ALPHA GREATER
C                 THAN -1.
C
      60 MUZERO = DGAMMA(ALPHA + 1.0D0)
            DO 61 I = 1, NM1
               A(I) = 2.0D0*I - 1.0D0 + ALPHA
      61       B(I) = DSQRT(I*(I + ALPHA))
            A(N) = 2.0D0*N - 1 + ALPHA
            RETURN
            END
C
C
            SUBROUTINE IMTQL2(N, D, E, Z, IERR)
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE IMTQL2,
C     NUM. MATH. 12, 377-383(1968) BY MARTIN AND WILKINSON,
C     AS MODIFIED IN NUM. MATH. 15, 450(1970) BY DUBRULLE.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 241-248(1971).
C     THIS IS A MODIFIED VERSION OF THE 'EISPACK' ROUTINE IMTQL2.
C
C     THIS SUBROUTINE FINDS THE EIGENVALUES AND FIRST COMPONENTS OF THE
C     EIGENVECTORS OF A SYMMETRIC TRIDIAGONAL MATRIX BY THE IMPLICIT QL
C     METHOD.
C
```

```
      C     ON INPUT:
      C
      C        N IS THE ORDER OF THE MATRIX;
      C
      C        D CONTAINS THE DIAGONAL ELEMENTS OF THE INPUT MATRIX;
      C
      C        E CONTAINS THE SUBDIAGONAL ELEMENTS OF THE INPUT MATRIX
      C          IN ITS FIRST N-1 POSITIONS.  E(N) IS ARBITRARY;
      C
      C        Z CONTAINS THE FIRST ROW OF THE IDENTITY MATRIX.
      C
      C     ON OUTPUT:
      C
      C        D CONTAINS THE EIGENVALUES IN ASCENDING ORDER.  IF AN
      C          ERROR EXIT IS MADE, THE EIGENVALUES ARE CORRECT BUT
      C          UNORDERED FOR INDICES 1, 2, ..., IERR-1;
      C
      C        E HAS BEEN DESTROYED;
      C
      C        Z CONTAINS THE FIRST COMPONENTS OF THE ORTHONORMAL EIGENVECTORS
      C          OF THE SYMMETRIC TRIDIAGONAL MATRIX.  IF AN ERROR EXIT IS
      C          MADE, Z CONTAINS THE EIGENVECTORS ASSOCIATED WITH THE STORED
      C          EIGENVALUES;
      C
      C        IERR IS SET TO
      C          ZERO       FOR NORMAL RETURN,
      C          J          IF THE J-TH EIGENVALUE HAS NOT BEEN
      C                     DETERMINED AFTER 30 ITERATIONS.
      C
      C     ------------------------------------------------------------------
      C
            INTEGER I, J, K, L, M, N, II, MML, IERR
            DOUBLE PRECISION D(N), E(N), Z(N), B, C, F, G, P, R, S, MACHEP
            DOUBLE PRECISION DSQRT, DABS, DSIGN
      C
      C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
      C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
      C                MACHEP = 16.0D0**(-13) FOR LONG FORM ARITHMETIC
      C                ON S360 **********
            DATA MACHEP/Z3410000000000000/
      C
            IERR = 0
            IF (N .EQ. 1) GO TO 1001
      C
            E(N) = 0.0D0
            DO 240 L = 1, N
               J = 0
      C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********
        105    DO 110 M = L, N
                  IF (M .EQ. N) GO TO 120
                  IF (DABS(E(M)) .LE. MACHEP * (DABS(D(M)) + DABS(D(M+1))))
           X         GO TO 120
        110    CONTINUE
      C
        120    P = D(L)
               IF (M .EQ. L) GO TO 240
               IF (J .EQ. 30) GO TO 1000
               J = J + 1
      C     ********** FORM SHIFT **********
               G = (D(L+1) - P) / (2.0D0 * E(L))
```

```
361.                R = DSQRT(G*G+1.0D0)
362.                G = D(M) - P + E(L) / (G + DSIGN(R, G))
363.                S = 1.0D0
364.                C = 1.0D0
365.                P = 0.0D0
366.                MML = M - L
367.      C
368.      C     :::::::::: FOR I=M-1 STEP -1 UNTIL L DO -- ::::::::::
369.                DO 200 II = 1, MML
370.                   I = M - II
371.                   F = S * E(I)
372.                   B = C * E(I)
373.                   IF (DABS(F) .LT. DABS(G)) GO TO 150
374.                   C = G / F
375.                   R = DSQRT(C*C+1.0D0)
376.                   E(I+1) = F * R
377.                   S = 1.0D0 / R
378.                   C = C * S
379.                   GO TO 160
380.        150        S = F / G
381.                   R = DSQRT(S*S+1.0D0)
382.                   E(I+1) = G * R
383.                   C = 1.0D0 / R
384.                   S = S * C
385.        160        G = D(I+1) - P
386.                   R = (D(I) - G) * S + 2.0D0 * C * B
387.                   P = S * R
388.                   D(I+1) = G + P
389.                   G = C * R - B
390.      C     :::::::::: FORM FIRST COMPONENT OF VECTOR ::::::::::
391.                   F = Z(I+1)
392.                   Z(I+1) = S * Z(I) + C * F
393.        200        Z(I) = C * Z(I) - S * F
394.      C
395.                D(L) = D(L) - P
396.                E(L) = G
397.                E(M) = 0.0D0
398.                GO TO 105
399.        240 CONTINUE
400.      C
401.      C     :::::::::: ORDER EIGENVALUES AND EIGENVECTORS ::::::::::
402.              DO 300 II = 2, N
403.                I = II - 1
404.                K = I
405.                P = D(I)
406.      C
407.                DO 260 J = II, N
408.                   IF (D(J) .GE. P) GO TO 260
409.                   K = J
410.                   P = D(J)
411.        260     CONTINUE
412.      C
413.                IF (K .EQ. I) GO TO 300
414.                D(K) = D(I)
415.                D(I) = P
416.                P = Z(I)
417.                Z(I) = Z(K)
418.                Z(K) = P
419.        300 CONTINUE
420.      C
```

```
421.            GO TO 1001
422.      C     :::::::::: SET ERROR -- NO CONVERGENCE TO AN
423.      C                EIGENVALUE AFTER 30 ITERATIONS ::::::::::
424.       1000 IERR = L
425.       1001 RETURN
426.      C     :::::::::: LAST CARD OF IMTQL2 ::::::::::
427.            END
```

CAPITULO V

SOLUCION NUMERICA DE ECUACIONES DIFERENCIALES ORDINARIAS

Ecuaciones Diferenciales Ordinarias.

En una ecuación diferencial de primer orden $y' = f(y,t)$, el objetivo es encontrar aquella función $y(t)$ que satisfaga la ecuación diferencial.

La ecuación diferencial

$$\frac{dy}{dt} = f(y,t) = y + t^2$$

tiene como solución una familia de curvas $y = y(t)$. Si la función $f(y,t)$ fuera igual a y, la ecuación diferencial tendría como solución la función $y(t) = Ce^t$

Al elegir un valor inicial $y_0 = y(0)$ se seleccionaría una de las curvas que integran la familia de curvas.

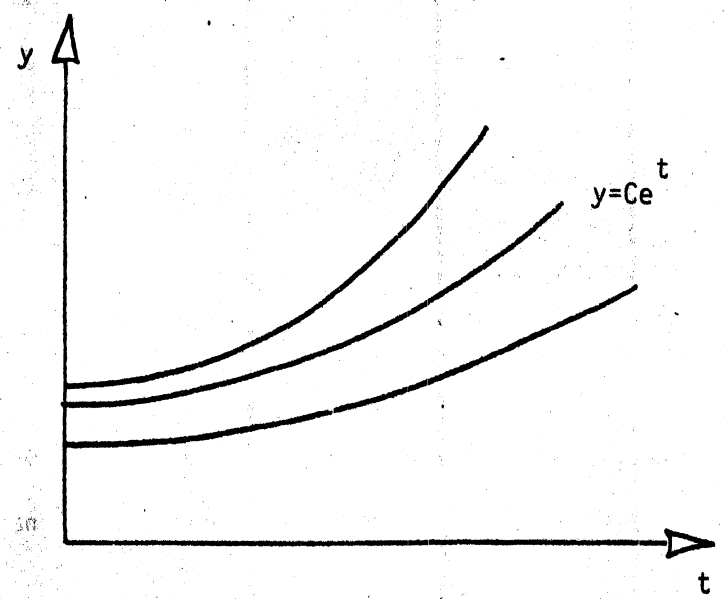


Figura 15. Familia de curvas solución a la EDO $y' = y$.

En el caso de ecuaciones diferenciales ordinarias (EDO) con varias variables dependientes, el problema consiste en resolver el sistema de EDO de forma

$$y' = f(y, z, t)$$

$$z' = g(y, z, t).$$

La solución analítica de este sistema involucra dos constantes, requiriéndose para ello dos piezas adicionales de información. Si los valores de y y z fuesen especificados para un cierto valor t_0 de la variable independiente t, entonces la solución sería única. El problema de la determinación de los valores de y y z para valores de $t > t_0$ es conocido con el nombre de "problema del valor inicial".

Finalmente, una EDO de orden n, donde la derivada de más alto orden es despejable, puede ser escrita también como un sistema de n ecuaciones diferenciales de primer orden introduciendo n-1 nuevas variables.

La ecuación diferencial ordinaria

$$u'' = g(u, u', t)$$

puede escribirse también como:

$$z' = g(u, z, t)$$

$$u' = z$$

Dado que en la práctica no todas las ecuaciones diferenciales pueden ser resueltas analíticamente, es deseable entonces definir otros mé-

todos de solución tales como los métodos numéricos. Además de la información requerida en la solución analítica de ecuaciones diferenciales, la solución numérica requiere de dos piezas adicionales de información:

(i) Una expresión que indique el error a tolerar, y

(ii) Una expresión que indique cuanto se desea pagar por alcanzar tal solución.

Las aproximaciones numéricas de interés en este capítulo involucran los métodos de paso a paso (o de diferencias o de variables discretas) los cuales consisten en la generación de una secuencia de puntos, t_0, t_1, ..., con intervalo variable $h_n = t_{n+1} - t_n$. En cada punto t_n, la solución $y(t_n)$ es aproximada por un número y_n el cual es calculado a partir de ciertos valores anteriormente estimados.

En general, si K valores estimados son empleados en el cálculo del valor y_{n+1}, el método se denomina, método de K pasos. Cuando K es igual a 1, el método recibe el nombre de método de paso simple, y cuando $K > 1$ el método se nombra método de multipasos.

Un ejemplo del método de paso-simple es el método de Euler, donde el valor y_{n+1} es calculado a través de una extrapolación lineal del valor anterior y_n. Tal es el caso de la ecuación diferencial ordinaria

$$y' = f(y, t) \quad \text{con condición inicial} \quad y(t_0) = y_0,$$

donde la pendiente de la solución y(t) es calculada a partir de la condición inicial, esto es:

$$y_0' = f(y_0, t_0)$$

y donde el valor estimado y_1, correspondiente a $y(t_1)$ es calculado empleando los dos primeros términos de la serie de Taylor

$$y(t_1) \simeq y_1 = y_0 + h_0 f(t_0, y_0).$$

Equivalentemente, para $t_2 = t_1 + h_1$, $y(t_2)$ será aproximada mediante la expresión

$$y_2 = y_1 + h_1 f(t_1, y_1)$$

y en general

$$y(t_{n+1}) \simeq y_{n+1} = y_n + h_n f(t_n, y_n)$$

El método de Euler, sin embargo, puede crear errores considerables en la solución de algunas EDO. En el caso de la ecuación $y' = y$, donde e^t es la solución, a medida que t aumenta, el error será multiplicado por un factor e^t produciendo inestabilidad en la ED (Figura 15), en cambio, para $y' = -y$, donde e^{-t} es la solución, los errores se verán disminuidos a medida que t aumente de valor, produciéndose una solución estable (Figura 16).

A continuación se presentan los métodos numéricos de aproximación más comúnmente empleados para la solución de ecuaciones diferenciales ordinarias. Unicamente se presentarán los algoritmos de solución sin hacer alución alguna al problema de inestabilidad.

(I) Método de Taylor

Sea y(t) la solución de cierta EDO, la cual al expresarse en tér-

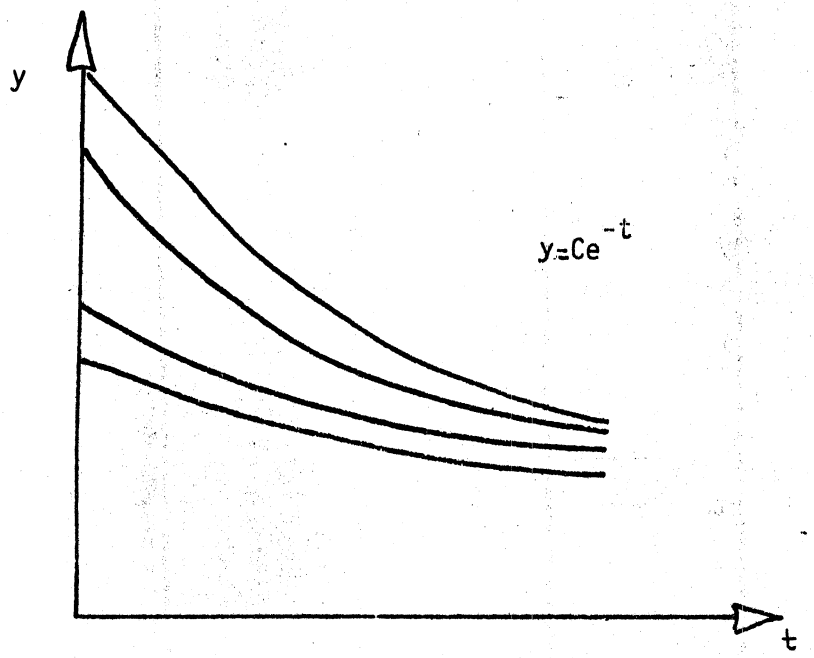


Figura 16. Familia de curvas solución a la EDO $y' = -y$.

minos de series de Taylor queda como:

$$y(t+h) = y(t) + h_n\, y'(t) + \frac{h_n^2}{2!}\, y''(t) + \ldots$$

El método de Euler puede ser visto como un caso particular del método de Taylor. Una aproximación de orden p tendría la forma siguiente:

$$y_{n+1} = y_n + h_n\, y_n' + \frac{h_n^2}{2!}\, y_n'' + \ldots + \frac{h_n^P}{P!}\, y_n^{(P)},$$

donde las derivadas $y', y'', \ldots$, serían evaluadas en términos de las derivadas parciales de la función f en la ecuación diferencial

$$y' = f(y, t).$$

Usando la regla de la cadena repetidas veces, pueden demostrarse las igualdades siguientes:

$$y'' = f_y\, y' + f_t$$

$$= f_y \cdot f + f_t$$

$$y''' = f_{yy}\, y' \cdot f + f_y\, f_y\, y' + f_y\, f_t + f_{tt} + 2f_{ty} \cdot y'$$

$$= f_{yy} \cdot f^2 + f_y^2 \cdot f + f_y\, f_t + 2f_{ty} \cdot f + f_{tt}$$

etc.

(II) Método Runge-Kutta

Este método tiene la ventaja fundamental sobre el método de Series de Taylor de que no requiere la evaluación de las derivadas de segundo y mayor órden. La aproximación se obtiene evaluando la función $f(y,t)$ varias veces.

El método de Runge-Kutta de 4° orden, por ejemplo, está dado por la expresión

$$y_{n+1} = y_n + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$$

donde

$$k_0 = h_n f(y_n, t_n)$$

$$k_1 = h_n f(y_n + \frac{1}{2} k_0, t_n + \frac{h_n}{2})$$

$$k_2 = h_n f(y_n + \frac{1}{2} k_1, t_n + \frac{h_n}{2})$$

$$k_3 = h_n f(y_n + k_2, t_n + h_n)$$

Nótese que cuatro evaluaciones de la función $f(y,t)$ son requeridas

en cada paso $y_n \rightarrow y_{n+1}$.

El método de Runge-Kutta puede ser visto como una extensión del método de Simpson a la solución numérica de ecuaciones diferenciales. Si f fuera función de t únicamente, Simpson y Runge-Kutta serían idénticos

$$\int_{t_n}^{t_{n+1}} f(t)dt \simeq \frac{h_n}{6} \left[f(t_n) + 4f(\frac{t_n+t_{n+1}}{2}) + f(t_{n+1})\right]$$

El método de Runge-Kutta es fácil de programar y en la mayoría de los problemas es numéricamente estable. El método puede iniciarse por sí mismo, dado que sólo se requiere de una solución inicial y_0. Además, el tamaño del intervalo h_n puede hacerse variar en cada evaluación.

(III) Método de Multipasos.

En los dos métodos anteriores los valores de y_{n+1} fueron calculados exclusivamente en base a los valores y_n, t_n y h_n. Es razonable pensar que si se empleara un número mayor de información, se mejoraría la exactitud. Por ejemplo, en la aproximación de $y(t_n)$ se podrían emplear los resultados previos,

$$y_{n-1}, y_{n-2}, \cdots \text{ y } f_{n-1}, f_{n-2}, \cdots$$

siendo cada f_i igual a $f(y_i, t_i)$. El método de multipasos se basa en esta idea y es sumamente efectivo. La fórmula general del método de multipasos lineal es

$$y_{n+1} = \sum_{i=1}^{k} \alpha_i \, y_{n+1-i} + h_n \sum_{i=0}^{k} \beta_i \, f_{n+1-i}$$

donde k es un entero y α_i ó β_i son coeficientes diferentes de cero. La fórmula es lineal en f.

Una vez que el método se ha inicializado, la evaluación del término y_{n+1} requerirá de los términos

$$y_n, y_{n-1}, \cdots, y_{n-k+1}, \text{ y } f_n, f_{n-1}, \cdots, f_{n-k+1}.$$

Es importante notar que si $\beta_0 = 0$, el método será explícito, y que si $\beta_0 \neq 0$ el método será implícito, ya que se requerirá el cálculo de ambos, f_{n+1} y y_{n+1}, en un mismo paso.

Por lo general, el método de multipasos se trabaja iterativamente de la manera siguiente:

i) Un método explícito, llamado predictor, es aplicado primeramente.

ii) Un método implícito, llamado corrector, es aplicado una o varias veces, a continuación.

Aplicando los dos métodos conjuntamente, el método completo se denomina método predictor-corrector. Un ejemplo es el método de Adams de cuarto orden, donde el método predictor está dado por la expresión

$$y_{n+1} = y_n + \frac{h_n}{24}(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3})$$

y el método corrector viene expresado como

$$y_{n+1} = y_n + \frac{h_n}{24} (9 f_{n+1} + 19 f_n - 5 f_{n-1} + f_{n-2})$$

Ambas fórmulas son de 4° orden, es decir, requieren de cuatro evaluaciones de la función f en cada paso.

El algoritmo iterativo para el cálculo de y_{n+1} es como sigue:

(i) Empleando el método predictor calcule $y_{(n+1)}^{(0)}$, o sea una primera aproximación de y_{n+1}.

(ii) Evalúe la función

$$f_{n+1}^{(i)} = f(y_{n+1}^{(i)}, t_{n+1})$$

(iii) Calcule una mejor aproximación de y_{n+1} usando el método corrector

($y_{n+1}^{(i)}$ es la i-ésima aproximación de y_{n+1}).

(iv) Si el valor absoluto de la diferencia $(y_{n+1}^{(i+1)} - y_{n+1}^{(i)})$ resulta mayor que una cierta tolerancia ε, entonces incremente i en una unidad y vaya al paso (ii); de otra manera defina $y_{n+1} = y_{n+1}^{(i+1)}$ y el problema se da por resuelto.

Tres son los procesos que tienen lugar en este algoritmo:

(a) El paso del predictor (i), el cual denotaremos por la letra P

(b) El paso de la evaluación (ii), denotado por E; y

(c) El paso de la corrección (iii) denotado por C.

El paso (iv) puede ser reemplazado por un paso en el cual se efectúan exactamente m iteraciones del corrector.

El método resultante se expresa en forma compacta como:

$P(EC)^m$

Problemas con Valores en la Frontera

Hasta aquí hemos discutido EDO con condiciones iniciales en la variable dependiente bien definidas. Ahora deseamos resolver un problema donde las condiciones de la variable dependiente se especifiquen para diversos valores de la variable independiente. Tal problema se reconoce como problema con valores en la "frontera".

Los métodos para la solución de estos problemas son completamente diferentes a los de valor inicial. Aquí daremos un método el cual permitirá reducir un problema con valores en la frontera a una secuencia de problemas con valor inicial.

Sea el siguiente sistema de EDO

$y' = f(y, t)$ donde $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$,

$y_1(0) = \alpha$

$y_2(b) = \beta$, y $b \neq 0$

El algoritmo de solución es como sigue:

(i) Seleccione un valor ξ que aproxime el valor de $y_2(0)$.

(ii) Resuelva el problema de valor inicial

$\hat{y}' = f(\hat{y}, t)$, $\hat{y}(0) = \begin{pmatrix} \alpha \\ \xi \end{pmatrix}$, donde $\hat{y}$ es una aproximación de la función y.

(iii) Si el valor absoluto $|\hat{y}_2(b)-\beta|$ es menor que una tolerancia ε, entonces haga $y(t) \simeq \hat{y}(t)$; de otra manera ajuste ξ y regrese al paso (ii).

El valor de ξ está estrechamente relacionado con los conceptos de región de convergencia del método de Newton-Raphson para la solución de sistemas de ecuaciones no-lineales.

APENDICE A

ALGORITMO DE THOMAS PARA LA SOLUCION DE MATRICES TRIDIAGONALES

Las ecuaciones son:

$$a_i u_{i-1} + b_i u_i + c_i u_{i+1} = d_i$$

para $1 \leq i \leq R$

con $a_1 = c_R = 0$

El algoritmo es como sigue:
Primero, estime

$$\beta_i = b_i - \frac{a_i c_{i-1}}{\beta_{i-1}} \quad \text{con} \quad \beta_1 = b_1$$

y

$$\gamma_i = \frac{d_i - a_i \gamma_{i-1}}{\beta_i} \quad \text{con} \quad \gamma_1 = \frac{d_1}{b_1}$$

Los valores de la variable dependiente se calculan a partir de

$$u_R = \gamma_R \qquad \text{y} \qquad u_i = \gamma_i - \frac{c_i u_{i-1}}{\beta_i}$$

ALGORITMO PARA LA SOLUCION DE MATRICES PENTADIAGONALES

Las ecuaciones son:

$$a_i u_{i-2} + b_i u_{i-1} + c_i u_i + d_i u_{i+1} + e_i u_{i+2} = f_i$$

para $1 \leq i \leq R$

con $a_1 = b_1 = a_2 = e_{R-1} = d_R = c_R = 0$

El algoritmo es como sigue:
Primero calcule

$$\delta_1 = d_1/c_1$$

$$\lambda_1 = e_1/c_1$$

$$\gamma_1 = f_1/c_1$$

y

$$\mu_2 = c_2 - b_2 \delta_1$$

$$\delta_2 = (d_2 - b_2\lambda_1)/\mu_2$$
$$\lambda_2 = e_2/\mu_2$$
$$\gamma_2 = (f - b_2\gamma_1)/\mu_2$$

Entonces para valores de i entre $3 \leq i \leq (R-2)$, calcule

$$\beta_i = b_i - a_i\delta_{i-2}$$
$$\mu_i = c_i - \beta_i\delta_{i-1} - a_i\lambda_{i-2}$$
$$\delta_i = (d_i - \beta_i\lambda_{i-1})/\mu_i$$
$$\lambda_i = e_i/\mu_i$$
$$\gamma_i = (f_i - \beta_i\gamma_{i-1} - a_i\gamma_{i-2})/\mu_i$$

Luego calcule

$$\beta_{R-1} = b_{R-1} - a_{R-1}\delta_{R-3}$$
$$\mu_{R-1} = c_{R-1} - \beta_{R-1}\delta_{R-2} - a_{R-1}\lambda_{R-3}$$
$$\delta_{R-1} = (d_{R-1} - \beta_{R-1}\delta_{R-2})/\mu_{R-1}$$
$$\gamma_{R-1} = (f_{R-1} - \beta_{R-1}\gamma_{R-2} - a_{R-1}\gamma_{R-3})/\mu_{R-1}$$

y

$$\beta_R = b_R - a_R\delta_{R-2}$$
$$\mu_R = c_R - \beta_R\delta_{R-1} - a_R\lambda_{R-2}$$
$$\gamma_R = (f_R - \beta_R\gamma_{R-1} - a_R\gamma_{R-2})\mu_R$$

Los β_i y μ_i son usados únicamente en el cálculo de δ_i, λ_i y γ_i y no necesitan ser almacenados en memoria después de ser usados. Los δ_i, λ_i y γ_i deben ser almacenados conforme son usados en la solución hacia atrás. Esto es

$$u_R = \gamma_R$$
$$u_{R-1} = \gamma_{R-1} - \delta_{R-1}u_R$$

y

$$u_i = \gamma_i - \delta_i u_{i+1} - \lambda_i u_{i+2}$$

para $(R-2) \geq i \geq 1$

ALGORITMO PARA LA SOLUCION DE MATRICES BI-TRIDIAGONALES

Las ecuaciones son:

$$a_i^{(1)} u_{i-1} + a_i^{(2)} v_{i-1} + b_i^{(1)} u_i + b_i^{(2)} v_i + c_i^{(1)} u_{i+1} + c_i^{(2)} v_{i+1} = d_i^{(1)}$$

y

$$a_i^{(3)} u_{i-1} + a_i^{(4)} v_{i-1} + b_i^{(3)} u_i + b_i^{(4)} v_i + c_i^{(3)} u_{i+1} + c_i^{(4)} v_{i+1} = d_i^{(2)}$$

para $1 \leq i \leq R$

con $a_1^{(m)} = c_R^{(m)} = 0$ para $1 \leq m \leq 4$

El algoritmo es como sigue:
Primero calcule

$$\beta_i^{(1)} = b_i^{(1)} - a_i^{(1)} \lambda_{i-1}^{(1)} - a_i^{(2)} \lambda_{i-1}^{(3)}$$

$$\beta_i^{(2)} = b_i^{(2)} - a_i^{(1)} \lambda_{i-1}^{(2)} - a_i^{(2)} \lambda_{i-1}^{(4)}$$

$$\beta_i^{(3)} = b_i^{(3)} - a_i^{(3)} \lambda_{i-1}^{(1)} - a_i^{(4)} \lambda_{i-1}^{(3)}$$

$$\beta_i^{(4)} = b_i^{(4)} - a_i^{(3)} \lambda_{i-1}^{(2)} - a_i^{(4)} \lambda_{i-1}^{(4)}$$

con $\beta_1^{(m)} = b_1^{(m)}$ para $1 \leq m \leq 4$

y

$$\delta_i^{(1)} = d_i^{(1)} - a_i^{(1)} \gamma_{i-1}^{(1)} - a_i^{(2)} \gamma_{i-1}^{(2)}$$

$$\delta_i^{(2)} = d_i^{(2)} - a_i^{(3)} \gamma_{i-1}^{(1)} - a_i^{(4)} \gamma_{i-1}^{(2)}$$

con $\delta_1^{(1)} = d_1^{(1)}$ y $\delta_1^{(2)} = d_1^{(2)}$

y $\mu_i = \beta_i^{(1)} \cdot \beta_i^{(4)} - \beta_i^{(2)} \cdot \beta_i^{(3)}$

Los $\beta_i^{(m)}$, $\delta_i^{(m)}$ y μ_i son almacenados en memoria para facilitar el cálculo de las siguientes funciones y no requieren ser almacenados después de su utilización en

$$\lambda_i^{(1)} = (\beta_i^{(4)} c_i^{(1)} - \beta_i^{(2)} c_i^{(3)}) / \mu_i$$

$$\lambda_i^{(2)} = (\beta_i^{(1)} c_i^{(2)} - \beta_i^{(3)} c_i^{(4)}) / \mu_i$$

$$\lambda_i^{(3)} = (\beta_i^{(1)} c_i^{(3)} - \beta_i^{(3)} c_i^{(1)}) / \mu_i$$

$$\lambda_i^{(4)} = (\beta_i^{(1)} c_i^{(4)} - \beta_i^{(3)} c_i^{(2)}) / \mu_i$$

y

$$\gamma_i^{(1)} = (\beta_i^{(4)} \delta_i^{(1)} - \beta_i^{(2)} \delta_i^{(2)}) / \mu_i$$

$$\gamma_i^{(2)} = (\beta_i^{(1)} \delta_i^{(2)} - \beta_i^{(3)} \delta_i^{(1)}) / \mu_i$$

Los valores de $\lambda_i^{(m)}$ y $\gamma_i^{(m)}$ deben ser almacenados conforme son empleados en la solución hacia atrás. Esto es

$$u_R = \gamma_R^{(1)}$$

$$v_R = \gamma_R^{(2)}$$

y

$$u_i = \gamma_i^{(1)} - \lambda_i^{(1)} u_{i+1} - \lambda_i^{(2)} v_{i+1}$$

$$v_i = \gamma_i^{(2)} - \lambda_i^{(3)} u_{i+1} - \lambda_i^{(4)} v_{i+1}$$

para $(R - 1) \geq i \geq 1$

ALGORITMO PARA LA SOLUCION DE MATRICES TRI-TRIDIAGONALES

Las ecuaciones son:

$$a_i^{(1)} u_{i-1} + a_i^{(2)} v_{i-1} + a_i^{(3)} w_{i-1} + b_i^{(1)} u_i + b_i^{(2)} v_i + b_i^{(3)} w_i + c_i^{(1)} u_{i+1} + c_i^{(2)} v_{i+1} + c_i^{(3)} w_{i+1} = d_i^{(1)}$$

$$a_i^{(4)} u_{i-1} + a_i^{(5)} v_{i-1} + a_i^{(6)} w_{i-1} + b_i^{(4)} u_i + b_i^{(5)} v_i + b_i^{(6)} w_i + c_i^{(4)} u_{i+1} + c_i^{(5)} v_{i+1} + c_i^{(6)} w_{i+1} = d_i^{(2)}$$

$$a_i^{(7)} u_{i-1} + a_i^{(8)} v_{i-1} + a_i^{(9)} w_{i-1} + b_i^{(7)} u_i + b_i^{(8)} v_i + b_i^{(9)} w_i + c_i^{(7)} u_{i+1} + c_i^{(8)} v_{i+1} + c_i^{(9)} w_{i+1} = d_i^{(3)}$$

para $1 \leq i \leq R$

con $a_1^{(m)} = c_R^{(m)} = 0$ para $1 \leq m \leq 9$

El algoritmo es como sigue:
Primero calcule

$$\beta_i^{(1)} = b_i^{(1)} - a_i^{(1)}\lambda_{i-1}^{(1)} - a_i^{(2)}\lambda_{i-1}^{(4)} - a_i^{(3)}\lambda_{i-1}^{(7)}$$
$$\beta_i^{(2)} = b_i^{(2)} - a_i^{(1)}\lambda_{i-1}^{(2)} - a_i^{(2)}\lambda_{i-1}^{(5)} - a_i^{(3)}\lambda_{i-1}^{(8)}$$
$$\beta_i^{(3)} = b_i^{(3)} - a_i^{(1)}\lambda_{i-1}^{(3)} - a_i^{(2)}\lambda_{i-1}^{(6)} - a_i^{(3)}\lambda_{i-1}^{(9)}$$
$$\beta_i^{(4)} = b_i^{(4)} - a_i^{(4)}\lambda_{i-1}^{(1)} - a_i^{(5)}\lambda_{i-1}^{(4)} - a_i^{(6)}\lambda_{i-1}^{(7)}$$
$$\beta_i^{(5)} = b_i^{(5)} - a_i^{(4)}\lambda_{i-1}^{(2)} - a_i^{(5)}\lambda_{i-1}^{(5)} - a_i^{(6)}\lambda_{i-1}^{(8)}$$
$$\beta_i^{(6)} = b_i^{(6)} - a_i^{(4)}\lambda_{i-1}^{(3)} - a_i^{(5)}\lambda_{i-1}^{(6)} - a_i^{(6)}\lambda_{i-1}^{(9)}$$
$$\beta_i^{(7)} = b_i^{(7)} - a_i^{(7)}\lambda_{i-1}^{(1)} - a_i^{(8)}\lambda_{i-1}^{(4)} - a_i^{(9)}\lambda_{i-1}^{(7)}$$
$$\beta_i^{(8)} = b_i^{(8)} - a_i^{(7)}\lambda_{i-1}^{(2)} - a_i^{(8)}\lambda_{i-1}^{(5)} - a_i^{(9)}\lambda_{i-1}^{(8)}$$
$$\beta_i^{(9)} = b_i^{(9)} - a_i^{(7)}\lambda_{i-1}^{(3)} - a_i^{(8)}\lambda_{i-1}^{(6)} - a_i^{(9)}\lambda_{i-1}^{(9)}$$

con $\beta_1^{(m)} = b_1^{(m)}$ para $1 \le m \le 9$

y

$$\delta_i^{(1)} = d_i^{(1)} - a_i^{(1)}\gamma_{i-1}^{(1)} - a_i^{(2)}\gamma_{i-1}^{(2)} - a_i^{(3)}\gamma_{i-1}^{(3)}$$
$$\delta_i^{(2)} = d_i^{(2)} - a_i^{(4)}\gamma_{i-1}^{(1)} - a_i^{(5)}\gamma_{i-1}^{(2)} - a_i^{(6)}\gamma_{i-1}^{(3)}$$
$$\delta_i^{(3)} = d_i^{(3)} - a_i^{(7)}\gamma_{i-1}^{(1)} - a_i^{(8)}\gamma_{i-1}^{(2)} - a_i^{(9)}\gamma_{i-1}^{(3)}$$

con $\delta_1^{(m)} = d_1^{(m)}$ para $1 \le m \le 3$

Los $\beta_i^{(m)}$ son usados en la estimación de otras funciones y no requieren ser almacenados después del cálculo de:

$$\Theta_i^{(1)} = \beta_i^{(5)}\beta_i^{(9)} - \beta_i^{(6)}\beta_i^{(8)}$$
$$\Theta_i^{(2)} = \beta_i^{(6)}\beta_i^{(7)} - \beta_i^{(4)}\beta_i^{(9)}$$
$$\Theta_i^{(3)} = \beta_i^{(4)}\beta_i^{(8)} - \beta_i^{(5)}\beta_i^{(7)}$$
$$\Theta_i^{(4)} = \beta_i^{(3)}\beta_i^{(8)} - \beta_i^{(2)}\beta_i^{(9)}$$
$$\Theta_i^{(5)} = \beta_i^{(1)}\beta_i^{(9)} - \beta_i^{(2)}\beta_i^{(7)}$$
$$\Theta_i^{(6)} = \beta_i^{(2)}\beta_i^{(7)} - \beta_i^{(1)}\beta_i^{(8)}$$
$$\Theta_i^{(7)} = \beta_i^{(2)}\beta_i^{(6)} - \beta_i^{(3)}\beta_i^{(5)}$$
$$\Theta_i^{(8)} = \beta_i^{(3)}\beta_i^{(4)} - \beta_i^{(1)}\beta_i^{(6)}$$
$$\Theta_i^{(9)} = \beta_i^{(1)}\beta_i^{(5)} - \beta_i^{(2)}\beta_i^{(4)}$$

$$\mu_i = \Theta_i^{(1)}\beta_i^{(1)} + \Theta_i^{(2)}\beta_i^{(2)} + \Theta_i^{(3)}\beta_i^{(3)}$$

Los $\delta_i^{(m)}$, $\Theta_i^{(m)}$ y μ_i son empleados en la estimación de otras funciones y no requieren ser almacenados después del cálculo de:

$$\lambda_i^{(1)} = (\Theta_i^{(1)}c_i^{(1)} + \Theta_i^{(4)}c_i^{(4)} + \Theta_i^{(7)}c_i^{(7)})/\mu_i$$
$$\lambda_i^{(2)} = (\Theta_i^{(1)}c_i^{(2)} + \Theta_i^{(4)}c_i^{(5)} + \Theta_i^{(7)}c_i^{(8)})/\mu_i$$
$$\lambda_i^{(3)} = (\Theta_i^{(1)}c_i^{(3)} + \Theta_i^{(4)}c_i^{(6)} + \Theta_i^{(7)}c_i^{(9)})/\mu_i$$
$$\lambda_i^{(4)} = (\Theta_i^{(2)}c_i^{(1)} + \Theta_i^{(5)}c_i^{(4)} + \Theta_i^{(8)}c_i^{(7)})/\mu_i$$
$$\lambda_i^{(5)} = (\Theta_i^{(2)}c_i^{(2)} + \Theta_i^{(5)}c_i^{(5)} + \Theta_i^{(8)}c_i^{(8)})/\mu_i$$
$$\lambda_i^{(6)} = (\Theta_i^{(2)}c_i^{(3)} + \Theta_i^{(5)}c_i^{(6)} + \Theta^{(8)}c_i^{(9)})/\mu_i$$
$$\lambda_i^{(7)} = (\Theta_i^{(3)}c_i^{(1)} + \Theta_i^{(6)}c_i^{(4)} + \Theta_i^{(9)}c_i^{(7)})/\mu_i$$
$$\lambda_i^{(8)} = (\Theta_i^{(3)}c_i^{(2)} + \Theta_i^{(6)}c_i^{(5)} + \Theta_i^{(9)}c_i^{(8)})/\mu_i$$
$$\lambda_i^{(9)} = (\Theta_i^{(3)}c_i^{(3)} + \Theta_i^{(6)}c_i^{(5)} + \Theta_i^{(9)}c_i^{(9)})/\mu_i$$

y

$$\gamma_i^{(1)} = (\Theta_i^{(1)}\delta_i^{(1)} + \Theta_i^{(4)}\delta_i^{(2)} + \Theta_i^{(7)}\delta_i^{(3)})/\mu_i$$
$$\gamma_i^{(2)} = (\Theta_i^{(2)}\delta_i^{(1)} + \Theta_i^{(5)}\delta_i^{(2)} + \Theta_i^{(8)}\delta_i^{(3)})/\mu_i$$
$$\gamma_i^{(3)} = (\Theta_i^{(3)}\delta^{(1)} + \Theta_i^{(6)}\delta_i^{(2)} + \Theta_i^{(9)}\delta_i^{(3)})/\mu_i$$

Los valores de $\lambda_i^{(m)}$ y $\gamma_i^{(m)}$ deben ser almacenados conforme son empleados en la solución hacia atrás. Esto es:

$$u_R = \gamma_R^{(1)}$$
$$v_R = \gamma_R^{(2)}$$
$$w_R = \gamma_R^{(3)}$$

y

$$u_i = \gamma_i^{(1)} - \lambda_i^{(1)}u_{i+1} - \lambda_i^{(2)}v_{i+1} - \lambda_i^{(3)}w_{i+1}$$
$$v_i = \gamma_i^{(2)} - \lambda_i^{(4)}u_{i+1} - \lambda_i^{(5)}v_{i+1} - \lambda_i^{(6)}w_{i+1}$$
$$w_i = \gamma_i^{(3)} - \lambda_i^{(7)}u_{i+1} - \lambda_i^{(8)}v_{i+1} - \lambda_i^{(9)}w_{i+1}$$

para $(R-1) \geq i \geq 1$

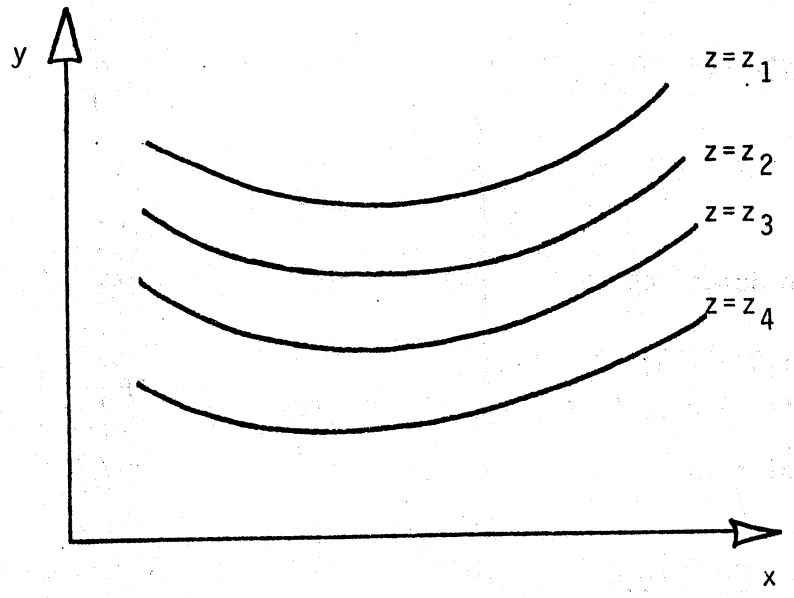
APENDICE B

PROCEDIMIENTO PARA EL AJUSTE DE FAMILIAS DE CURVAS *

Supongamos una función de dos variables independientes x y z,

$$y = y(x,z)$$

o gráficamente



Un procedimiento para ajustar un polinomio a esta familia de curvas es la siguiente:

a) Leer los valores de x vs y, para $z = z_1$

b) Ajustar un polinomio (por mínimos cuadrados) a estos puntos, obteniéndose una ecuación de la forma.

$$y = a_{1,0} + a_{1,1}x + a_{1,2}x^2 + \ldots + a_{1,n}x^n \quad \text{para } z = z_1$$

* *Según notas de Tomás Limón.*

en donde los coeficientes $a_{1,0}$ $a_{1,1}$, ... $a_{1,n}$ se determinan mediante el ajuste polinomial.

c) Repetir los pasos (a) y (b) para las curvas correspondientes a $z = z_2$, $z = z_3$, ..., etc., obteniéndose

$$y = a_{2,0} + a_{2,1} x + a_{2,2} x^2 + \ldots a_{2,n} x^n \quad \text{para } z = z_2$$

$$y = a_{3,0} + a_{3,1} x + a_{3,2} x^2 + \ldots + a_{3,n} x^n \quad \text{para } z = z_3 \ \ldots \text{ etc.}$$

d) Este conjunto de polinomios se reduce a uno del tipo

$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$$

en donde los coeficientes b's son funciones de z, notando que $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, ..., corresponden a $z = z_1$, $z = z_2$, $z = z_3$, ..., etc. Por lo tanto, los coeficientes en (a) están dados por expresiones del tipo

$$b_0 = C_{0,0} + C_{0,1} z + C_{0,2} z^2 + \ldots$$

$$b_1 = C_{1,0} + C_{1,1} z + C_{1,2} z^2 + \ldots$$

$$\vdots$$

$$b_n = C_{n,0} + C_{n,1} z + C_{n,2} z^2 + \ldots$$

Los coeficientes $C_{0,0}$, $C_{0,1}$, $C_{0,2}$, ..., de la ecuación para b_0, son el resultado de un ajuste polinomial donde se han considerado los puntos $(z_1, a_{1,0})$, $(z_2, a_{2,0})$, $(z_3, a_{3,0})$, ..., etc.

De manera semejante, los coeficientes $C_{1,0}$, $C_{1,1}$, $C_{1,2}$ son el re-

sultado de un ajuste polinomial a los siguientes puntos

$(z_1, a_{1,1})$, $(z_2, a_{2,1})$, $(z_3, a_{3,1})$, ..., etc.

y así sucesivamente hasta obtener los coeficientes de los polinomios de b_2, b_3, ..., b_n.

APENDICE C

El siguiente es un listado por nombre de 99 subrutinas principales y alrededor de 140 subrutinas auxiliares. Estas subrutinas pueden constituir una herramienta útil para aquellos lectores que en el futuro su trabajo demande la elaboración de programas de cómputo ligados a problemas reales más específicos.

Las subrutinas se han clasificado en cinco grupos de acuerdo al problema que intentan resolver. Al margen izquierdo aparece el nombre de la subrutina principal, y a continuación, en el margen derecho, aparece una breve descripción del problema. En algunos casos, generalmente donde el problema a resolver es más complejo, se enlistan inmediatamente después de la descripción, un grupo de subrutinas auxiliares las cuales deberán acompañar a la subrutina principal.

PAQUETE DE SUBRUTINAS BASICAS

DECOMP Y SOLVE	Usadas conjuntamente permiten resolver el sistema AX=B. Donde A - matriz de coeficiente B - vector columna X - vector solución
SPLINE Y SEVAL	Usadas conjuntamente permiten interpolar aplicando el método de "splines" cúbicos.
QUANC8	Integra la función f(x) entre los límites A y B usando el método de 8-páneles de Newton-Cotes.
RKF45	Diseñada para resolver sistemas de ecuaciones diferenciales de primer orden por el método de Runge-Kutta-Fehlberg de cuarto-quinto orden. RKFS FEHL
ZEROIN	Calcula las raíces de la función f(x) entre los intervalos AX y BX.
FMIN	Localiza el punto X dentro del intervalo AX, BX donde la función f(x) adquiere un mínimo.

PAQUETE DE SUBRUTINAS NUMERO UNO

ZRPOLY	Calcula raíces de polinomios con coeficientes reales. UERTST ZRPQLB ZRPQLC ZRPQLD ZRPQLE ZRPQLF ZRPQLG ZRPQLH ZRPQLI
FFT	Aplica transformada de Fourier a varias variables complejas.

ICSFKU

Ajusta por mínimos cuadrados una serie de polinomios de tercer grado (splines) a un conjunto de puntos fijos en el plano.

ICSFKV
UERTST

ROOT

Calcula una raíz de la ecuación no lineal f(x)=0.

LINSYS

Paquete de subrutinas empleadas en la solución de sistemas de ecuaciones no simétricas del orden de 150 o menor.

LSDCMP*
LSSOLV
LSIMPR

CSPLIN

Versión modificada de la subrutina "SPLINE" (ver SPLINE).

CSEVAL

Versión modificada de la subrutina "SEVAL" (ver SEVAL).

LINSLSQ

Resuelve el problema ponderado de mínimos cuadrados siguiente:

$$\min_{x} \sum_{i=1}^{M} (W(i)*(B(i)-f(x,T(i))**2)$$

donde f(x,T(i)) es un "modelo pre-especificado.

HECOMP
HOLVE
RESID
COVAR

QNMDER

Encuentra el mínimo de la función $f(x_1,\ldots,x_N)$ usando el método "quasi-Newton" con primeras derivadas parciales.

LNSRCH
OUTPT
C1MDCH
UPCHOL

QNMDIF

Encuentra el mínimo de la función $f(x_1,\ldots,x_N)$ usando el método "quasi-Newton" con diferencias finitas.

APROXG
LINSCH
OUTPT
C1MDCH
UPCHOL

NSOIA	Resuelve un sistema de ecuaciones no lineales de forma f(x)=0, donde f y x son vectores de N componentes. LSINVT* INVRT2*
LCMNA	Intenta definir el punto X en el cual la función escalar f(x), dos veces derivable y continua, adquiere su mínimo, sujeta a M ecuaciones restricción, pudiendo ser estas ecuaciones estructuradas con igualdades o con desigualdades. DOT LSOL ELTSOL LDLSOL WQDLSL PSMNVC FNDBND FRMCHL DELCON ADDCON KTMULT NEWCON PRJHSS UPZTGZ RFINEX LNSRCH
MNA	Encuentra el mínimo de la función f(x) de N variables independientes sujeta a no restricción. LNSRCH OUTPT FORMCH HESS
ZANLYT	Define las raíces de una función analítica compleja usando el método de Mueller. UERTST
ZSYSTM	Calcula el vector solución del sistema no lineal de N ecuaciones y N incógnitas f(x)=0. Donde f y x son vectores de N componentes. UERTST

LCLIN

Resuelve el siguiente problema de programación lineal:

minimiza la función lineal $C_1X_1+\ldots+C_NX_N$ sujeta a las restricciones lineales

$$\begin{array}{lll} A_{11}X_1+\ldots+A_{1N}X_N & R(1) & B_1 \\ \vdots & \vdots & \vdots \\ A_{M1}X_1+\ldots+A_{MN}X_N & R(M) & B_M \end{array}$$

donde cada R(I) puede ser definica como

" $\leq$ ", " $\geq$ ", ó " = ".

LDLSOL
SQDLSL
PSMNVC
DELCON
ADDCON
KTMULT
RFINEX
NEWCON

VARPRO

Regresión múltiple no lineal. Dado un conjunto de N observaciones, Y(1), ..., Y(N) de una variable dependiente Y, donde Y(I) corresponde a la IV variable(s) independiente T(I,1),...,T(I, IV), VARPRO trata de calcular un ajuste ponderado de mínimos cuadrados a la función ETA (el modelo) definida como:

$$\text{ETA(alf,beta; T)} = \sum_{j=1}^{L} \text{beta}_j*\text{phi}_j(\text{alf;T}) + \text{phi}_{L+1}(\text{alf;T})$$

donde phi son funciones no lineales. En otras palabras, determina los parámetros lineales beta(j) y el vector de parámetros no lineales alf, minimizando

$$\text{NORMA(RESIDUAL)}^2 = \sum_{k=1}^{N} W_i*(Y_i-\text{ETA(alf,beta;}T_i))$$

DPA
ORFAC1
ORFAC2
BACSUB.
POSTPR
COV

	XNORM INIT VARERR
REALTR	Realiza la transformada de Fourier de un conjunto 2*N de datos reales.
LAGRN1	Ajusta un polinomio de Lagrange de grado N, a N+1 puntos.
BESIO	Calcula valores de la función Besel modificada de primera clase y orden cero. FCNMON MONERR NATSIO
BESEIO	Calcula valores de la función Besel modificada de primera clase y orden cero multiplicada por EXP(-X). FCNMON MONERR NATSIO
BESEI1	Calcula valores de la función Besel modificada de primera clase y orden uno multiplicada por EXP(-X). FCNMON MONERR NATSI1
BESI1	Calcula valores de la función Besel modificada de primera clase y orden uno. FCNMON MONERR NATSI1
BESJO	Calcula valores de la función Besel de primera clase y orden cero. FCNMON MONERR
BESJ1	Calcula valores de la función Besel de primera clase y orden uno. FCNMON MONERR

BESKO

Calcula valores de la función Besel modificada de segunda clase y orden cero.

FCNMON
MONERR
NATSKO

BESEKO

Calcula valores de la función Besel modificada de segunda clase y orden cero multiplicada por EXP(-X).

FCNMON
MONERR
NATSKO

ICSSCU

Realiza suavizamiento de datos a través de "splines" cúbicos.

UERTST

IKATCU

Aproxima una función f(x) por el método de Chebyshev.

LEQT1F
LUDATF
LUELMF
UERTST

PAQUETE DE SUBRUTINAS NUMERO DOS

DVDQ

Solución de ecuaciones diferenciales ordinarias de orden variable.

DEROOT

Integra un sistema de hasta 20 ecuaciones diferenciales ordinarias de primer orden de la forma

$$\frac{DY(I)}{DT} = f(T,Y(1),\ldots,Y(NEQM))$$

Y(I) dada al tiempo T.

INTERP
STEP
ROOT

QUADS2	Aproxima la integral de la función f(x) entre ciertos límites definidos por el tipo de cuadratura especificada.
GAUSSQ	Calculada nodos y coeficientes para la aproximación de una integral usando el método de la cuadratura. Aplicable en aproximaciones a la integral $\int_A^B f(x)W(X)DX \approx \sum_{j=1}^{N} W_j F(T_j)$ SOLVE CLASS IMTQL2
SQUANK	Integra la función FUN por el método de la cuadratura de SIMPSON ("sin ruido").
GEAR	Conjunto de subrutinas empleadas para la solución de sistemas de ecuaciones diferenciales ordinarias. INTERP STIFF COSET PSET DEC SOL
DCADRE	Integra la función f(x) entre los límites A y B usando el método de extrapolación de Romberg. UERTST
ODE	Integra un sistema de NEQN ecuaciones diferenciales ordinarias de primer orden. DE INTRP STEP
DCSQDU	Integra una función "spline" cúbica entre los puntos A y B. UERTST
DBCEVU	Evalúa derivadas parciales mixtas utilizando "splines" bicúbicos. UERTST

PAQUETE DE SUBRUTINAS NUMERO TRES

RSGAB

Calcular valores característicos y vectores característicos del sistema ABX = (LAMBDA)X.

donde A - matriz real y simétrica
B - matriz real y simétrica definida positivamente.
X - vector característico
LAMBA - valor característico

REDUC2
TRED1
TQLRAT
TRED2
TQL2
REBAK

RSGBA

Calcula valores característicos y vectores característicos del sistema real y simétrico
BAX = (LAMBDA)X.

donde A - matriz real y simétrica
B - matriz real y simétrica definida positivamente
X - vector característico
LAMBDA - valor característico

REDUC2
TRED1
TQLRAT
TRED2
TQL2
REBAKB

RSG

Calcula valores característicos y vectores característicos del sistema real y simétrico
AX = (LAMBDA)BX

REDUC
TRED1
TQLRAT
TRED2
TQL2
REBAK

HSITIM

Mejora el vector solución del sistema lineal AX = B por un método iterativo.

SOLVEH

WTBND	Pondera los elementos del sistema lineal AX = B. CHLBND ENDSL1 CBNDJM
SOLVEH	Usa el método de la decomposición de "Householder" en la matriz A = Q * U (Q ortogonal, U triangular superior) para encontrar el vector que minimiza la norma euclidiana de (AX - B).
HDEC	Usa transformación "Householder" en la decomposición de una matriz no singular A en A = QU.
WTVEC	Pondera los elementos del sistema lineal AX=B con la matriz diagonal D, tal que DAX = DB.
WTMTX	Pondera los elementos del sistema lineal AX = B siguiendo el método de Cholesky. CHLSKY CHSLV1 CHITIM
LSDCMP	Dada la matriz A, define las matrices triangular inferior y superior L y U, y una matriz permutación P tal que A = PLU o forma canónica de la matriz A.
SVD	Determina la "decomposición singular" A = USV de una matriz real.
RS	Encuentra valores y vectores característicos de una matriz real simétrica. TRED1 TQLRAT TRED2 TOL2
RG	Encuentra valores y vectores característicos de una matriz real general. BALANC ELMHES HQR ELTRAN HQR2 BALBAK

RT	Encuentra valores y vectores característicos de una matriz real tridiagonal. FIGI IMTQL1 FIGI2 IMTQL2
RSP	Encuentra valores y vectores característicos de una matriz real "compacta" y simétrica. TRED3 TQLRAT TQI2 TRBAK3
RSB	Encuentra valores y vectores característicos de una matriz real en "banda" y simétrica. BANDR TQLRAT TQL2
RST	Calcula valores y vectores característicos de una matriz real, simétrica y tridiagonal. IMTQL1 IMTQL2
LINSY2	Encuentra la solución del sistema de ecuaciones AX = B. DECMP2 SOLVE2 IMPRV2
DPPUT y IPTOLT	Ejecuta conjuntamente acumulaciones de productos escalares de dos vectores.
CG	Calcula valores y vectores característicos de una matriz compleja general. CBAL CORTH COMQR COMQR2 CBABK2

CHLBND	Expresa una matriz A, positivamente definida, simétrica y en banda, como el producto de una matriz triangular inferior en banda y su transpuesta A = L L*
BNDSL1	Resuelve el sistema lineal AX = B donde A es una matriz real, triangular inferior y en banda.
CBNDIM	Por medio de un proceso iterativo, el vector solución del problema anterior, es mejorado.
CHSLV1	Resuelve el sistema lineal AX = B donde A es una matriz triangular inferior.
CHITIM	Por medio de un proceso iterativo, el vector solución del problema anterior, es mejorado.
CHLSKY	Particiona la matriz A real, simétrica y definida positivamente, en el producto de una matriz triangular inferior y su transpuesta A = L L*.

PAQUETE DE SUBRUTINAS NUMERO CUATRO

MNSTD	Calcula la media aritmética y la desviación standard insezgada de los renglones de la matriz A.
REGRSN	Calcula los coeficientes de regresión del sistema lineal AX = B.

MNSTD
WTVEC
COVARC
WTMTX
WTBND*
HDEC*
SOLVEH
HSITIM

DISCR	Calcula un conjunto de funciones lineales las cuales sirven de índices en la clasificación de un individuo en distintos grupos (análisis de discriminación).

CORRE	Calcula medias, desviaciones standard, sumas de productos cruzados de desviaciones y coeficientes de correlación.
CANOR	Calcula la correlación canónica entre dos conjuntos de datos. MINV * NROOT* EIGEN*
URAND	Generador de números aleatorios uniformemente distribuidos (U(-0.5, 0.5). Recomendado sobre RANDU (versión de IBM).
RLONE	Analiza un modelo simple de regresión lineal. MDBET MDBETI RLPRD1 UERTST
RLMUL	Analiza un modelo múltiple de regresión lineal. LUELMP MDBET MDBETI UERTST VMULFS
USPLH	Grafica en la impresora hasta 10 funciones. UERTST USMNMX *
USHIST	Imprime histogramas. UERTST
USHIUT	Imprime histogramas. Permite escribir dos frecuencias/barra. UERTST
RLEAP	Determina un cierto número de regresiones óptimas basadas en subconjuntos de datos pertenecientes a aquel conjunto de datos de una regresión total. MDFD RLEAP1 RLEAP2 RLEAP3 UERTST

USLEAP	Imprime las regresiones óptimas calculadas por RLEAP.
GGAMA	Genera pseudo-números aleatorios con distribución Gama (A,1). También puede usarse en la generación de números aleatorios con distribución exponencial, chi-cuadrada, chi, beta, t, y F. GGNOR GGUB MERFI MONRIS MERFCI UERTST
GFIT	Prueba chi-cuadrada de bondad en el ajuste de ciertos datos a la distribución Gaussiana. MDCDFI UERTST
FTAUTO	Dada una serie de tiempo calcula: 1. Media y variancia 2. Autocovariancias 3. Autocovariancias y autocorrelaciones 4. Autocorrelaciones parciales
CTRBYC	Analiza tablas de contingencia. UERTST
BEMIRI	Calcula medias, coeficientes de regresión con medidas de error y desviaciones standard para arreglos que puedan contener valores "perdidos". UERTST
MDTPOS	Estima la densidad de probabilidad y la distribución acumulativa de una variable aleatoria con distribución Poisson.
MDTD	Calcula valores de la distribución de probabilidad t "student". UERTST
MDCDFI	Calcula valores de la distribución de probabilidad chi-cuadrada. UERTST

MDBIN	Calcula valores de la distribución de probabilidad binomial. UERTST
MDFI	Calcula valores de la distribución inversa de probabilidad F. MDBETA MDBETI UERTST
MDFDRE	Distribución de probabilidad F. MDBETA UERTST
MD	Calcula valores de la distribución de probabilidad F. UERTST
BEIUGR	Estima parámetros estadísticos en datos desagrupados. UERTST
BEIGRP	Estima parámetros estadísticos en datos agrupados. UERTST VABMX$ * VABMXF
ALSQAN	Analiza datos en "diseño cuadrado latino". UERTST

BIBLIOGRAFIA

1. Amyx, J.W., Bass, D.M. Jr., and Whiting, R.L., 1960. Petroleum Reservoir Engineering. McGraw-Hill.

2. Baxendell, P.B., and Thomas, R., 1961. The Calculation of Pressure Gradients in High Rate Flowing Wells. Journal of Pet. Tech.

3. Beckmann, P., 1973. Orthogonal Polynomials for Engineers and Physicists. The Golem Press. Boulder, Colorado.

4. Carnahan, B., Luther, H.A., and Wilkes, J.O., 1969. Applied Numerical Methods. John Wiley & Sons.

5. Forsythe, G.E., Malcolm, M.A., and Moler, C.B., 1977. Computer Methods for Mathematical Computations. Prentice-Hall.

6. Gill, W.N., and Scher, M., 1961. A Modification of the Momentum Transport Hypothesis, A.I.Ch.E. Journal, 7, 61-65.

7. Isaacson, E., and Keller, H.B., 1966. Analysis of Numerical Methods. John Wiley & Sons.

8. Journel, A.G., and Huijbregts, Ch. J., 1978. Mining Geostatistics. Academic Press, London.

9. Krilov, V.I., 1962. Approximate Calculation of Integrals. Macmillan, New York.

10. McCraken, D.D., 1972. A Guide to Fortran IV Programming. Second Edition. John Wiley & Sons.

11. Poettmann, F.H., and Carpenter, P.G., 1952. The Multiphase Flow of Gas, Oil and Water Through Vertical Flow Strings with Application to the Design of Gas Lift Installations. Drill and Prod. Proc., API.

12. Ralston A., 1965. A First Course in Numerical Analysis. McGraw-Hill.

13. Remson, I., Hornberger, G.M., and Molz, F. J., 1971. Numerical Methods in Subsurface Hydrology. Wiley-Interscience.

14. Sóbol, I. M., 1976. Método de Monte Carlo. Lecciones Populares de Matemáticas. MIR, Moscú.

15. Stewart, G.W., 1973. Introduction to Matrix Computations. Academic Press, New York.

16. Tomás, L.J., 1974. Transporte de Gas en Régimen Permanente. Proyecto D-341A. Publication No. 74BII/164. Instituto Mexicano del Petróleo.